USB Mass Storage Host Library for Analog Devices ADSP-SC58x/7x User's Guide Revision 1.12

Closed Loop Design, LLC

support@cld-llc.com

Table of Contents

Disclaimer

This software is supplied "AS IS" without any warranties, express, implied or statutory, including but not limited to the implied warranties of fitness for purpose, satisfactory quality and non-infringement. Closed Loop Design LLC extends you a royalty-free right to use, reproduce, and distribute executable files created using this software for use on Analog Devices ADSP-SC5xx family processors only. Nothing else gives you the right to use this software.

Introduction

The Closed Loop Design (CLD) SC5xx USB Mass Storage Host Library (CLD SC5xx USB Library) creates a simplified interface for developing a USB Host supporting the USB Mass Storage Device Class Bulk-Only Transport using a USB port of the Analog Devices ADSP-SC5xx. The CLD SC5xx USB Library also includes timer functions that facilitate creating timed events quickly and easily. The library's User application interface is comprised of parameters used to customize the library's functionality as well as callback functions used to notify the User application of events. These parameters and functions are described in greater detail in the CLD SC5xx USB Library API section of this document.

USB Background

In order to take advantage of the CLD SC5xx USB Library you will need at least a basic understanding of the USB 2.0 protocol, and the Mass Storage Class Bulk-Only Transport. Additionally, the Mass Storage Class uses SCSI reduced block commands, so familiarity with the SCSI message structures is also required. The following are some resources to refer to when working with USB, Mass Storage Class Bulk-Only Transport, and SCSI reduced block commands

- The USB 2.0 Specification
- Mass Storage Class Specification Overview v1.4
- Mass Storage Bulk Only v1.0
- USB in a Nutshell: A free online wiki that explains USB concepts. http://www.beyondlogic.org/usbnutshell/usb1.shtml
- "USB Complete" by Jan Axelson ISBN: 1931448086
- “USB Mass Storage” by Jan Axelson ISBN: 9781931448048

Dependencies

In order to function properly, the CLD SC5xx USB Library requires the following resources:

- 24Mhz clock input connected to the SC5xx USB_CLKIN pin.
- The User firmware is responsible for configuring all other non-USB specific peripherals, including clocks, power modes, etc.

CLD SC5xx USB Library Scope and Intended Use

CLD SC5xx USB Library implements the required functionality to implement a USB Mass Storage Device Bulk-Only Host, as well as providing time measurements functionality. The USB Host support having a single USB Mass Storage Device connected at a time, and does not support USB Hubs. The CLD SC5xx USB Library is designed to be added to an existing User project, and as such only includes the functionality needed to implement the above-mentioned USB, and timer keeping features. All other aspects of SC5xx processor configuration must be implemented by the User code.

CLD MSD USB Host Example Description

The CLD_SC58x_MSD_USB_Host_Ex_v1_12 and CLD_SC57x_MSD_USB_Host_Ex_v1_12 projects provided with the CLD SC5xx USB Library implement a Mass Storage Device USB Host supporting the FAT file system. The example projects use the FatFS Generic FAT Filesystem Module to support the FAT filesystem. The CLD_SC58x_MSD_USB_Host_Ex_v1_12example is designed to run on the ADSP-SC589, while the CLD_SC57x_MSD_USB_Host_Ex_v1_12 project is designed to run on the ADSP-SC573 Ez-Board.

In order to use the USB1 port of the SC589 on the EZ-Board you will need to connect a GPIO pin to the EN input (Pin 1) of U60. For the example project EN was connected to Pin 9 of the P2 connector (GPIO port E pin 11). You will also need to configure the CLD library so it uses the selected GPIO pin as shown below:

- use_built_in_vbus_ctrl = CLD_FALSE
- vbus_en_port = CLD_GPIO_PORT_x
- vbus_en_pin = CLD_GPIO_PIN_x

The example project uses the SC589/573 EZ-Board's USB to Serial converter connected to a serial terminal on your PC to exercise the FatFs functionality. The table below lists the supported FatFS functionality and the corresponding control characters.

Key	Thumb drive operation
M or m	Mounts the drive
D or d	Reads the attached thumb drives directory structure and outputs it to the serial terminal.
U or u	Unmounts the drive
W or w	Opens the msd_test.txt file on the thumb drive if it exists, and writes "Fat file write test" to the text file.
R or r	Opens the msd_test.txt file on the thumb drive if it exists, and reads the test string written by the above 'W' command.

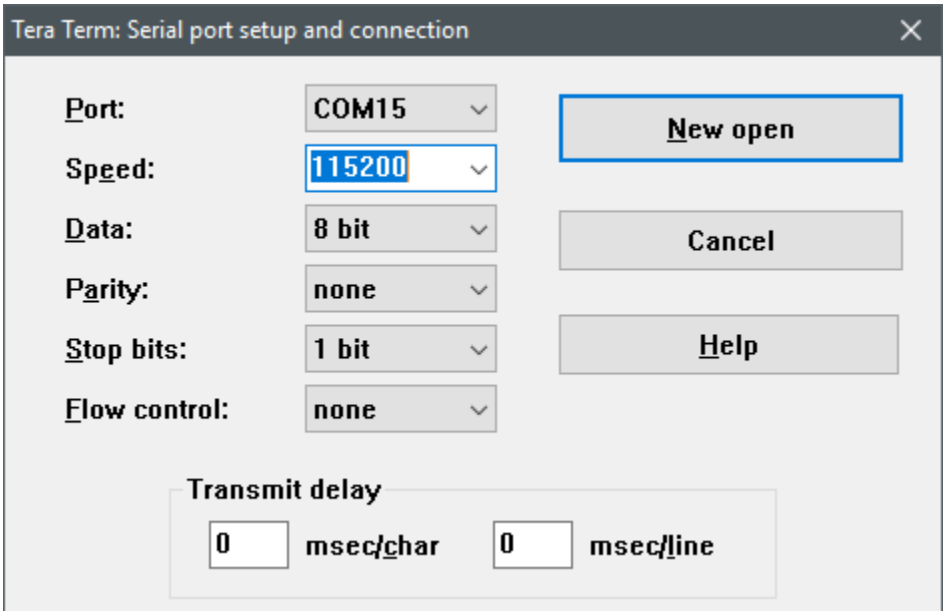
Also, the FatFS module uses dynamic memory allocation (malloc and free) when accessing the Mass Storage Device memory. If you configure the CLD library to use DMA, make sure that the SC5xx Core 0 heap is located in non-cached memory.

Running the Example Project

1. The SC589 example projects toggles the LED connected to GPIO port E pin 13 (GPIO port E pin 14 for the SC573 example) every 250 milliseconds to provide a visual indicator the project is running.

1. Once the example project is running on the EZ Board connect a USB cable from a PC to the USB port labeled Terminal in the Using the ADSP-SC589 and ADSP-SC573 Ez-Board. If the operating system fails to install the necessary FTDI drivers please consult the FTDI website for drivers.

2. Using TeraTerm, or another serial terminal program, connect to the FTDI serial port as shown below and click New Open:



3. The example project will output status messages to the terminal window with a USB mass storage device is connected. The commands listed in the previous table may be used to interface with the drive (as long as it is formatted using the FAT file system).

CLD SC5xx Library API

The following CLD library API descriptions include callback functions that are called by the library based on USB events. The following color code is used to identify if the callback function is called from the USB interrupt service routine, or from mainline. The callback functions called from the USB interrupt service routine are also italicized so they can be identified when printed in black and white.

Callback called from the mainline context
Callback called from the USB interrupt service routine

cld_sc5xx_msd_host_lib_init

```
CLD_RV cld_sc5xx_msd_host_lib_init (CLD_SC5xx_MSD_Host_Lib_Init_Params *
p_lib_params)
```

Initializes the CLD SC5xx USB Library.

Arguments

p_lib_params	Pointer to a CLD_SC5xx_MSD_Host_Lib_Init_Params structure that has been initialized with the User Application specific data.

Return Value

This function returns the CLD_RV type which represents the status of the CLD SC5xx USB Library initialization process. The CLD_RV type has the following values:

CLD_SUCCESS	The library was initialized successfully
CLD_FAIL	There was a problem initializing the library
CLD_ONGOING	The library initialization is being processed

Details

The cld_sc5xx_msd_host_lib_init function is called as part of the device initialization and must be repeatedly called until the function returns CLD_SUCCESS or CLD_FAIL. If CLD_FAIL is returned the library will report an error status identifying the cause of the failure using the fp_cld_lib_status function if defined by the User application. Once the library has been initialized successfully the main program loop can start.

The CLD_SC5xx_MSD_Host_Lib_Init_Params structure is described below:

```
typedef struct
{
    CLD_USB_Port    usb_port
    CLD_Boolean     enable_dma;

    CLD_Boolean     use_built_in_vbus_ctrl;
    CLD_Boolean     vbus_ctrl_open_drain;
    CLD_Boolean     vbus_ctrl_inverted;
    CLD_GPIO_Port   vbus_en_port;
    CLD_GPIO_PIN    vbus_en_pin;
```

```
    void (*fp_cld_usb_event_callback) (CLD_USB_Event event);

    void (*fp_cld_lib_status) (unsigned short status_code,
                               void * p_additional_data,
                               unsigned short additional_data_size);

} CLD_SC5xx_MSD_Host_Lib_Init_Params;
```

A description of the CLD_SC5xx_MSD_Host_Lib_Init_Params structure elements is included below:

<table>
<tr><th>Structure Element</th><th>Description</th></tr>
<tr><td>usb_port</td><td>Specifies which of the SC5xx USB Ports the library should use.</td></tr>
<tr><td>enable_dma</td><td>Used to enable/disable USB DMA support.
When set to CLD_TRUE DMA is enabled for transfers larger than 32 bytes that are aligned to a 4-byte boundary.

Note: When DMA is enabled make sure the data buffers are located in un-cached memory to avoid cache coherency issues.</td></tr>
<tr><td>use_built_in_vbus_ctrl</td><td>Used to select if the SC5xx USB VBC output is used to control the external Vbus switch.
- CLD_TRUE = Use VBC signal (Not available on USB 1 of the SC58x)
- CLD_FALSE = Use GPIO pin specified by the vbus_en_port & vbus_en_pin parameters.</td></tr>
<tr><td>vbus_ctrl_open_drain</td><td>When use_built_in_vbus_ctrl = CLD_TRUE this parameter selects if VBC output is configured as open drain.</td></tr>
<tr><td>vbus_ctrl_inverted</td><td>Selects the polarity of the Vbus control.
- CLD_TRUE = Vbus enable is active high
- CLD_FALSE = Vbus enable is active low</td></tr>
<tr><td>vbus_en_port</td><td>When use_built_in_vbus_ctrl = CLD_FALSE this parameter selects GPIO port used to control Vbus.</td></tr>
<tr><td>vbus_en_pin</td><td>When use_built_in_vbus_ctrl = CLD_FALSE this parameter selects GPIO pin used to control Vbus.</td></tr>
<tr><td>fp_cld_usb_event_callback</td><td>Function that is called when one of the following USB events occurs. This function has a single CLD_USB_Event parameter.

Note: This callback can be called from the USB interrupt or mainline context depending on which USB event was detected. The CLD_USB_Event values in the table below are highlighted to show the context the callback is called for each event.

The CLD_USB_Event has the following values:
<table>
<tr><th>Return Value</th><th>Description</th></tr>
<tr><td>CLD_USB_ENUMERATED_CONFIGURED_HS</td><td>High-Speed USB Mass Storage Device enumerated (USB Configuration set to a non-zero value)</td></tr>
<tr><td>CLD_USB_ENUMERATED_CONFIGURED_FS</td><td>Full-Speed USB Mass Storage Device</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td></td><td>enumerated (USB Configuration set to a non-zero value)</td></tr>
<tr><td>CLD_USB_MSD_DISCONNECTED</td><td>Mass Storage Device removed</td></tr>
</table></td></tr>
<tr><td>fp_cld_lib_status</td><td>Pointer to the function that is called when the CLD library has a status to report. This function has the following parameters:

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>status_code</td><td>16-bit status code. If the most significant bit is a '1' the status being reported is an Error.</td></tr>
<tr><td>p_additional_data</td><td>Pointer to additional data included with the status.</td></tr>
<tr><td>additional_data_size</td><td>The number of bytes in the specified additional data.</td></tr>
</table>
If the User plans on processing outside of the fp_cld_lib_status function they will need to copy the additional data to a User buffer.</td></tr>
</table>

cld_sc5xx_msd_host_lib_main

```
void cld_sc5xx_msd_host_lib_main (void)
```

CLD SC5xx USB Library mainline function

Arguments

None

Return Value

None.

Details

The cld_sc5xx_msd_host_lib_main function is the CLD SC5xx USB Library mainline function that must be called in every iteration of the main program loop in order for the library to function properly. Alternatively, it can be called from a timer interrupt, but must be at a lower priority then the USB interrupt, and 125 microsecond timer interrupt used to call the cld_time_125us_tick function.

How often the cld_sc5xx_msd_host_lib_main is given runtime impacts the USB performance, so it is recommended to evaluate USB the performance and adjust how frequently the function is called as desired.

cld_sc5xx_msd_host_lib_send_command

```
CLD_RV cld_sc5xx_msd_host_lib_send_command (CLD_SC5xx_MSD_Host_Cmd_Params *
p_params)
```

CLD SC5xx USB MSD Host Library function used to execute a Bulk-Only SCSI command to an attached Mass Storage Device.

Arguments

p_params	Pointer to a CLD_SC5xx_MSD_Host_Cmd_Params structure used to describe the SCSI command being executed.

Return Value

This function returns the CLD_RV type which reports if the requested command was initiated. The CLD_RV type has the following value used by this function:

CLD_SUCCESS	The library has started the requested command.
CLD_FAIL	The library failed to start the requested command. This will happen if a previously requested command is still being processed, a Mass Storage Device isn't connected, or if one of the command parameters is invalid.

Details

The cld_sc5xx_msd_host_lib_send_command function executes the Mass Storage Bulk-Only SCSI command specified by the p_params parameter to the attached mass storage device using Bulk IN/OUT endpoints.

The CLD_SC5xx_MSD_Host_Cmd_Params structure is described below.

```
typedef struct
{
    CLD_SC5xx_MSD_Host_Commands cmd;
    CLD_SC5xx_MSD_Host_Commands cmd_params;
    unsigned long data_transport_size;
    unsigned char * p_data_transport;
    void (*fp_cmd_successful_callback) (unsigned long data_transport_size);
    void (*fp_cmd_failed_callback) (CLD_SC5xx_MSD_Host_Cmd_Status status);
} CLD_SC5xx_MSD_Host_Cmd_Params;
```

A description of the CLD_SC5xx_MSD_Host_Cmd_Params structure elements is included below:

Structure Element	Description
cmd	The Mass Storage Device Bulk-Only transport SCSI command to process.
cmd_params	Union of command specific parameters. The User should initialize the parameters for the command specified by cmd. For more information about the various command specific parameters refer to the SCSI command documentation.

<table>
<tr><td>data_transport_size</td><td>The number of bytes to transfer during the data transport stage of the command.</td></tr>
<tr><td>p_data_transport</td><td>Pointer to the memory location to read or write the data during the data transport stage of the command.</td></tr>
<tr><td>fp_cmd_successful_callback</td><td>Function called when the specified command has been completed. The callback is passed the number of bytes transferred during the data transport stage, since it can be less than the number of bytes the User specified. This function pointer can be set to CLD_NULL if the User application doesn't want to be notified when the command is complete.</td></tr>
<tr><td>fp_cmd_failed_callback</td><td>Function called if there is a problem completing the specified command, and is passed the reason for the failure. This function can be set to CLD_NULL if the User application doesn't want to be notified if a problem occurs.

Below are the possible values for CLD_SC5xx_MSD_Host_Cmd_Status passed to the fp_cmd_failled_callback.

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>COMMAND_FAILED</td><td>Device reported the command failed in the Command Status Wrapper.</td></tr>
<tr><td>COMMAND_PHASE_ERROR</td><td>Device reported a phase error in the Command Status Wrapper.</td></tr>
<tr><td>NAK_LIMIT_REACHED</td><td>The Device NAKed the Host longer then was allowed.</td></tr>
</table>
</td></tr>
</table>

cld_sc5xx_msd_host_lib_abort_command

```
void cld_sc5xx_msd_host_lib_abort_command (void)
```

CLD SC5xx USB Library function used to abort a command requested using the cld_sc5xx_msd_host_lib_send_command function.

Arguments

None

Return Value

None

Details

The cld_sc5xx_msd_host_lib_abort_command will attempt to abort an active Mass Storage Device command.

cld_sc5xx_msd_host_lib_get_max_lun

```
CLD_RV cld_sc5xx_msd_host_lib_get_max_lun
        (CLD_SC5xx_MSD_Host_Get_Max_Lun_Params * p_params)
```

CLD SC5xx USB Library function used request the Max Lun of the attached Mass Storage Device.

Arguments

p_params	Pointer to a CLD_SC5xx_MSD_Host_Get_Max_Lun_Params structure used by the Get Lun request.

Return Value

This function returns the CLD_RV type which reports if the Get Max Lun request was scheduled successfully. The CLD_RV type has the following values:

CLD_SUCCESS	The library has started the requested Get Max Lun request.
CLD_FAIL	The library failed to start the requested Get Max Lun request. This will happen if a Mass Storage Device isn't attached, or if the p_params->p_max_lun isn't defined.

Details

The cld_sc5xx_msd_host_lib_get_max_lun function transmits the Get Max Lun request using the specified p_params parameter.

The CLD_SC5xx_MSD_Host_Get_Max_Lun_Params structure is described below.

```
typedef struct
{
    unsigned char * p_max_lun;
    void (*fp_cmd_successful_callback) (void);
    void (*fp_cmd_failed_callback)
            (CLD_SC5xx_MSD_Host_Ctrl_Req_Status reason);
} CLD_SC5xx_MSD_Host_Get_Max_Lun_Params;
```

A description of the CLD_SC5xx_MSD_Host_Get_Max_Lun_Params structure elements is included below:

Structure Element	Description
p_max_lun	Pointer to the memory location to store the max lun value returned by the attached Device.
fp_cmd_successful_callback	Function called when the Get Max Lun command has been completed. This function pointer can be set to CLD_NULL if the User application doesn't want to be notified when the command has been completed.
fp_cmd_failed_callback	Function called if there is a problem processing the Get Max Lun request. This function can be set to CLD_NULL if the User application doesn't want to be notified if a problem occurs.

	The fp_cmd_failed_callback function is passed the reason for the failure. A description of the CLD_SC5xx_MSD_Host_Ctrl_Req_Status values is below.

Value	Description
CTRL_REQ_STALLED	The Device Stalled the control endpoint.
CTRL_REQ_NAK_LIMIT	The Nak limit was reached
CTRL_REQ_TIMEOUT	The Host detected a timeout condition.
CTRL_REQ_ERROR_READING_FX_FIFO	There was an error reading the Max Lun value.

cld_sc5xx_msd_host_lib_bulk_only_reset

```
CLD_RV cld_sc5xx_msd_host_lib_bulk_only_reset
        (CLD_SC5xx_MSD_Host_Bulk_Only_Reset_Params * p_params)
```

CLD SC5xx USB Library function used request the device to perform a Bulk-Only Reset. This can be used to resync with the attached Mass Storage Device.

Arguments

p_params	Pointer to a CLD_SC5xx_MSD_Host_Bulk_Only_Reset_Paramsstructure used to process the request.

Return Value

This function returns the CLD_RV type which reports if the Bulk-Only Reset request was scheduled successfully. The CLD_RV type has the following values:

CLD_SUCCESS	The library has started the requested Bulk-Only Reset command.
CLD_FAIL	The library failed to start the requested Bulk-Only Reset request. This will happen if a Mass Storage Device isn't attached.

Details

The cld_sc5xx_msd_host_lib_bulk_only_reset function transmits the Bulk-Only Reset request using the specified p_params parameter.

The CLD_SC5xx_MSD_Host_Bulk_Only_Reset_Params structure is described below.

```
typedef struct
{
    void (*fp_cmd_successful_callback) (void);
    void (*fp_cmd_failed_callback)
            (CLD_SC5xx_MSD_Host_Ctrl_Req_Status reason);
} CLD_SC5xx_MSD_Host_Bulk_Only_Reset_Params;
```

A description of the CLD_SC5xx_MSD_Host_Bulk_Only_Reset_Params structure elements is included below:

<table>
<tr><th>Structure Element</th><th>Description</th></tr>
<tr><td><code>fp_cmd_successful_callback</code></td><td>Function called when the Bulk-Only Reset command has been completed. This function pointer can be set to CLD_NULL if the User application doesn't want to be notified when the command has been completed.</td></tr>
<tr><td>fp_cmd_failed_callback</td><td>Function called if there is a problem processing the Bulk-Only Reset request. This function can be set to CLD_NULL if the User application doesn't want to be notified if a problem occurs.

The fp_cmd_failed_callback function is passed the reason for the failure. A description of the CLD_SC5xx_MSD_Host_Ctrl_Req_Status values is below.

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CTRL_REQ_STALLED</td><td>The Device Stalled the control endpoint.</td></tr>
<tr><td>CTRL_REQ_NAK_LIMIT</td><td>The Nak limit was reached</td></tr>
<tr><td>CTRL_REQ_TIMEOUT</td><td>The Host detected a timeout condition.</td></tr>
<tr><td>CTRL_REQ_ERROR_READING_FX_FIFO</td><td>There was an error reading the Max Lun value.</td></tr>
</table>
</td></tr>
</table>

cld_time_125us_tick

```
void cld_time_125us_tick (void)
```

CLD SC5xx USB Library timer function that should be called once per 125 microseconds.

Arguments

None

Return Value

None.

Details

This function should be called once every 125 microseconds in order to the CLD to processed periodic events.

cld_usb0_isr_callback

```
void cld_usb0_isr_callback (void)
```

CLD SC5xx USB Library USB interrupt service routine for the USB0 port.

Arguments
None

Return Value
None.

Details
These USB ISR functions should be called from the corresponding SC5xx USB Port Interrupt Service Routines as shown in the CLD provided example projects.

cld_usb1_isr_callback

```
void cld_usb1_isr_callback (void)
```

CLD SC5xx USB Library USB interrupt service routine for the USB1 port

Arguments
None

Return Value
None.

Details
These USB ISR functions should be called from the corresponding SC5xx USB Port Interrupt Service Routines as shown in the CLD provided example projects.

cld_time_get

```
CLD_Time cld_time_get(void)
```

CLD SC5xx USB Library function used to get the current CLD time in milliseconds.

Arguments
None

Return Value
The current CLD library time.

Details
The cld_time_get function is used in conjunction with the cld_time_passed_ms function to measure how much time has passed between the cld_time_get and the cld_time_passed_ms function calls in milliseconds.

cld_time_passed_ms

```
CLD_Time cld_time_passed_ms(CLD_Time time)
```

CLD SC5xx USB Library function used to measure the amount of time that has passed in milliseconds.

Arguments

time	A CLD_Time value returned by a cld_time_get function call.

Return Value

The number of milliseconds that have passed since the cld_time_get function call that returned the CLD_Time value passed to the cld_time_passed_ms function.

Details

The cld_time_passed_ms function is used in conjunction with the cld_time_get function to measure how much time has passed between the cld_time_get and the cld_time_passed_ms function calls in milliseconds.

cld_time_get_125us

```
CLD_Time cld_time_get_125us(void)
```

CLD SC5xx USB Library function used to get the current CLD time in 125 microsecond increments.

Arguments

None

Return Value

The current CLD library time.

Details

The cld_time_get_125us function is used in conjunction with the cld_time_passed_125us function to measure how much time has passed between the cld_time_get_125us and the cld_time_passed_125us function calls in 125 microsecond increments.

cld_time_passed_125us

```
CLD_Time cld_time_passed_125us(CLD_Time time)
```

CLD SC5xx USB Library function used to measure the amount of time that has passed in 125 microsecond increments.

Arguments

time	A CLD_Time value returned by a cld_time_get_125us function call.

Return Value

The number of 125microsecond increments that have passed since the cld_time_get_125us function call that returned the CLD_Time value passed to the cld_time_passed_125us function.

Details

The cld_time_passed_125us function is used in conjunction with the cld_time_get_125us function to measure how much time has passed between the cld_time_get_125us and the cld_time_passed_125us function calls in 125 microsecond increments.

cld_lib_status_decode

```
char * cld_lib_status_decode (unsigned short status_cod,
                              void * p_additional_data,
                              unsigned short additional_data_size)
```

CLD Library function that returns a NULL terminated string describing the status passed to the function.

Arguments

status_code	16-bit status code returned by the CLD library. Note: If the most significant bit is a '1' the status is an error.
p_additional_data	Pointer to the additional data returned by the CLD library (if any).
additional_data_size	Size of the additional data returned by the CLD library.

Return Value

This function returns a decoded Null terminated ASCII string.

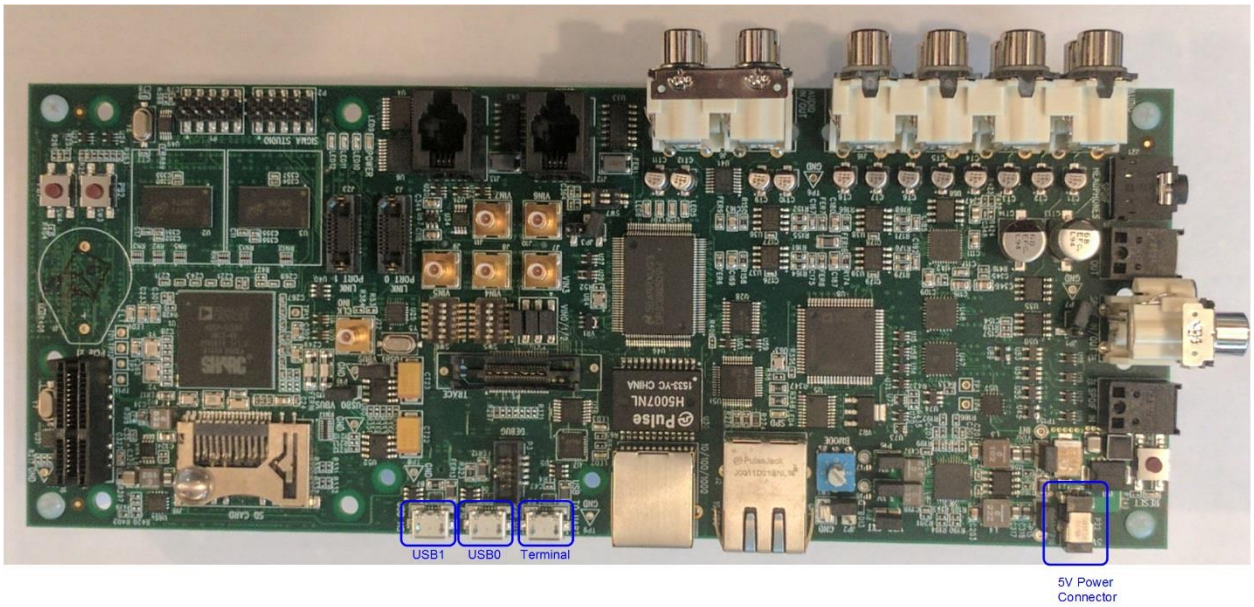
Details

The cld_lib_status_decode function can be used to generate an ASCII string which describes the CLD library status passed to the function. The resulting string can be used by the User to determine the meaning of the status codes returned by the CLD library.

Using the ADSP-SC589 and ADSP-SC573 Ez-Board

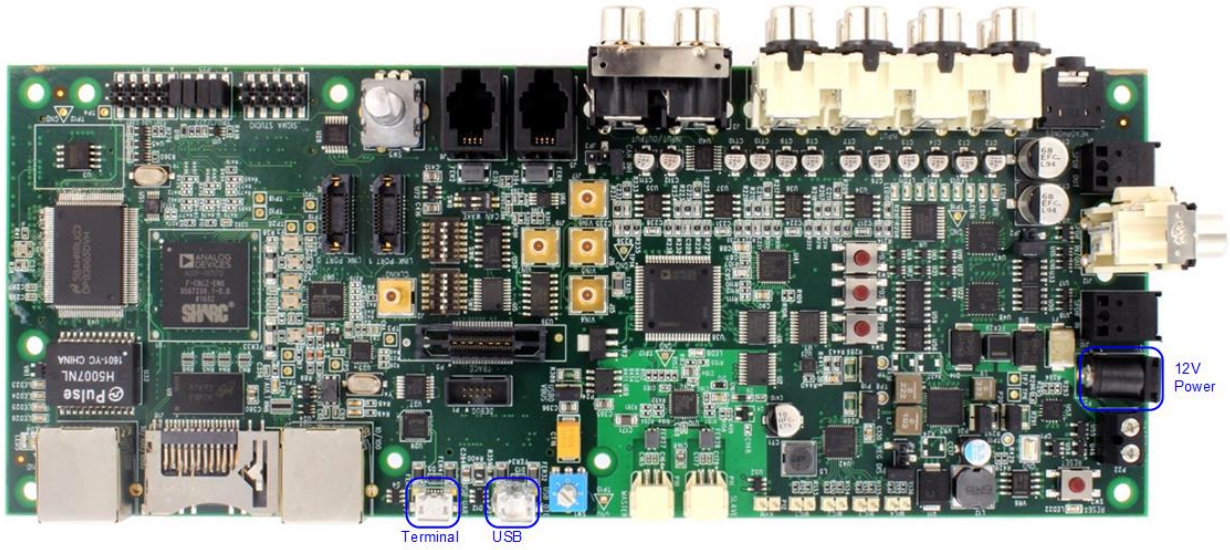
SC589 Connections:

Blue circled USB0 or USB1 connection is used for the example project to connect to a Mass Storage Device using a USB Micro-B OTG adapter like the one shown below. The Terminal connector is used to input the commands descripted in the CLD MSD USB Host Example Description section of this document.



SC573 Connections:

Blue circled USB connection is used for the example project to connect to a Mass Storage Device using a USB Micro-B OTG adapter like the one shown below. The Terminal connector is used to input the commands descripted in the CLD MSD USB Host Example Description section of this document.



USB On-the-Go (OTG) Adapter

Note: Make sure the OTG adapter you use grounds the USB ID pin.

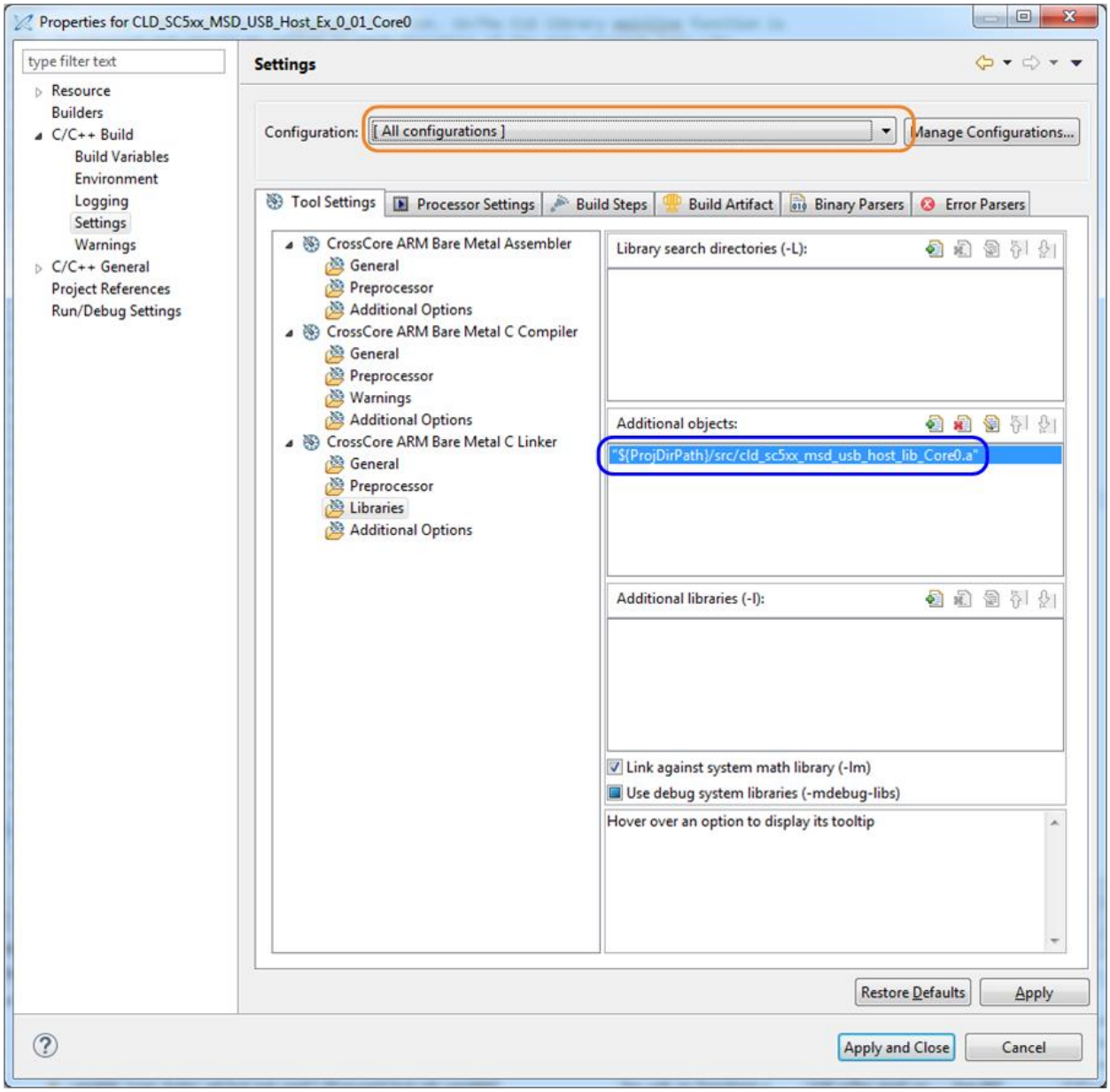
Adding the CLD SC5xx USB Library to an Existing CrossCore Embedded Studio Project

In order to include the CLD SC5xx USB Library in a CrossCore Embedded Studio (CCES) project you must configure the project linker settings so it can locate the library. The following steps outline how this is done.

1. Copy the cld_sc5xx_msd_usb_host_lib.h, cld_sc5xx_msd_defs.h, and cld_sc5xx_msd_usb_host_lib_Core0.a files to the project's src directory.
2. Open the project in CrossCore Embedded Studio.
3. Right click the project in the 'C/C++ Projects' window and select Properties.

   If you cannot find the 'C/C++ Projects" window make sure C/C++ Perspective is active. If the C/C++ Perspective is active and you still cannot locate the 'C/C++ Projects' window select Window → Show View → C/C++ Projects.
4. You should now see a project properties window similar to the one shown below.

   Navigate to the C/C++ Build → Settings page and select the CrossCore ARM Bare Metal C Linker's Libraries page. The CLD SC5xx USB Library needs to be included in the projects 'Additional objects' as shown in the diagram below (circled in blue). This lets the linker know where the cld_sc5xx_msd_usb_host_lib_Core0.a file is located.



5. The 'Additional objects' setting needs to be set for all configurations (Debug, Release, etc). This can be done individually for each configuration, or all at once by selecting the [All Configurations] option as shown in the previous figure (circled in orange).