WonderPro™ Fancy Device API - sample document with multiple groups using C header files

Reference Guide

February 11, 2011

Submit technical questions at:

https://support.cdmatech.com

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Contents

List of Figures

List of Tables

Revision History

Revision	Date	Description
A	Jan 2010	Initial release
B	May 2010	Updated to include advanced Doxygen markup commands to support Rev B standards
C	Sep 2010	Updated to support Rev C standards and Doxygen 1.7.0

1 Introduction

1.1 Purpose

This document describes the WonderPro™ Fancy Device (WFD) API. The WFD API provides an interface to some wonderful features.

1.2 Scope

This document is intended for software developers who will be using the WFD API.

This document provides the public interfaces necessary to use the features provided by the WFD API. A high-level overview and information on leveraging the interface functionality are also provided.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font. For example, `#include`.

Code variables appear in angle brackets. For example, `<number>`.

Commands and command variables appear in a different font. For example, **`copy a:*.* b:`**.

Parameter directions are indicated as follows:

- `[in]` indicates an input parameter.
- `[out]` indicates an output parameter.
- `[in,out]` indicates a parameter used for both input and output.

1.4 References

Reference documents, which might include Qualcomm documents and non-Qualcomm standards and resources, are listed below:

- *Application Note: Software Glossary for Customers* (CL93-V3077-1)
- *QCT Doxygen Markup Standards* (80-VP989-1)

1.5 Technical Assistance

For assistance or clarification on information in this guide, submit a case to Qualcomm CDMA Technologies at https://support.cdmatech.com.

If you do not have access to the CDMATech Support Services website, register for access or send email to support.cdmatech@qualcomm.com.

1.6 Acronyms

For definitions of terms and abbreviations, refer to the *Application Note: Software Glossary for Customers* (CL93-V3077-1). The following terms are specific to this document.

Table 1-1 Acronyms

Acronym	Definition
API	application programming interface
AR	access rights
WFD	WonderPro™ Fancy Device

2 Functional Overview

The WFD API is a truly wonderful device.

2.1 Sample API Reference Guide

This sample API Reference Guide document was generated using the SampleHeaderFile1_multigroups_C.h, SampleHeaderFile2_multigroups_C.h, and SampleHeaderFile3_multigroups_C.h, and WFD_mainpage_multigroups_C.dox source files. This document includes example output for the marked up code items identified in the *QCT Doxygen Markup Standards* (80-VP989-1).

Figure 2-1 shows the WFD Software Arhitecture flow.

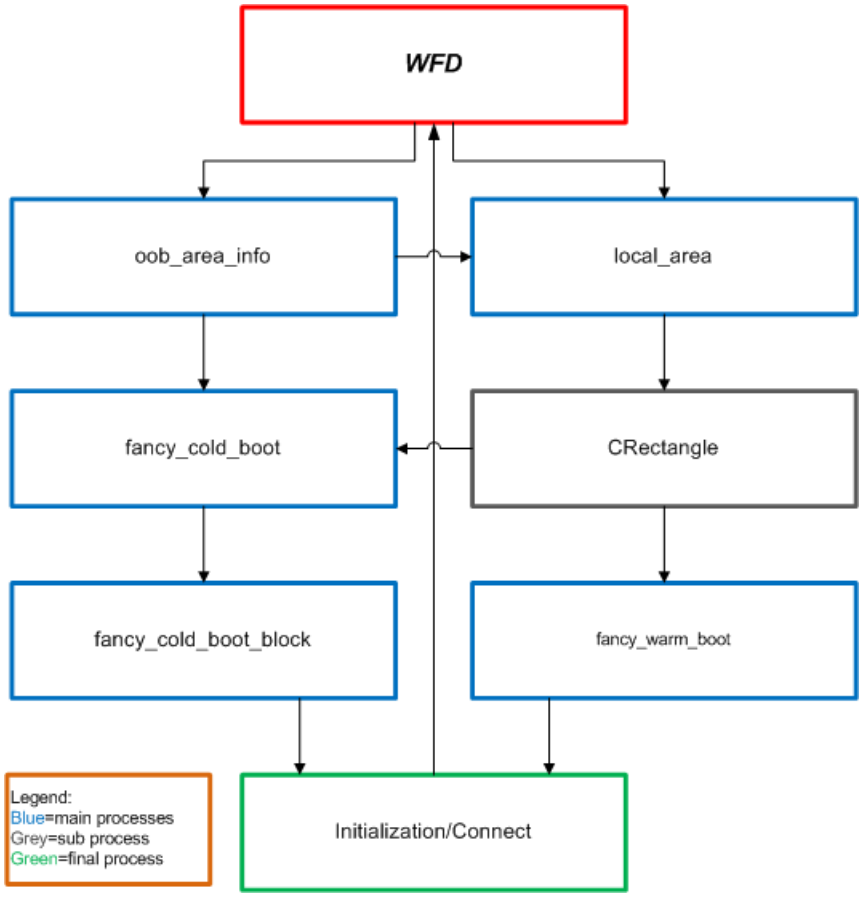

Figure 2-1 WFD Software Architecture

3 Interfaces Index

3.1 Interfaces

Here is a list of all interfaces:

4 Interfaces

4.1 WFD APIs

The WFD APIs provide functionality that is truly wonderful. This is an optional detailed description. The WFD APIs are merely samples created to generate sample groups in the PDF output. They are not intended to be functional. These sentences all stay together in one paragraph in the PDF.

Interfaces

- First Fancy Device API
- Second Fancy Device API
- Third Fancy Device API

4.2 First Fancy Device API

This is a brief description for the First Fancy Device API.

This is the detailed description for the first API group. The

keeps it with the brief comment in the PDF.

Data Structures

- struct oob_area_info

  Keeps track of out-of-bounds block area information and the size of each block.

- struct local_area

WFD Cold Boot Values

- #define FANCY_COLD_BOOT_START_BLOCK_VALUE 0
- #define FANCY_MAX_COLD_BOOT_END_BLOCK_VALUE 1
- #define FANCY_COLD_BOOT_BLOCK_VALUE_INVALID ((cold_blk_invalid_type) 0xFF)
- #define FANCY_COLD_BOOT_BLOCK_VALUE_UNASSIGNED ((cold_blk_unassigned_type) 0xFE)

4.2.1 Define Documentation

4.2.1.1 #define FANCY_COLD_BOOT_START_BLOCK_VALUE 0

Initial value for the start block.

4.2.1.2 #define FANCY_MAX_COLD_BOOT_END_BLOCK_VALUE 1

Maximum value for the end block.

4.2.1.3 #define FANCY_COLD_BOOT_BLOCK_VALUE_INVALID ((cold_blk_invalid_type) 0xFF)

Value for the cold boot block is invalid.

4.2.1.4 #define FANCY_COLD_BOOT_BLOCK_VALUE_UNASSIGNED ((cold_blk_unassigned_type) 0xFE)

Value for the cold boot block is unassigned.

4.3 Second Fancy Device API

This is a brief description for the Second Fancy Device API.

Data Structures

- class CRectangle

  Takes the height and length of a rectangle and returns the area value.

- struct fancy_warm_boot_block

  Holds the new and previous warm boot block numbers.

Namespaces

- namespace AR
- namespace fancy_audio_1

Defines

- #define FANCY_WARM_BOOT_START_BLOCK_VALUE 0
- #define FANCY_MAX_WARM_BOOT_END_BLOCK_VALUE 1

- #define FANCY_WARM_BOOT_BLOCK_VALUE_INVALID ((warm_blk_invalid_type) 0xFF)
- #define FANCY_WARM_BOOT_BLOCK_VALUE_UNASSIGNED ((warm_blk_unassigned_type) 0xFE)

Enumerations

- enum download_type { download_OOB, download_NOOB, download_ONENOOB, download_MULTI }
- enum download_tech { download_SLC, download_MLC }
- enum download_ID { download_x8 = 8, download_x16 = 16 }
- enum {

  FANCY_READ_MAIN = 0, **FANCY_READ_SPARE**, **FANCY_READ_MAIN_SPARE**, **FANCY_READ_RAW**,

  FANCY_READ_BYTES }

Functions

- void **CRectangle::set_values** (int, int)

4.3.1 Detailed Description

This is the detailed description for the second API group. It appears in the Detailed Description section.

4.3.2 Define Documentation

4.3.2.1 #define FANCY_WARM_BOOT_START_BLOCK_VALUE 0

Initial value for the start block.

4.3.2.2 #define FANCY_MAX_WARM_BOOT_END_BLOCK_VALUE 1

Maximum value for the end block.

4.3.2.3 #define FANCY_WARM_BOOT_BLOCK_VALUE_INVALID ((warm_blk_invalid_type) 0xFF)

Warm boot block value is invalid.

4.3.2.4 #define FANCY_WARM_BOOT_BLOCK_VALUE_UNASSIGNED ((warm_blk_-unassigned_type) 0xFE)

Warm boot block value is unassigned.

4.3.3 Enumeration Type Documentation

4.3.3.1 enum download_type

Identifies the device types for downloading software to one or more devices.

Enumerator:

download_OOB Out-Of-Bounds device.
download_NOOB More than one Not-Out-Of-Bounds device.
Note: This is an example of a note for an enum member (note1 command). This can also be used in param descriptions.
download_ONENOOB OneNOOB device.
download_MULTI Multiple devices.

4.3.3.2 enum download_tech

Identifies the bits per cell for the device.

Enumerator:

download_SLC Single-Level Cell device.
download_MLC Multi-Level Cell device.

4.3.3.3 enum download_ID

Identifies the download ID.

Enumerator:

download_x8 8-bit interface download ID.
download_x16 16-bit interface download ID.

4.4 Third Fancy Device API

This is a brief description for the third Fancy Device API.

Data Structures

- struct fancy_cold_boot_block

  Holds the new and previous cold boot block numbers.

- union _AfancyAudioEvent

  Accesses the header values. This union includes two data structures.

- struct fancy_qos_params_type

  Stores Fancy Quality of Service parameters.

Namespaces

- namespace fancy_audio_2

Defines

- #define FANCY_DEVICE_DONE 0
- #define FANCY_DEVICE_FAIL (-1)
- #define FANCY_DEVICE_NOT_SUPPORTED (-2)
- #define FANCY_DEVICE_CP_READ_FAIL (-3)
- #define FANCY_INTERFACE_VERSION DALINTERFACE_VERSION(1,0)

Enumerations

- enum fancy_header_comp_e_type { FANCY_HEADER_COMP_OFF = 0, FANCY_HEADER_COMP_ON = 1, FANCY_HEADER_COMP_MAX = 0xFF }
- enum fancy_dev_val_e_type { FANCY_DEVICE_VAL_OFF = 0, FANCY_DEVICE_VAL_ON = 1 }
- enum FancySysTimeType1 { **IAO_SYSTIME_UMTS** = 0x00, IAO_SYSTIME_GSM = 0x01, IAO_SYSTIME_TOT }
- enum WfdWSEvtType { WFD_WS_EVENT_TOT }

Functions

- static int CreateInstance (FANCYCLID clsid, IEnv *pEnvironment, IPrivSet *pPrivSet, void **ppNewObj)
- virtual int NDQueryInterface (FANCYID idReq, IQI **ppIface)
- FancyResult NameThread ()
- FancyInterfere (FANCYCLSID clsid, pEnv *pEnvironment, int &result)
- int printf (const char *fmt,...)

4.4.1 Detailed Description

This is the detailed description for the third API group. It appears in the Detailed Description section.

4.4.2 Define Documentation

4.4.2.1 #define FANCY_DEVICE_DONE 0

Operation passed.

4.4.2.2 #define FANCY_DEVICE_FAIL (-1)

Operation failed.

4.4.2.3 #define FANCY_DEVICE_NOT_SUPPORTED (-2)

Device is not supported.

4.4.2.4 #define FANCY_DEVICE_CP_READ_FAIL (-3)

Copy page read failure.

4.4.2.5 #define FANCY_INTERFACE_VERSION DALINTERFACE_VERSION(1,0)

Defines the interface version. The interface version can be 0 (the default) to indicate that this is the first instance, or it can be 1 to indicate that this is the latest version.

4.4.3 Enumeration Type Documentation

4.4.3.1 enum fancy_header_comp_e_type

Fancy header compression types.

Enumerator:

FANCY_HEADER_COMP_OFF Compression is off (default).
FANCY_HEADER_COMP_ON Compression is on when selected.
FANCY_HEADER_COMP_MAX Forces the maximum compression to 0xff so the enumeration is defined as a byte.

4.4.3.2 enum fancy_dev_val_e_type

Fancy device validation modes.

Enumerator:

FANCY_DEVICE_VAL_OFF Device validation mode is off (default).
FANCY_DEVICE_VAL_ON Device validation mode is on when selected.

4.4.3.3 enum FancySysTimeType1

Enumerator:

IAO_SYSTIME_GSM GSM system time.
IAO_SYSTIME_TOT Total number of system time types.

4.4.3.4 enum WfdWSEvtType

Supported WFD warm start events.

Enumerator:

WFD_WS_EVENT_TOT Total number of warm start events

4.4.4 Function Documentation

4.4.4.1 static int CreateInstance (FANCYCLID *clsid,* IEnv * *pEnvironment,* IPrivSet * *pPrivSet,* void ** *ppNewObj*)

Provides a public entry point for the module. A new instance of FancyInterfere is created for FancyModule, and the default interface is returned.

Note

Note text 1.
Note text 2.
Note text 3.

Parameters

`[in]` ***clsid*** Class ID of the module.
`[in]` ***pEnvironment*** Interface to the Fancy Services environment.
`[out]` ***pPrivSet*** Privilege set of the caller.
`[in,out]` ***ppNewObj*** Set to the interface pointer of the new instance.

Returns

SUCCESS – Instance was created successfully.
ENOMEMORY – Indicates a memory allocation failure.

Dependencies

None.

Side effects

An invalid class ID may produce erroneous results.

See also

NDQueryInterface

4.4.4.2 virtual int NDQueryInterface (FANCYID *idReq,* IQI ** *ppIface*) `[virtual]`

Handles interface pointers for non-based classes.

This method must be implemented if the FancyInterfere module supports any interfaces not handled by its base classes.

Parameters

`[in]` ***idReq*** Unique ID of the requested interface.
`[out]` ***ppIface*** Interface pointer of the requested interface.

Returns

SUCCESS – Interface is returned successfully.
ECLASSNOTSUPPORT – Interface is not supported.

4.4.4.3 FancyResult NameThread ( )

Handles thread name intialization.

The deriving class can override this method to provide a descriptive name for the optional output thread. This helps distinguish it in debugging applications. The default implementation provides a generic name.

Note: This is an example of a single hanging indent note (note1hang command). It is primarily intended for use in fancy tools during software download.

Returns

SUCCESS – Initialization was successful.
FAILURE – Initialization failed.

4.4.4.4 FancyInterfere (FANCYCLSID *clsid,* pEnv * *pEnvironment,* int & *result*)

Provides a class constructor.

This is the component constructor, meant to be called by the class factory of the component.

Parameters

`[in]` ***clsid*** Class ID of the module.
`[in]` ***pEnvironment*** Interface to the Fancy Services environment.
`[out]` ***result*** Result of the function.

Returns

The following results may be returned:
- Valid class object
- Invalid class object

4.4.4.5 int printf (const char * *fmt,* ...)

Standard print function. The extern keyword is used to inform the compiler about variables declared outside of the current file. Variables described by extern statements do not have any space allocated for them because they have been previously defined elsewhere.

4.5 Fancy Device Common Data

This is a brief description for the Fancy Device Common Data group. This is the detailed description for the Fancy Device Common Data group. It appears in the same paragraph as the brief comment.

- typedef struct fancy_handle **fancy_handle**
- typedef fancy_handle * **fancy_handle_t**
- #define FANCY_DEFAULT_DATA_PROFILE_VERSION 3

4.5.1 Define Documentation

4.5.1.1 #define FANCY_DEFAULT_DATA_PROFILE_VERSION 3

Default profile version. The value of the Fancy Device target's profile version depends on the following conditional settings:

- If FANCY_DATA_TARGET1 is TRUE, FANCY_DATA_PROFILE_VERSION = 7.
- If FANCY_DATA_TARGET2 is TRUE, FANCY_DATA_PROFILE_VERSION = 9.
- If FANCY_DATA_TARGET3 is TRUE, FANCY_DATA_PROFILE_VERSION = 12.
- If none of the above conditions are TRUE, FANCY_DEFAULT_DATA_PROFILE_VERSION = 3.

5 Namespace Documentation

5.1 AR Namespace Reference

Access rights namespace.

Enumerations

- enum **RightsChangeReason** {

  RightsChangeReason_Added, **RightsChangeReason_Deleted**, **RightsChangeReason_Modified**, **RightsChangeReason_Unknown**,

  _AR_PLACEHOLDER_RightsChangeReason = 0x7fffffff }

Variables

- const ::FANCYID FANCYID_IRightsChange = 0x1074feb

5.1.1 Variable Documentation

5.1.1.1 const ::FANCYID AR::FANCYID_IRightsChange = 0x1074feb

Changes the access rights.

5.2 fancy_audio_1 Namespace Reference

Provides SamplerState and ChannelMode1 enumeration types.

Enumerations

- enum **ResamplerState** { **RS_OK**, **RS_NEED_INPUT_BUF**, **RS_NEED_OUTPUT_BUF** }
- enum **ChannelMode1** { **MONO** = 1, **STEREO** = 2 }

Functions

- static int CreateInstance (FANCYCLSID clsid, IEnv *pEnvironment, IPrivSet *pPrivSet, void **ppNewObj)

Variables

- GenericResamplerLib * m_pResampler
- bool m_Enabled
- ICritSect * m_pRxBufferLock
- std::queue< FancyCommand * > m_fancyRxBufferQueue
- std::queue< FancyCommand * > **m_fancyTxBufferQueue**
- DECLARE_IQI

5.2.1 Detailed Description

This is the first fancy audio namespace.

5.2.2 Function Documentation

5.2.2.1 static int fancy_audio_1::CreateInstance (FANCYCLSID *clsid*, IEnv * *pEnvironment*, IPrivSet * *pPrivSet*, void ** *ppNewObj*)

Provides a public entry point for the module.

A new instance of FancyPlayback is created, and the default interface (FancyMediaModule) is returned.

Parameters

`[in]` ***clsid*** FANCYCLSID of the module.
`[in]` ***pEnvironment*** Interface to the Fancy Services environment.
`[out]` ***pPrivSet*** Privilege set of the caller.
`[in,out]` ***ppNewObj*** Interface pointer of the new instance.

Returns

SUCCESS – Instance was created successfully.
ENOMEMORY – Memory allocation failure.

Comments

A memory allocation failure requires a reboot.

Errors

If the new environment value is not an integer, the compiler generates an error.
If the new environment object is called outside the Fancy Services environment without the pointer, a constraint error occurs.

5.2.3 Variable Documentation

5.2.3.1 GenericResamplerLib* fancy_audio_1::m_pResampler

Generic Resampler library object.

5.2.3.2 bool fancy_audio_1::m_Enabled

Enables the master flag during FancyPlayback.

5.2.3.3 ICritSect* fancy_audio_1::m_pRxBufferLock

Critical section for the Rx buffer queue.

5.2.3.4 std::queue<FancyCommand*> fancy_audio_1::m_fancyRxBufferQueue

Rx and Tx queues for Fancy Device buffers received from the application.

5.2.3.5 fancy_audio_1::DECLARE_IQI

Pulls in IQueryInterface method definitions.

5.3 fancy_audio_2 Namespace Reference

Provides ResamplerState and ChannelMode2 enumeration types.

Enumerations

- enum SamplerState { S_OK, S_NEED_INPUT_BUF, S_NEED_OUTPUT_BUF }
- enum ChannelMode2 { MONO = 1, STEREO = 2 }

Functions

- virtual int CDECL NDQueryInterface (FANCYCLSIDRQ idReq, IQI **ppIface)
- FancyResult NameThread ()
- FancyInterfere (FANCYCLSID clsid, IEnv *pEnvironment, int &result)

Variables

- GenericSamplerLib * m_pResampler
- uint32 m_uiInpBufferSize
- boolean m_fSampTypeSet
- bool m_Enabled
- bool m_EnabledRead
- ISignalQ * m_pEventSignalQ
- std::queue< FancyCommand * > m_fancyRxBufferQueue
- std::queue< FancyCommand * > m_fancyTxBufferQueue

- DECLARE_IQI

5.3.1 Detailed Description

This is the detailed fancy audio namespace.

5.3.2 Enumeration Type Documentation

5.3.2.1 enum fancy_audio_2::SamplerState

Sample comment for enum inside a namespace.

Enumerator:

S_OK Resampler state is in OK mode.
S_NEED_INPUT_BUF Resampler state needs an input buffer.
S_NEED_OUTPUT_BUF Resampler state needs an output buffer.

5.3.2.2 enum fancy_audio_2::ChannelMode2

Another sample comment for enum inside a namespace.

Enumerator:

MONO Channel Mode 2 is in mono mode.
STEREO Channel Mode 2 is in stereo mode.

5.3.3 Function Documentation

5.3.3.1 virtual int CDECL fancy_audio_2::NDQueryInterface (FANCYCLSIDRQ *idReq,* IQI ** *ppIface*) [virtual]

Provides an interface to support the FancyInterface module.

This method must be implemented if the FancyInterfere module supports any interfaces not handled by its base classes.

Parameters

`[in]` ***idReq*** Unique ID of the requested interface.
`[out]` ***ppIface*** Interface pointer of the requested interface.

Returns

SUCCESS – Interface is returned successfully.
ECLASSNOTSUPPORT – Interface is not supported.

5.3.3.2 FancyResult fancy_audio_2::NameThread ( )

Overrides a generic name.

The deriving class may override this method to provide a descriptive name for the optional output thread to help distinguish it in debugging applications. The default implementation provides a generic name.

Returns

SUCCESS – Initialization was successful.
FAILURE – Initialization failed.

5.3.3.3 fancy_audio_2::FancyInterfere (FANCYCLSID *clsid,* IEnv * *pEnvironment,* int & *result*)

Provides a class constructor.

This is the component constructor, meant to be called by the class factory of the component.

Parameters

`[in]` ***clsid*** Class ID of the module.
`[in]` ***pEnvironment*** Interface to the component services environment.
`[out]` ***result*** Result of the function.

Returns

New class object.

5.3.4 Variable Documentation

5.3.4.1 GenericSamplerLib* fancy_audio_2::m_pResampler

Generic Sampler library object.

5.3.4.2 uint32 fancy_audio_2::m_uiInpBufferSize

Size of the input buffer delivered to the sampler.

5.3.4.3 boolean fancy_audio_2::m_fSampTypeSet

SamplerType Set flag.

5.3.4.4 bool fancy_audio_2::m_Enabled

Enables the flag master for FancyInterfere.

5.3.4.5 bool fancy_audio_2::m_EnabledRead

Enables FancyInterfere on the Read channel.

5.3.4.6 ISignalQ* fancy_audio_2::m_pEventSignalQ

Signal queue for blocking events.

5.3.4.7 std::queue<FancyCommand*> fancy_audio_2::m_fancyRxBufferQueue

Rx queues for Fancy Device buffers received from the application.

5.3.4.8 std::queue<FancyCommand*> fancy_audio_2::m_fancyTxBufferQueue

Tx queues for Fancy Device buffers received from the application.

5.3.4.9 fancy_audio_2::DECLARE_IQI

Pulls in IQueryInterface method definitions.

6 Data Structure Documentation

6.1 _AfancyAudioEvent Union Reference

Accesses the header values. This union includes two data structures.

Data Fields

- struct AfancyAudioAnyEvent **any**
- struct AfancyAudioAnyEvent **header**

6.2 CRectangle Class Reference

Takes the height and length of a rectangle and returns the area value.

Public Member Functions

- void **set_values** (int, int)
- int **area** ()

Data Fields

- int **x**
- int **y**

6.2.1 Detailed Description

This class (CRectangle) contains four members: two input parameter members of type integer (x, y, where x=height and y=length) and two member functions (set_values, area) with public access.

This class calls the set_values member function to set the input values (x, y) to integers. This function is void, which indicates that is does not return a result; it just holds the results temporarily. It then calls the area member function, which takes the values in the set_values member function (x, y), multiplies them (x∗y), and returns the result (area of the rectangle).

Parameters

`[in]` ***x*** Data member x of type integer with private access. This member is private by default.

[in] ***y*** Data member y of type integer with private access. This member is private by default.

Returns

Area of the rectangle as an integer value.

Dependencies

None.

msc_callflow_sample.txt "Sample Callflow"

6.3 fancy_cold_boot_block Struct Reference

Holds the new and previous cold boot block numbers.

Data Fields

- int cold_boot_new_block
- int cold_boot_old_block

6.3.1 Detailed Description

This is an optional detailed description. This structure is used in Fancy Device tools during software initialization.

6.3.2 Field Documentation

6.3.2.1 int fancy_cold_boot_block::cold_boot_new_block

New cold boot block number.

6.3.2.2 int fancy_cold_boot_block::cold_boot_old_block

Old cold boot block number.

6.4 fancy_qos_params_type Struct Reference

Stores Fancy Quality of Service parameters.

Data Fields

- boolean valid_flg
- uint32 precedence
- uint32 mean

6.4.1 Detailed Description

This is an optional detailed description. This structure includes three members.

6.4.2 Field Documentation

6.4.2.1 boolean fancy_qos_params_type::valid_flg

Indicates whether the parameters are set and valid. This is a test detailed sentence.

6.4.2.2 uint32 fancy_qos_params_type::precedence

Precedence class. This is a test detailed sentence.

6.4.2.3 uint32 fancy_qos_params_type::mean

Mean throughput class.

6.5 fancy_warm_boot_block Struct Reference

Holds the new and previous warm boot block numbers.

Data Fields

- int warm_boot_new_block
- int warm_boot_old_block

6.5.1 Detailed Description

This is an optional detailed description. This structure is used in Fancy Device tools during software download.

6.5.2 Field Documentation

6.5.2.1 int fancy_warm_boot_block::warm_boot_new_block

New warm boot block number.

6.5.2.2 int fancy_warm_boot_block::warm_boot_old_block

Old warm boot block number.

6.6 local_area Struct Reference

Data Fields

- uint32 area_count
- uint32 area_size_in_bytes

6.6.1 Field Documentation

6.6.1.1 uint32 local_area::area_count

Total number of blocks in the local area.

6.6.1.2 uint32 local_area::area_size_in_bytes

Size of each block in the local area.

6.7 oob_area_info Struct Reference

Keeps track of out-of-bounds block area information and the size of each block.

Data Fields

- uint8 block_count
- uint8 block_size_in_bytes

6.7.1 Field Documentation

6.7.1.1 uint8 oob_area_info::block_count

Number of blocks in the out-of-bounds area.

6.7.1.2 uint8 oob_area_info::block_size_in_bytes

Size of each block in the out-of-bounds area.