libdeci4p Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Purpose and Characteristics

libdeci4p is a library on the Development Kit that is used to support data exchanges with the development host computer.

The library on the development host computer is the Target Manager API. By using the libdeci4p and Target Manager APIs, it is possible to implement connectivity between the applications on the development host computer and the applications on the Development Kit.

Main Features

libdeci4p has the following main features:

- Feature to send/receive data
- Feature to execute applications in development host computers

Resource Consumption

The system resources used by libdeci4p are listed below.

Resource	Description
Footprint	None
Work memory	Use dedicated Development Kit memory
Threads	None
Processor time	Negligible

Embedding in Programs

Include libdeci4p.h in the source program. (A number of header files will also be automatically included.)

When building a program, link libSceDeci4p_stub.a.

Sample Program

The following program is provided as a sample program that uses libdeci4p, so refer to it as needed.

sample_code/developer_tools/api_deci4p/

This is a sample of the basic use of libdeci4p.

Reference Materials

Refer to the following document for details on the library developed on the development host computer.

- Target Manager API Reference

For details on the settings required for executing an application in a development host computer using the libdeci4p features, refer to the following document.

- Neighborhood and Utilities User's Guide

2 Using the Library

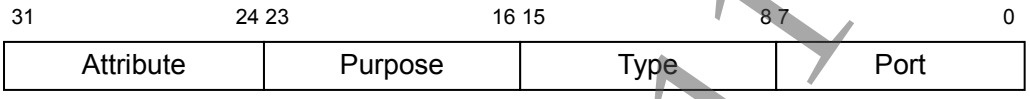
Data Sending/Receiving

This section explains the basic procedure for data sending/receiving with libdeci4p. An outline of the process flow is as follows.

(1) Open the DECI4p socket
(2) Send/receive data to/from the DECI4p socket
(3) Close the DECI4p socket.

Preparations

Shared DECI4p protocol numbers are used by the programs on the development host computer and the programs on the Development Kit. The DECI4p protocol number fields are defined as follows.



Bits 31 to 24 are defined as protocol attributes.

Attribute	Description
0x00	Unicast protocol attribute
0x80	Multicast protocol attribute

Bits 23 to 16 are defined according to the purpose.

Purpose	Description
0x00 - 0x03	For SCE use
0x04 – 0xdf	For Tool & Middleware
0xe0 – 0xef	For local use by a licensee
0xf0 – 0xff	Reserved

Bits 15 to 8 are defined as the protocol type.

Bits 7 to 0 are defined as port numbers. Specify 0 when no port number is used.

Basic Usage Procedure

(1) Open the DECI4p socket

Open the DECI4p socket by calling `sceKernelDeci4pOpen()`. The socket is returned as the initialization result. Set a buffer size larger than the size to be received.

(2) Send/receive data to/from the DECI4p socket

Call `sceKernelDeci4pWrite()` to perform data sending, and call `sceKernelDeci4pRead()` to perform receiving.

(3) Close the DECI4p socket.

When the socket is no longer needed, call `sceKernelDeci4pClose()` to terminate the processing. As a result, the resources granted during initialization are released.

Interruption of processing

If an error occurs during processing or to interrupt the processing through user operation, call `sceKernelDeci4pClose()`.

Registering Callback

By using `sceKernelDeci4pRegisterCallback()` to register a callback (created with `sceKernelCreateCallback()`) to DECI4p socket, it will be possible to receive callback notification when packets are received and when communication channel status changes occur.

Main APIs Used for Sending/Receiving Data

API	Description
`sceKernelDeci4pOpen()`	Opens DECI4p protocol socket
`sceKernelDeci4pClose()`	Closes DECI4p protocol socket
`sceKernelDeci4pWrite()`	Sends data to DECI4p protocol socket
`sceKernelDeci4pRead()`	Receives data from DECI4p protocol socket
`sceKernelDeci4pRegisterCallback()`	Registers a callback function to DECI4p protocol socket

Application Execution in a Development Host Computer

This section explains the basic procedure for executing applications in development host computers with libdeci4p. An outline of the process flow is as follows.

(1) Allow application start in the development host computer from the PlayStation®Vita

(2) Set the application startup parameters

(3) Execute the application in the development host computer

Note that all of the processing related to DECI4p protocol communication in this processing will be concealed in the library, therefore applications do not need to be aware of DECI4p protocols.

For details on the APIs shown in this section, refer to the "libdeci4p Reference" document.

Basic Usage Procedure

(1) Allow application start in the development host computer from the PlayStation®Vita

Set "Allow game to spawn processes on the host PC (hostexec)" to "Enabled" in Preferences in Neighborhood in the development host computer.

It is also possible to execute "`psp2ctrl hostexec enable`" using the psp2ctrl utility.

(2) Set the application startup parameters

Prepare an `SceKernelDeci4pCreateHostProcessParam` structure and set appropriate values in each member in the structure as follows.

For *`pathName`*, set the path name of the application to start beginning with host0:. Both relative paths from the development host computer file server directory and absolute paths can be used. In addition, environment variables can be used as in the following example. If passing a command line argument to the application is required, specify the command line argument for *`cmdLine`* with a character string. In addition, specify appropriate values for *`flags`* and *`workDir`* when specifying the current directory upon application execution is required.

```
SceKernelDeci4pCreateHostProcessParam param;
memset(&param, 0, sizeof(param));
param.pathName = "host0:%WINDIR%\\notepad.exe";
param.cmdLine = NULL;
param.flags = SCE_KERNEL_DECI4P_HOST_PROCESS_WINDOW_SHOW |
              SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_FSROOT;
param.workDir = NULL;
```

(3) Execute the application

Call `sceKernelDeci4pCreateHostProcessAndWait()` to start the application in the development host computer. The caller thread will be blocked until the application exits, and the exit code will be stored in *`hostProcessExitCode`*.

```
SceKernelDeci4pCreateHostProcessResult result;
SceUInt32 hostProcessExitCode;
SceInt32 ret = sceKernelDeci4pCreateHostProcessAndWait(
        &param, &result, &hostProcessExitCode);
if (ret < 0) {
        // Error handling
} else {
        printf("Exit code = 0x%08x\n", hostProcessExitCode);
}
```

Asynchronously executing an application

When it is desired to continue processing without waiting for the application in the development host computer to exit, use `sceKernelDeci4pCreateHostProcess()` instead of `sceKernelDeci4pCreateHostProcessAndWait()`. `sceKernelDeci4pCreateHostProcess()` will return immediately after the application in the development host computer starts without waiting for the application to exit. In addition, by specifying a callback created with `sceKernelCreateCallback()`, it will be possible to receive notification upon application exit.

For details on `sceKernelDeci4pCreateHostProcess()` and related structures, refer to the "libdeci4p Reference" document. For details on `sceKernelCreateCallback()`, refer to the "Kernel Reference" document.

Main APIs Used for Application Execution

API	Description
`SceKernelDeci4pCreateHostProcessParam`	Startup parameters for application in development host computer
`SceKernelDeci4pCreateHostProcessResult`	Startup result for application in development host computer
`SceKernelDeci4pHostProcessExitInfo`	Exit information for application in development host computer
`sceKernelDeci4pCreateHostProcess()`	Asynchronously executes application in development host computer
`sceKernelDeci4pCreateHostProcessAndWait()`	Executes application in development host computer and waits for exit

Obtaining the Host Debugger Connection Status

The host debugger connection status for the calling user process can be obtained with the `sceKernelDeci4pIsProcessAttached()` function.

Enabling/Disabling Data Breakpoints (Watchpoints)

Using the `sceKernelDeci4pDisableWatchpoint()` function, data breakpoints (watchpoints) set with the host debugger can be temporarily disabled. In addition, they can be re-enabled using the `sceKernelDeci4pEnableWatchpoint()` function.

3 Precautions

Limitations

This library can be used only for the Development Kit.

The size of data transferred between the Development Kit and the development host computer must be 66 KB or less.