dsclient-py Documentation

Cameron Webb

Apr 13, 2019

Contents

dsclient-py is a python client for interacting with debugserver-js. Use this module to connect to a running debugserver-js instance and send commands over TCP/IP.

CHAPTER 1

Getting Started

1.1 Requirements

debugserver-js dsclient-py is designed to interact with debugserver-js therefore you'll need to have the debugserver-js installed and an instance running to connect to.

Code Composer Studio Texas Instrument's eclipse based IDE which includes DSS (required by debugserver-js)

1.2 Install

PyPi:

```
pip install dsclient
```

Source:

```
git clone https://github.com/tiflash/dsclient-py
cd dsclient-py
pip install .
```

CHAPTER 2

API Reference

The `dsclient` module provides two classes for interacting with a debugserver-js instance:

DebugServer
DebugSession

Warning: You should not instantiate the `DebugSession` class directly. Instead use the `DebugServer.open_session()` command to obtain a handle to a `DebugSession` object.

2.1 DebugServer

class DebugServer(*host=None*, *port=None*)

DebugServer Class for creating and communicating with DebugServer-js

Initializes DebugServer object

Parameters

- **host** (*str, optional*) – hostname of existing DebugServer to connect to (default="localhost")
- **port** (*int*) – port number of existing DebugServer to connect to

attach_ccs()

Opens a CCS GUI instance for the DebugServer

Raises `Exception` – raises exception if problem opening CCS

create_config(*name*, *connection=None*, *device=None*, *board=None*, *directory=None*)

Creates a ccxml file using the provided parameters

Parameters

- **name** (*str*) – name of ccxml file to create

- **connection** (*str*) – connection name to use (required if board is ommitted)
- **device** (*str*) – devicetype name to use (required if board is ommitted)
- **board** (*str*) – board name to use (required if connection + device ommitted)
- **directory** (*str*) – full path to directory location to place file

get_config()

Get ccxml file in use by DebugServer

Returns ccxml file in use by DebugServer

Return type str

get_list_of_configurations()

Returns list of configuration files

Returns list of configuration files

Return type list

get_list_of_connections()

Returns list of connection names

Returns list of connection names

Return type list

get_list_of_cpus()

Returns list of CPU names

Returns list of CPU names

Return type list

get_list_of_devices()

Returns list of device names

Returns list of device names

Return type list

get_list_of_sessions()

Returns list of open sessions

Returns list of open sessions

Return type list

get_session(*name*)

Returns handle to the open session

Parameters **name** (*str*) – name of open session to retrieve handle for

Returns DebugSession object

Return type *DebugSession*

kill()

Kills Debug Server (including any open sessions)

open_session(*name*)

Open a session for the provided session name

Parameters **name** (*str*) – session name to open

Returns DebugSession object

Return type *DebugSession*

set_config(*ccxml_path*)
Set ccxml file for DebugServer

Parameters **ccxml_path** (*str*) – full path to ccxml file to set

terminate_session(*name*)
Terminates an open session

Parameters **name** (*str*) – name of session to terminate

Raises `Exception` – raises exception if problem terminating session

2.2 DebugSession

class DebugSession(*host=None*, *port=None*)
DebugSession class for controlling session

Parameters

- **host** (*str, optional*) – hostname of DebugSession to connect to (default="localhost")
- **port** (*int*) – port number of DebugSession to connect to

> **Warning:** You should never instantiate this class directly. Instead call the `DebugServer.open_session()` function to create a DebugSession object

connect()
Connect to the device.

disconnect()
Disconnect from the device.

erase()
Erases device's flash memory.

evaluate(*expression*, *file=None*)
Evaluates an expression (after loading optional symbols file)

Parameters

- **expression** (*str*) – C/GEL expression to evaluate
- **file** (*str, optional*) – path to file containing symbols to load before evaluating

Returns result of evaluated expression

Return type int

Raises Exception if expression is invalid.

get_option(*option_id*)
Get the value of a device option

Parameters **option_id** (*str*) – name of device option

Returns value of option

Return type any

Raises Exception if option id is invalid.

halt (*wait=False*)
Halts the device

Parameters **wait** (*boolean*) – wait until device is actually halted before returning

load (*file*, *binary=False*, *address=None*)
Loads image into device's flash.

Parameters

- **file** (*str*) – full path to file to load into flash
- **binary** (*boolean, optional*) – specify to load image as binary (default = False)
- **address** (*int, optional*) – specify to load binary image at specifc address (only to be used when 'binary' is True; default=0x0)

Raises Exception if image fails to load

perform_operation (*opcode*)
Performs flash operation

Parameters **opcode** (*str*) – name of operation to perform (opcode)

Returns returns value of performing operation

Return type any

Raises Exception if opcode is invalid.

read_data (*address*, *page=0*, *num_bytes=1*)
Read memory from device

Parameters

- **address** (*int*) – address to read data from
- **page** (*int, optional*) – page in memory to get address from (default = 0)
- **num_bytes** (*int, optional*) – number of bytes to read

Returns list of bytes(ints) read

Return type list

Raises Exception if address location is invalid.

read_register (*name*)
Read value from register

Parameters **name** (*str*) – register name to read

Returns value of register read

Return type int

Raises Exception if register name is invalid.

reset ()
Resets device.

run (*asynchronous=False*)
Issues the run command to the device

Parameters **asynchronous** (*boolean, optional*) – run and return control immediately (default = False)

set_option (*option_id*, *value*)
Set the value of a device option

Parameters

- **option_id** (*str*) – name of device option
- **value** (*any*) – value to set option to

Raises Exception if option id is invalid.

stop ()
Stops the session thread but does not terminate the session.

verify (*file*, *binary=False*, *address=None*)
Verifies image in device's flash.

Parameters

- **file** (*str*) – full path to file to verify in flash
- **binary** (*boolean, optional*) – specify to verify image as binary (default = False)
- **address** (*int, optional*) – specify to verify binary image at specifc address (only to be used when 'binary' is True; default=0x0)

Raises Exception if image fails verification process

write_data (*data*, *address*, *page=0*)
Write to memory on device

Parameters

- **data** (*list*) – list of bytes (ints) to write to memory
- **address** (*int*) – address to read data from
- **page** (*int, optional*) – page in memory to get address from (default = 0)

Raises Exception if address location is invalid.

write_register (*name*, *value*)
Write value to register on device

Parameters

- **name** (*str*) – register name to write to
- **value** (*int*) – value to write to register

Raises Exception if register name is invalid.

CHAPTER 3

License

DSClient-py is released under the MIT license:

```
Copyright (c) 2019 Cameron Webb

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is␣
↪furnished
to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in␣
↪all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

CHAPTER 4

Disclaimer

```
This project is NOT supported by (nor affiliated with) Texas Instruments Inc.

Code Composer Studio is a trademark of Texas Instruments Inc.
Any and all other trademarks are the property of their respective owners.
```

CHAPTER 5

Release Change Log

5.1 0.2.0-beta

Date: 04.13.2019

- added attach_ccs() function (#1)
- added autocompletion of session names (#2)
- added get_session() function (#3)
- changed `get_list_of_CPUs()` -> `get_list_of_cpus()`

5.2 0.1.0-beta

Date: 04.06.2019

- initial (pre)release

Python Module Index

d

Index