cadence

Cadence I3C Master Controller Interface Firmware Driver User Guide

Product Version 1.0.3

January 2024

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Cadence I3C Master Controller Interface Firmware Driver User Guide

Cadence Design Systems

Hardware Identifier:3b7280f2c9ba6578cc418ade334264cc

Table of Contents

Chapter 1. Acronyms

API	Application Programming Interface.
CCC	Common Command Code
Core Driver	Cadence Firmware component that provides IP programming abstraction.
CPS	Cadence Platform Services. A set of basic platform specific access functions acting as hardware abstraction layer for the Core Driver to operate.
FM	Fast Mode
FM+	Fast Mode Plus
ISR	Interrupt Service Routine
P2P	Peer-to-Peer

Chapter 2. Document Purpose

This document is intended to serve as an overview and a reference for customers looking to use the Cadence I3C Firmware Core Driver for the Cadence I3C Master Controller Interface IP Core.

Chapter 3. Description

The FW provided by Cadence is :

1. Production Code: Code built to production standards. i.e. it is fully verified with positive and negative testing.

2. Reference Code: This is example code for the platform specific portions of the customers code base. Reference code is only provided as an example to help the customer with his environment. Ideally code snippets from this will be looked at, and the customer will create his own versions of the reference code that is specific to his environment.

3.1. Naming conventions

The Cadence I3C Master Controller Interface Core Driver API will use the i3c_ prefix for public top-level names of functions, constants and data objects definitions.

Chapter 4. Safety Manual

4.1. System Requirements

The Core Driver is written to be operating system independent and so does not use any OS features like:

Memory allocation	All memory required by driver shall be allocated or statically instantiated by customer. This means that the driver's PrivateData structure must be declared outside the Core Driver. There must be one instance of PrivateData for each instance of IP in a system.
Threads	The Core Driver was written as one thread component. It does not use any synchronization mechanism like mutexes or semaphores. It is the customer's responsibility to support synchronization for interrupts and application calls to the driver between threads in multi-threaded or multi-cores systems including interrupts and application calls to the Core Driver.

4.2. ASF Interrupts

To allow detection of various issues which may happen during system's runtime, like parity errors or checksum errors, it is strongly recommended to handle all ASF interrupts by calling the API function *ASF_Isr()* and to handle appropriate events in the customer's IRQ callback routine. The customer should then decide what should be done when ASF raises some unexpected event. All interrupts should be enabled using *ASF_EnableEvent()*, and *ASF_Isr()* should then be called when the interrupt is received by the customer interrupt handling code. If there is a risk that interrupts might be lost, the customer code should instead regularly poll the *ASF_Isr()* function. A list of interrupts generated by ASF can be found in the ASF 'User Guide'.

4.3. I3C Master Interrupts

Controller's interrupts should be enabled as well. These events inform about important protocol events. Ignoring such events may lead to lost data. All interrupts should be enabled by API call: *I3C_Start()* and *I3C_Isr()* should then be called when the interrupt is received by the customer interrupt handling code. If there is a risk that interrupts might be lost, the customer code should instead regularly poll the *I3C_Isr()* function. A list of interrupts generated by the controller can be found in the controller’s 'User Guide'.

4.4. Commands

- When commands are executed is not possible to modify Bus Mode and configure devices. To perform such operations please check status of all commands by checking bit*MST_STATUS0.IDLE*.
- If controller cannot execute commands stored in FIFO queue please make sure, that submitted commands were executed as well.
- Threshould should be configured at higher value than number of commands executed.

Chapter 5. Core Driver Usage Model

5.1. Description

In order to support multiple instances and provide the minimum name space pollution, the Core Driver is implemented as a structure that contains a function pointer table.

A public header is provided to define the layer API object structure and the data structures used. The file is contained in the include directory of the distribution as *i3c.h*.

The API requires a private data pointer that provides the Driver with instance information. It is critical that the upper software layer abides by the programming model.

Every API call apart from *probe* requires the private data address as the first parameter, to identify the Driver instance and provide access to the internal state of that instance.

There is no resource protection mechanism provided in the I3C Core Driver. It is expected that the higher level software stack will be responsible for protecting calls to the Core Driver where appropriate. In critical areas such as the calling of the *isr* function, usage is expected to be single threaded.

5.2. Probing the Hardware

The client code must obtain the object pointer to the Core Driver. It can be retrieved by issuing the *I3C_GetInstance()* function. This provides access to all of the API functions, but initially only *probe* may be called.

Before initializing the Core Driver, the client must first invoke the *probe* function. The *probe* function returns the memory requirements for the Driver's internal data, depending on the configuration specified. The configuration information is passed to *probe* in an *I3C_Config* struct, although at this stage only those fields which affect the requirements need to be set, as indicated in the struct description. The client may repeat *probe* calls to customise configuration depending on available system resources.

The client code must then allocate the memory to the sizes set by the probe function in return privDataSize variable.

```
I3C_SysReq req = { 0 };
drv->probe(cfg, &req);
*pd = malloc(req.memReq);
```

5.3. Initialization

After this stage, the client must reserve the necessary memory area for the Driver's private data and hardware configuration (if needed). The *I3C_Config* struct and *I3C_Callbacks* structures should be filled out with the values for all initial configuration fields. The configuration is not fixed and can be altered in later time. If any callback in *I3C_Callbacks* structure is not used, client should assign *NULL* as function pointer.

The Driver can now be initialized with an *init* call, which will:

- Initialize the Driver instance.
- Configure the hardware as specified in the *config* parameter field.
- Assign callbacks set in *callbacks* parameter field.

e.g. *init* example, following on from *probe* example above:

```
if (drv->init(*pd, cfg, &callbacks)) {
    log("Error. Failed to initialize Core Driver.\n");
    return EINVAL;
}
```

When ready, *start* should be called to enable interrupt handling and enabled callbacks which in result will start the Driver. Callbacks can be enabled by registering user's event handlers and setting appropriate events to be reported.

5.4. Event/Interrupt Handlers

For each interrupt user can assign a handler. Handler will be called upon interrupt occurrence. Handler prototype is defined as follows:

void (*I3C_InterruptEvent)(void* pd);
In addition there are several status events which can be assigned with handlers as well. Status events are as follows:

- receive data fifo full,
- In-Band Interrupt fifo full,
- Dynamic Address Assign completed,
- Double Data Rate transfer error,
- Immediate Commad completed.

Status event handler prototype is:

void (*I3C_StatusEvent)(void* pd, I3C_CommandStatus sts);

5.5. Interrupt Handling

The Core Driver notifies upper layer software of new events and errors by means of the *isr* API function. The *isr* function may be invoked by upper layer software as part of the interrupt processing being handled by the upper layer software. Alternatively, if interrupt-driven operation is not desired, *isr* can be called by frequent polling. In the later case, the I3C interrupt should not lead to a read of the interrupt status register by the client software, as this must be interpreted and cleared by the Core Driver's *isr* function.

The invocation of the *isr* function of the Core Driver will result in a callback to the function registered as part of the *init* process if there is an enabled event that is encountered by the Core Driver. It is the responsibility of the upper layer software to deal with any errors that are reported by the Core Driver.

Actions performed by the upper layer during a callback function should be kept to a minimum, as the function must return before further *isr* interrupt checking can proceed.

5.6. Stopping and destroying the Driver

The *stop* function can be called to end the Driver's active operation.

A *destroy* function is provided. *destroy* function automatically calls *stop* function which disables I3C Master Controller component and therefore stops the Driver. User may use either of those functions after finishing work with Driver.

Chapter 6. Dynamic Data Structures

6.1. Introduction

This section defines the data structures used by the driver to provide hardware information, modification and dynamic operation of the driver.

These data structures are defined in the header file of the core driver and utilized by the API.

6.2. I3C_InterruptConfig_s

Type: struct

Description

This structure holds configuration of interrupts.

Fields

`cmdResponseOverflow`	If cmdResponseOverflow interrupt should be enabled.
`cmdResponseUnderflow`	If cmdResponseUnderflow interrupt should be enabled.
`cmdResponseThreshold`	If cmdResponseThreshold interrupt should be enabled.
`cmdDescriptorOverflow`	If cmdDescriptorOverflow interrupt should be enabled.
`cmdDescriptorEmpty`	If cmdDescriptorEmpty interrupt should be enabled.
`cmdDescriptorThreshold`	If cmdDescriptorThreshold interrupt should be enabled.
`rxDataFifoUnderflow`	If rxDataFifoUnderflow interrupt should be enabled.
`rxFifoThreshold`	If rxFifoThreshold interrupt should be enabled.
`ibiResponseOverflow`	If ibiResponseOverflow interrupt should be enabled.
`ibiResponseUnderflow`	If ibiResponseUnderflow interrupt should be enabled.
`ibiResponseThreshold`	If ibiResponseThreshold interrupt should be enabled.
`ibiDataUnderflow`	If ibiDataUnderflow interrupt should be enabled.
`ibiDataThreshold`	If ibiDataThreshold interrupt should be enabled.
`txDataFifoOverflow`	If txDataFifoOverflow interrupt should be enabled.
`txFifoThreshold`	If txFifoThreshold interrupt should be enabled.
`immComplete`	If immComplete interrupt should be enabled.
`mastershipDone`	If mastershipDone interrupt should be enabled.

`halted`	If halted interrupt should be enabled.

6.3. I3C_SlaveInterruptConfig_s

Type: struct

Description

This structure holds configuration of slave-mode interrupts.

Fields

`sdrWrComplete`	If sdrWrComplete interrupt should be enabled.
`sdrRdComplete`	If sdrWrComplete interrupt should be enabled.
`ddrWrComplete`	If ddrWrComplete interrupt should be enabled.
`ddrRdComplete`	If ddrRdComplete interrupt should be enabled.
`sdrTxDataFifoOverflow`	If sdrTxDataFifoOverflow interrupt should be enabled.
`sdrRxDataFifoUnderflow`	If sdrRxDataFifoUnderflow interrupt should be enabled.
`ddrTxDataFifoOverflow`	If ddrTxDataFifoOverflow interrupt should be enabled.
`ddrRxDataFifoUnderflow`	If ddrRxDataFifoUnderflow interrupt should be enabled.
`sdrTxFifoThreshold`	If sdrTxFifoThreshold interrupt should be enabled.
`sdrRxFifoThreshold`	If sdrRxFifoThreshold interrupt should be enabled.
`ddrTxFifoThreshold`	If ddrTxFifoThreshold interrupt should be enabled.
`ddrRxFifoThreshold`	If ddrRxFifoThreshold interrupt should be enabled.
`masterReadAbort`	If masterReadAbort interrupt should be enabled.
`ddrFail`	If ddrFail interrupt should be enabled.
`sdrFail`	If sdrFail interrupt should be enabled.
`dynamicAddrUpdated`	If dynamicAddrUpdated interrupt should be enabled.
`ibiDone`	If ibiDone interrupt should be enabled.
`ibiNack`	If ibiNack interrupt should be enabled.
`mrDone`	If mrDone interrupt should be enabled.
`hotJoinDone`	If hotJoinDone interrupt should be enabled.
`hotJoinNack`	If hotJoinNack interrupt should be enabled.

`eventUpdate`	If eventUpdate interrupt should be enabled.
`protocolError`	If protocolError interrupt should be enabled.
`testMode`	If testMode interrupt should be enabled.
`defslvs`	If defslvs interrupt should be enabled.
`ibid_thr`	If ibid_thr interrupt should be enabled.
`mwl_up`	If mwl_up interrupt should be enabled.
`mrl_up`	If mrl_up interrupt should be enabled.
`resetDaa`	If resetDaa interrupt should be enabled.
`buscon_up`	If buscon_up interrupt should be enabled.
`flush_done`	If flush interrupt should be enabled.

6.4. I3C_ThresholdConfig_s

Type: struct

Description

Configuration of threshold level.

Fields

`txFifoThreshold`	Threshold level for data TX FIFO.
`rxFifoThreshold`	Threshold level for data RX FIFO.
`ibirFifoThreshold`	Threshold level for IBIR FIFO.
`cmdrFifoThreshold`	Threshold level for CMDR FIFO.

6.5. I3C_Config_s

Type: struct

Description

Configuration parameters passed to probe & init functions.

Fields

`regsBase`	Base address of the register space.
`sysClk`	System clock frequency [Hz].

`i2cFreq`	SCL clock frequency of the slowest I2C Slave [Hz].
`sdrFreq`	SDR Speed frequency [Hz].
`sclLowPeriod`	Required SCL low period [ns].
`maxSclFreq`	Maximal frequency supported by slave [Hz].
`devs`	Array of I3C/I2C Device configuration descriptors.
`numDevs`	Number of the I3C/I2C Devices in the configuration structure.
	Minimum number of required configurations is 2 and the maximum is 64.
`ibiSirCfgs`	Pointer to array of IBI configuration structures.
`numSirCfgs`	Number of I3C IBI configured devices.
`cmdFifoSize`	Maximum number of items that can be put in the Command FIFO.
`cmdFifoThreshold`	Threshold level for command FIFO.
`txFifoSize`	Maximum number of bytes that can be put in the TX FIFO.
`rxFifoSize`	Maximum number of bytes that can be put in the RX FIFO.
`thresholdConfig`	Threshold configuration.
	Levels for threshold interrupts.
`interruptConfig`	Configuration for enabling and disabling interrupts.
`slaveInterruptConfig`	Slave interrupt configuration.

6.6. I3C_SysReq_s

Type: struct

Description

System requirements returned by probe.

Fields

`memReq` Size of memory required for driver's private data.

6.7. I3C_Callbacks_s

Type: struct

Description

Configuration parameters passed to probe & init functions.

Fields

`cmdResponseOverflow`	This function will be called while attempting to write a response to full command response FIFO.
`cmdResponseUnderflow`	This function will be called while attempting to read response from empty response FIFO.
`cmdResponseThreshold`	This function will be called when command response FIFO threshold is reached.
`cmdDescriptorOverflow`	This function will be called while attempting to write a command to full command descriptor FIFO.
`cmdDescriptorEmpty`	This function will be called while attempting to read data from empty command descriptor FIFO.
`cmdDescriptorThreshold`	This function will be called when command descriptor FIFO threshold is reached.
`rxDataFifoUnderflow`	This function will be called when driver attempts to read from empty RX FIFO.
`rxFifoThreshold`	This function will be called when receive data FIFO threshold is reached.
`ibiResponseOverflow`	This function will be called while attempting to write a command to full ibi response FIFO.
`ibiResponseUnderflow`	This function will be called while attempting to read data from empty ibi response FIFO.
`ibiResponseThreshold`	This function will be called when ibi response FIFO threshold is reached.
`ibiDataUnderflow`	This function will be called while attempting to read data from empty ibi data FIFO.
`ibiDataThreshold`	This function will be called when ibi data FIFO threshold is reached.
`txFifoThreshold`	This function will be called when transmit data FIFO threshold is reached.
`txDataFifoOverflow`	This function will be called when driver attempts to write data to full TX FIFO.
`immComplete`	This function will be called when Immediate Command(s) processing is completed.
`mastershipDone`	This function will be called when Mastership Request interrupt occurs.
`halted`	This function will be called when Halted interrupt occurs.
`commandBufferEmpty`	This function will be called when all commands form command buffer are sent.
`commandComplete`	This function will be called when command is processed without errors.
`daaComplete`	This function will be called when DAA procedure is finished.
`mastershipRequest`	This function will be called when mastership request occurs.
`hotjoin`	This function will be called when Hot-Join occurs.

`inbandInterrupt`	This function will be called when Hot-Join occurs.
`slaveSdrWrComplete`	This function will be called when SDR Write is completed and master operates in slave mode.
`slaveSdrRdComplete`	This function will be called when SDR Read is completed and master operates in slave mode.
`slaveDdrWrComplete`	This function will be called when DDR Write is completed and master operates in slave mode.
`slaveDdrRdComplete`	This function will be called when DDR Read is completed and master operates in slave mode.
`slaveIBIDone`	This function will be called when IBI request done is detected and master operates in slave mode.
`slaveSdrError`	This function will be called when SDR error is detected and master operates in slave mode.
`testMode`	This function will be called when TestMode command arrives.
`slaveMwlChange`	This function will be called when SETMWL CCC command arrives.
`slaveMrlChange`	This function will be called when SETMRL CCC command arrives.
`slaveResetDaa`	This function will be called when RSTDAA CCC command arrives.
`slaveBusconUp`	This function will be called when SETBUSCON CCC command arrives.
`slaveFlushDone`	This function will be called when flushing of SDR RX/TX is done.

6.8. I3C_DeviceDescriptor_s

Type: struct

Description

I3C/I2C Device characteristics.

Fields

`i2cIdx`	I2C Legacy Virtual Register Index. Not used for I3C Devices.
`i2cFmPlusSpeed`	I2C Device operation speed. If true then Device works in FM+ Speed. Otherwise it works in FM Speed. Not used for I3C Devices.
`i2cRsvd`	Reserved for 15 available codes for describing the Device capabilities and functions on the sensors system.

	Not used if Device is in I3C Mode.
`i2c10bAddr`	If true, then addr contains 10-bit address for of the Legacy I2C Slave. If false, then the address is 7-bit. Not used for I3C Devices.
`hdrCapable`	Is I3C Device Capable of performing High Data Rate transfers. Not used in I2C Mode.
`legacyI2CDev`	If true then it is a Legacy I2C Device. Otherwise it is a I3C Device
`isActive`	Is the Device active.
`addr`	7-bit Device address in case of I3C Device. 7-bit or 10-bit Device address in case of Legacy I2C Slave.
`provIdHi`	Provisional ID value bits 47:32. Bits 47:33 - MIPI Manufacturer ID. Bit 32 - Type Selector (0 - Vendor Fixed Value, 1 - Random Value).
`provIdLo`	Provisional ID value bits 31:0. If Type Selector (Provisional ID bit 32) is set to 1, then: Bits 31:16 Part ID, Bits 15:12 - Instance ID, Bits 11:0 - left for definition with additional meaning. Please refer to the MIPI I3C specification chapter 5.1.4.1.1 for additional information. If Type Selector is set to 0, then this field will receive random 32-bit value from the Device.
`bcr`	Bus Characteristic Register. I3C only.
`dcr`	Device Characteristics Register. I3C only.
`priv`	For driver's internal use only.

6.9. I3C_SlaveDescriptor_s

Type: struct

Description

I3C/I2C Slave descriptor .

Fields

`da`	Slave's Dynamic Address assigned by the Master.

`dcrType` Slave's Device Characteristics Register value.

`bcrType` Slave's Bus Characteristics Register value.

`sa` Slave's original I2C Static Address (0 if no address).

6.10. I3C_SlaveDescriptors_s

Type: struct

Description

Structure for holding multiple I3C/I2C Slave descriptors.

Fields

`slave_count` Number of Slaves.

`descriptors` Array of slave descriptors.

6.11. I3C_IbiSirConfiguration_s

Type: struct

Description

Configuration of IBI capable I3C device.

Fields

`ibiResp` IBI response that will be sent to requesting device.

`addr` 7-bit Device address in case of I3C Device.

`ibiPayloadSize` Expected IBI payload size.

`ibiPayloadSpeedLimit` Speed limit of payload transfer from IBI requesting device.

`ibiReqDevRole` Role device configured for IBI.

6.12. I3C_Ibi_s

Type: struct

Description

Strucutre describing IBI.

Fields

`ibi_type` The type of an IBI request.

`xfer_bytes`	Number of bytes transferred.
`error`	Indicates if error occured.
`slv_id`	ID of slave issuing IBI.
`acked`	Indicates if IBI was acked or nacked.

6.13. I3C_CommandDescriptor_s

Type: struct

Description

Structure describing command.

Fields

`commomCommandCode`	CommomCommandCode of a command.
`payload`	Pointer to data of command.
`payloadSize`	Size of the payload when payload provided or 0 if there is no payload to be transferred.

6.14. I3C_cmd_t_s

Type: struct

Description

Structure describing command.

Fields

`id`	ID of command.
`cmd`	Two command words combined.
`payload_sdr`	Pointer to data of command.
`payload_ddr`	Pointer to data of command.
`payload_size`	Size of the payload when payload provided or 0 if there is no payload to be transferred.
`priv`	Used internally for most of the CCC's.
`is_wr_cmd`	Field indicates if it is a write command.
`in_cmd_fifo`	Indicates if command has been written to command fifo.
`bytes_fared`	Number of bytes that was either send or received.
`payload_iterator`	Points which byte of payload is read or written.

	Used as index of payload_sdr or payload_ddr array
`ddr_crc`	CRC of DDR.
`transfer_in_progress`	Reading from/to FIFO in progress.

6.15. I3C_cmd_list_element_s

Type: struct

Description

List of commands to send.

Fields

`cmd`	Command.
`used`	Indicates if this node is used.

6.16. I3C_cmd_list_s

Type: struct

Description

List of commands to send.

Fields

`capacity`	Maximum number of descriptors in the buffer.
`buffer`	Number of descriptors in the buffer.
`count`	Number of descriptors in the buffer.

6.17. I3C_CsrData_s

Type: struct

Description

information about CSR

Fields

`csr`	CSR address.
`csr16`	Specifies if CSR address is 16-bit (true) or 8-bit (false).

6.18. I3C_PayloadData_s

Type: struct

Description

Information about payload.

Fields

`sdr_payload` Pointer to SDR buffer containing payload to be sent.

`ddr_payload` Pointer to DDR buffer containing payload to be sent.

`payloadSize` Size of the payload [B].

6.19. I3C_TransmissionParameters_s

Type: struct

Description

Structure contains information about transmission parameters.

Fields

`broadcast` Determines if the Message is addressed to the specific I3C Device (false) or to all I3C Devices (true).

`immediate` If true, the Command will be send via priority queue.

`rsbc` If true, the Repeated Start Between Commands is enabled.

6.20. I3C_PrivData_s

Type: struct

Description

Driver's Private Data.

Fields

`regs_base`

`isr_en`

`use_fifos`

`num_gpo`

`num_gpi`

`max_devs`	
`max_ibi_devs`	
`callbacks`	
`devs`	
`interrupt_config`	Interrupt configuration.
`slaveInterruptConfig`	Slave interrupt configuration.
`threshold_config`	Threshold configuration.
`dev_role`	
`bus_mode`	
`cmd_comp`	
`cmd_empty`	Indicates if software command FIFO is empty.
`cmd_abort`	
`cmd_in_progress`	
`imd_in_progress`	
`cmd_fifo_used`	
`cmd_fifo_empty`	Indicates if hardware command FIFO is empty.
`tx_fifo_used`	
`rx_fifo_used`	
`wait_for_rx`	
`cmd_fifo_size`	
`ibi_fifo_size`	
`cmd_fifo_threshold`	
`tx_fifo_size`	
`rx_fifo_size`	
`imd_cmd`	
`cmd_list`	
`CMD_LIST_BUFFER`	
`tx_cmd_in_progress`	Indicates if tx fifo write command is in progress.
`rx_cmd_in_progress`	Indicates if rx fifo read command is in progress.

`next_cmd_id`	Store next CMD ID.
`ibi`	Stores last IBI.
`devs_active`	Mask which indicates which devices are active.
`hj_devices`	Mask which indicates which devices hoined the bus.
`max_ibi_payload_size`	permissible max count of ibi payload.
`num_apb_rw_csrs`	permissible max count of user purpose regs accessible through apb.
`num_apb_ro_csrs`	permissible max count of user purpose regs accessible through apb.
`num_gpi_csrs`	permissible max count of user gpi.
`num_gpo_csrs`	permissible max count of user gpo.

6.21. I3C_AsfInfo_s

Type: struct

Description

Information about ASF in I3C controller.

Fields

`regs_base`	ASF register start addresses.

6.22. I3C_MaxReadLength_s

Type: struct

Description

Information about max.

read length.

Fields

`payload_length`	Max. read length for regular payload.
`ibi_payload_length`	Max. read length for In-Band Interrupt payload.

6.23. I3C_MaxDataSpeed_s

Type: struct

Description

Information about max.

data speed.

Fields

`read_speed` Max.

read spead

`write_speed` Max.

write spead.

6.24. I3C_Tcam0Data_s

Type: struct

Description

Information about TCAM0 data on GETXTIME.

Fields

`supportedModes` Supported Mode Bytes.

`stateByte` State Bytes.

`freqByte` Frequency Byte.

`InaccuracyByte` Inaccuracy Byte.

6.25. I3C_GroupDescriptors_s

Type: struct

Description

Structure for holding details of perticular Group.

Fields

`group_address` group address

`group_descriptor` Identifier of this group.

`target_count` Number of targets.

`da` dyanamic address of targets

Chapter 7. Driver API Object

7.1. Introduction

API listing for the driver.

The API is contained in the object as function pointers in the object structure. As the actual functions resides in the Driver Object, the client software must first use the global GetInstance function to obtain the Driver Object Pointer. The actual APIs then can be invoked using obj->(api_name)() syntax. These functions are defined in the header file of the core driver and utilized by the API.

7.2. I3C_OBJ

Function Declaration

```
typedef struct I3C_OBJ_s I3C_OBJ();
```

Description

7.3. I3C_GetInstance

Function Declaration

```
I3C_OBJ* I3C_GetInstance();

void
```

Description

In order to access the I3C APIs, the upper layer software must call this global function to obtain the pointer to the driver object.

Return Value

I3C_OBJ* Driver Object Pointer

I3C_OBJ_s

7.4. probe

Function Declaration

```
uint32_t ();

;
```

Parameter List

`config` [in]

Driver and hardware configuration.

`sysReq` [out]

Returns the memory requirements for given configuration.

Description

Returns the memory requirements for a driver instance.

Return Value

0 On success.

EINVAL If config contains invalid values or not supported configuration.

7.5. init

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`config` [in]

Specifies driver/hardware configuration.

`callbacks` [in]

Event Handlers and Callbacks.

Description

Instantiates the I3C Core Driver, given the required blocks of memory (this includes initializing the instance and the underlying hardware).

If a client configuration is required (likely to always be true), it is passed in also. Returns an instance pointer, which the client must maintain and pass to all other driver functions. (except probe).

Return Value

CDN_EOK On success

CDN_EINVAL If illegal/inconsistent values in 'config' doesn't support feature(s) required by 'config' parameters.

CDN_EIO if operation failed

7.6. isr

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object filled by init.

Description

I3C Core Driver's ISR.

Platform-specific code is responsible for ensuring this gets called when the corresponding hardware's interrupt is asserted. Registering the ISR should be done after calling init, and before calling start. The driver's ISR will not attempt to lock any locks, but will perform client callbacks. If the client wishes to defer processing to non-interrupt time, it is responsible for doing so. This function must not be called after calling destroy and releasing private data memory.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.7. start

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Start the I3C driver, enabling interrupts.

This is called after the client has successfully initialized the driver and hooked the driver's ISR (the isr member of this struct) to the IRQ.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.8. stop

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

The client may call this to disable the hardware (disabling its IRQ at the source and disconnecting it if applicable).

Also, a best-effort is made to cancel any pending transactions.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.9. destroy

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

This performs an automatic stop and then de-initializes the driver.

The client may not make further requests on this instance.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.10. enableCore

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Enables the I3C Master Core.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.11. disableCore

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Waits for all operations to be completed and then it disables the I3C Master Core.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.12. setBusMode

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

Bus Mode.

Description

Sets Bus Mode.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.13. getBusMode

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

Pointer to Bus Mode object.

Description

Reads Bus Mode.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.14. setBcr

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`bcr` [in]

BCR value.

`devId` [in]

Retaining register number.

Description

Sets BCR value in one of the retaining registers (DeviceIDx).

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.15. setDcr

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`dcr` [in]

DCR value.

`devId` [in]

Retaining register number.

Description

Sets DCR value in one of the retaining registers (DeviceIDx).

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.16. setPid

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`pid` [in]

Ponter to buffer containing 48-bit Provisional ID value.

`devId` [in]

Retaining register number.

Description

Sets Provisional ID value in one of the retaining registers (DeviceIDx).

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.17. configureDevices

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`devs` [in]

Pointer to the structure containing configuration and capabilities of the Devices.

`numDevs` [in]

Total number of the Devices.

Description

Configures Devices connected to the I3C Bus (including Master) accordingly to the capabilities of these Devices.

Configuration of the Devices must be provided in following order: Master, I3C Slaves with Dynamic Address support, I3C Slaves with static addresses and at the end Legacy I2C Devices. Position of the Device configuration in each group determines priority of the Device. Configuration of at least two Devices (including Master) need to be provided. I3C Master Core must be disabled before calling this function.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.18. configureDevice

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`dev` [in]

Pointer to the structure containing configuration and capabilities of the Device.

`index` [in]

Index of the device corresponding to index of devs array (field of the I3C_PrivData structure).

Description

Configures Device connected to the I3C Bus (including Master) accordingly to the capabilities of this device.

I3C Master Core must be disabled before calling this function.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.19. configurePrescalers

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`config` [in]

Specifies driver/hardware configuration.

Description

Calculates prescaler values for Legacy I2C and SDR frequencies.

Please make sure core is disabled.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.20. clearRrOfDevice

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`device_no` [in]

Number corresponding to Device position in retaining register.

Description

This function clears retaining registers set for a chosen device.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.21. getNewDevices

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`hj_devices` [out]

This mask indicates which devices joined the bus.

Description

Returns devices which joined (by HJ) the bus since last check.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.22. configureInterrupts

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`interruptConfig` [in]

Interrupt configuration.

Description

Enables interrupts chosen by user.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.23. configureThresholds

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`thresholdConfig` [in]

Threshold configuration.

Description

Sets thresholds to levels selected by user.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.24. cmdExec

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Executes all queued Commands.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if Commands are already being processed

CDN_EOPNOTSUPP if Command list is empty

7.25. cmdExecImmediate

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Executes Immediate Command.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.26. enableMcs

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Enables Manual Command Start.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.27. disableMcs

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Disabled Manual Command Start.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.28. manualCommandStart

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Executes commands stored in HW.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.29. cmdCount

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Returns number of queued Commands.

Return Value

0 if input parameters are invalid or if there are no queued Commands

7.30. cmdMaxCount

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Returns maximum number of Commands that can be put in queue.

Return Value

0 if input parameters are invalid or if there are no queued Commands

7.31. cmdClearAll

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Clears Commands queue.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EOK on success

7.32. devPrint

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Prints info on I3C devices on the bus.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EOK on success

7.33. enableTc

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Enables decoding of timing information in Master.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.34. cmdAddPrivWrite

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`da` [in]

Device Address to which data will be transferred.

`csr_data` [in]

CSR information with CSR address and address lenght.

`payload_data` [in]

Pointer to structure with payload information

`xmitMmode` [in]

Transmission Mode.

Description

Adds SDR Mode private Write Command to the Commands queue.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.35. cmdAddPrivRead

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`da` [in]

Device Address from which data will be transferred.

`csr_data` [in]

CSR information with CSR address and address lenght.

`payload_data` [in]

Pointer to structure with payload information

`xmitMmode` [in]

Transmission Mode.

Description

Adds SDR Mode private Read Command to the Commands queue.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.36. cmdAddPrivI2CWrite

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`da` [in]

Device Address to which data will be transferred.

`payload` [in]

Pointer to buffer containing payload to be sent.

`payloadSize` [in]

Size of the payload [B].

Description

Adds Legacy I2C SDR Mode private Write Command to the Commands queue.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.37. cmdAddPrivI2CRead

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`da` [in]

Device Address from which data will be transferred.

`payload` [in]

Pointer to buffer containing payload to be sent.

`payloadSize` [in]

Size of the payload [B].

Description

Adds Legacy I2C SDR Mode private Read Command to the Commands queue.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.38. cmdAddDdrWrite

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd`	[in]
	Pointer to driver's private data object.
`da`	[in]
	Device Address to which data will be transferred.
`command`	[in]
	DDR Command.
`hdrMode`	[in]
	Specifies which one of the HDR modes to use.
`payload_data`	[in]
	Pointer to structure with payload information

Description

Adds DDR Mode private Write Command to the Commands queue.

I3C Core Driver will automatically calculate CRC, Preambles and Parity Bits for the DDR Command and its Payload.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.39. cmdAddDdrRead

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd`	[in]

Pointer to driver's private data object.

`da` [in]

Device Address from which data will be transferred.

`command` [in]

DDR Command.

`hdrMode` [in]

Specifies which one of the HDR modes to use.

`payload_data` [in]

Pointer to structure with payload information

Description

Adds DDR Mode private Read Command to the Commands queue.

I3C Core Driver will automatically check CRC, Preambles and Parity Bits for the DDR Command and its Payload.

Return Value

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

CDN_EOK on success

7.40. cmdAddSetSlaveEvents

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`eventsMask` [in]

Mask of Events to be enabled or disabled.

`enable` [in]

If true, specified events will be enabled; if false, specified events will be disabled.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptors

Description

Adds to queue Command to that sends message to I3C Device (or to all I3C Devices in case of a broadcast) which event interrupts should be enabled or disabled.

Return Value

CDN_EOK on success

CDN_EPROTO if slave event mask is not valid

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

7.41. cmdAddEnterActivityState

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`state` [in]

Activity State value.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptor

Description

Creates Enter Activity State Command to be sent to the I3C Slaves.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

7.42. cmdAddResetDaa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`devAddr` [in]

Address of the I3C Device in case of Direct Command. This parameter is ignored in case of Broadcast Command.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for Dynamic Address Assignment process to the Commands queue.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.43. cmdAddEnterDaa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for Dynamic Address Assignment process to the Commands queue.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.44. cmdAddSetMaxWriteLength

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`length` [in]

Maximum Write Length to be set.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for setting the Maximum Write Length for all or specific device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.45. cmdAddGetMaxWriteLength

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`length` [in]

Pointer to 16-bit variable to which Maximum Write Length will be written.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the Maximum Write Length value for a specific device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.46. cmdAddSetMaxReadLength

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`max_read_length` [in]

Maximum Read Length to be set.

`length_size` [in]

Number of bytes required to store Maximum Read Length. Alloved values are 2 or 3.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptors

Description

This function adds CCC for setting the Maximum Read Length for all or specific device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.47. cmdAddGetMaxReadLength

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`max_read_length` [in]

Pointer to a structure where Maximum Read Length.

`length_size` [in]

	Number of bytes required to store Maximum Read Length. Alloved values are 2 or 3.
`devAddr`	[in]
	Address of the I3C Device to which Command will be sent.
`tx_params`	[in]
	Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the Maximum Read Length value for a specific device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.48. cmdAddGetMaxDataSpeed

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd`	[in]
	Pointer to driver's private data object.
`max_data_speed`	[in]
	Pointer to structure with max data speeds.
`devAddr`	[in]
	Address of the I3C Device to which Command will be sent.
`tx_params`	[in]
	Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the Maximum Data Speed (SCL Frequency of the slave)

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.49. getSlavesList

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`slaveDescs` [in]

Pointer the structure of Slave descriptors. Memory will be written with slave count and slave data.

Description

This function gets information about Slaves present on the bus.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.50. cmdAddDefineSlavesList

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`slaveDescs` [in]

Pointer the structure of Slave descriptors.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for informing Secondary Masters about Slaves present on the bus.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.51. cmdAddEnterTestMode

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

Test mode code. 0x00 - Exit Test Mode. 0x01 - Vendor Test Mode. 0x02 - General Test Mode. 0x03-0x0F - MIPI Reserved. 0x10-0xFF - Vendor Definable.

`tx_params` [in]

Transmission parameters - command descriptor

Description

Prepares CCC broadcast Command for all I3C Devices that the Master is entering specific Test Mode.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.52. cmdAddSetBuscon

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`context_buf` [in]

Pointer to bus context data

`length_size` [in]

length of context data

`tx_params` [in]

Transmission parameters - command descriptor

Description

Prepares CCC broadcast Command for all I3C Devices to set bus context(i3c specification version that bus will use)

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.53. cmdAddEnterHdrMode

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

`tx_params` [in]

Transmission parameters - command descriptor

Description

Sends broadcast message to all I3C Devices that the Bus is being switched over to the indicated HDR Mode.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.54. cmdAddSetAaSa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for Dynamic Address Assignment using static address to all target devices in broadcast mode.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.55. cmdSetDaFromSa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`address` [in]

Dynamic Address to be set.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

Sends Command that addresses specific Device with Dynamic Address using its I2C Address.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.56. cmdAddSetNewDa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`da` [in]

Dynamic Address to be set.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

Sends Command that sets a new Dynamic Address for specific I3C Slave Device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.57. cmdAddSetRstAct

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`DefiningByte` [in]

Value corressponding to reset action.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptors

Description

Configure the next reset action for following reset pattern in I3C Device (or in all I3C Devices in case of a broadcast).

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

7.58. cmdAddSetXTime

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`SubCommandByte` [in]

Value of sub command code.

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptors

Description

Exchange event timing information (for Timing Control Async Mode) with I3C Device (or with all I3C Devices in case of a broadcast)

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_EBUSY if there is no space for new Command or payload

7.59. cmdAddSetGroupAddr

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`grpaddr` [in]

Group address value.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

Sends Command that assign a Group Address to an I3C Target supporting the Group Address feature.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.60. cmdAddDefineGroupList

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`groupDescs` [in]

Pointer the structure of Group descriptors.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for informing Secondary Masters mode details about a particular Group,including the list of I3C Targets that are m embers of the Group.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.61. cmdAddResetGrpa

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`grpAddr` [in]

value of group address

`devAddr` [in]

Address of the I3C Device in case of Direct Message.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function add CC to remove one or more I3C target devices from a Group by resetting their assigned Group Address.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.62. cmdAddGetProvisionalId

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buff` [in]

Pointer to variable to which 48-bit Provisional ID will be written.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining provisional ID of device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.63. cmdAddGetBcr

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`bcr` [in]

Pointer to variable to which 8-bit DCR value will be written.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the Bus Characteristics Register.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.64. cmdAddGetDcr

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`dcr` [in]

Pointer to variable to which 8-bit BCR value will be written.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the Device Characteristics Register.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.65. cmdAddGetStatus

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`status` [in]

Pointer to variable to which 16-bit status value will be written.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining status of the I3C Device.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.66. cmdAddGetAccMst

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`recData` [in]

Pointer to data received from device that GETACCMST CCC is sent to. After command execution this should hold address of receiving device.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for granting Mastership.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.67. cmdAddGetXTime

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`tCamData` [in]

Pointer to buf which stores data bytes

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for obtaining the timing control Aysnc Mode 0 data.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.68. CmdSetNCAMode

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

Mode value for enabling/disabling data pattern with sub framing.

`devAddr` [in]

Address of the I3C Device to which Command will be sent.

`tx_params` [in]

Transmission parameters - command descriptor

Description

This function adds CCC for enabling NCA mode in private write.

Return Value

CDN_EOK on success

CDN_EBUSY if Commands are being processed by the Core Driver

CDN_EINVAL if input parameters are invalid

7.69. ibiConfigureDevices

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`ibi_sir_cfg` [in]

Pointer to array that will hold configuration of IBI related devices.

`num_sir_cfgs` [in]

Size of configuration array.

Description

This function configures devices issuing IBI.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.70. ibiModifyDeviceConfig

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`ibi_sir_cfg` [in]

Pointer to struct that holds configuration of IBI device.

`da` [in]

Device DA.

Description

This function modifies device configuration issuing IBI.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOENT if device with DA not found

7.71. ibiGetAddressOfIssuer

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

This function extracts address of device that issued IBI.

Return Value

0 on error invalid

valid I3C address on success

7.72. ibiGetData

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer that will hold received payload.

`payload_size` [in]

Size of payload.

Description

This function extracts IBI payload.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.73. hjConfigureResponse

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`hj_response` [in]

Should I3C controller ACK or NACK Hot Join request.

`hj_disable_set` [in]

Should I3C controller disable slaves from requesting Hot Join.

Description

This function configures response of I3C conftroller to Hot Join request.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.74. configureSlaveInterrupts

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`slaveInterruptConfig` [in]

Interrupt configuration.

Description

Enables chosen interrupts in slave mode.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.75. slaveModeConfigure

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`slaveInterruptConfig` [in]

Pointer to slave mode interrupt configuration.

Description

Configures Hardware and Software to operate in Slave mode.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.76. slaveModeReqSdrRead

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer where data will be stored.

`num_bytes` [in]

Number of bytes to read.

Description

Adds SDR Read request to the queue.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.77. slaveModeReqSdrWrite

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer with data to be sent.

`num_bytes` [in]

Number of bytes to send.

Description

Adds SDR Write request to the queue.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.78. slaveModeReqDdrRead

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer where data will be stored.

`num_words` [in]

Number of bytes to read.

Description

Adds DDR Read request to the queue.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOENT if no DDR data is available for read

CDN_EPROTO if reveived command code is not to DDR Write

7.79. slaveModeReqDdrWrite

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf_in` [in]

Pointer to buffer with data to be sent.

`num_words_in` [in]

Number of 16-bit words to send.

Description

Adds DDR Write request to the queue.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if DDR FIFOs are not implemented

7.80. slaveModeRequestHotJoin

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Sends Hot Join request.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.81. slaveModeMastershipReq

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

Description

Sends Mastership request.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.82. slaveModeRequestIbi

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

`tCam0Event` [in]

TCAM0_EVENT value to sent in ctrl register.

`ibiTCam0` [in]

If IBI request is specific to TCAM0.

Description

Sends In-Band Interrupt request.

Return Value

CDN_EOK on success

CDN_EIO if operation failed

CDN_EINVAL if input parameters are invalid

7.83. slaveModeWriteIbiPayload

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer with In-Band Interrupt Payload.

`numbytes` [in]

Number of bytes to write.

Description

Writes Payload for In-Band Interrupt.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.84. slaveModeReadIbiPayload

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`buf` [in]

Pointer to buffer to capture IBI playload

`numbytes` [in]

Number of bytes to read

Description

Read Payload from IBI.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.85. slaveModeReqFlowControl

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`mode` [in]

flow control mode param

Description

Control flow control NACK/ACK request for next private transfer.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.86. slaveModeReadApbRo

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`idx` [in]

Index of the APB RO register.

`val` [out]

Pointer to variable where APB RO register value will be stored.

Description

Reads APB RO register value.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if there are no APB RO CSRS

7.87. slaveModeReadApbRw

Function Declaration

```
uint32_t ();
```

```
;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`idx` [in]

Index of the APB RW register.

`val` [out]

Pointer to variable where APB RW register value will be stored.

Description

Reads APB RW register value.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if there are no APB RW CSRS

7.88. slaveModeWriteApbRw

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`idx` [in]

Index of the APB RW register.

`val` [in]

Value to be written in the APB RW register.

Description

Writes value to APB RW register.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if there are no APB RW CSRS

7.89. slaveModeReadGpo

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`idx` [in]

Index of the GPO register.

`val` [out]

Pointer to variable where GPO register value will be stored.

Description

Reads GPO register value.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if there are no GPO CSRS

7.90. slaveModeReadGpi

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`idx` [in]

Index of the GPI register.

`val` [out]

Pointer to variable where GPI register value will be stored.

Description

Reads GPI register value.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

CDN_ENOTSUP if there are no GPI CSRS

7.91. slaveModeReadMWL

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

[ ]

Description

This function is for obtaining the Maximum Write Length.

7.92. slaveModeReadMRL

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd`	[in]
	Pointer to driver's private data object.
`maxReadLength`	[in]
	Maximum Read Length value.
`ibiPayloadSize`	[in]
	ibi payload size.

Description

This function is for obtaining the Maximum Read Length.

Return Value

CDN_EOK on success

CDN_EBUSY if operation failed

CDN_EINVAL if input parameters are invalid

7.93. getFifoFillLvl

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`fillLvl_tx` [in]

no of words available in TX FIFO.

`fillLvl_rx` [in]

no of words availale in RX FIFO.

Description

This function is for obtaining the fill level for TX_FIFO and RX_FIFO in SDR.

Return Value

CDN_EOK on success

CDN_EBUSY if operation failed

CDN_EINVAL if input parameters are invalid

7.94. setSdrFifoFlush

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`flush_tx` [in]

flush TX FIFO.

`flush_rx` [in]

flush RX FIFO.

Description

This function is flush TX and RX fifo in SDR.

Return Value

CDN_EOK on success

CDN_EBUSY if operation failed

CDN_EINVAL if input parameters are invalid

7.95. slaveGetIbiStatus

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`ibi_stat` [in]

IBI pending status.

`hj_stat` [in]

Hot join request pending status.

`mr_stat` [in]

Mastership change request pending status.

Description

This function is for obtaining the IBI/HJ/MR request status.

Return Value

CDN_EOK on success

CDN_EBUSY if operation failed

CDN_EINVAL if input parameters are invalid

7.96. getAsfInfo

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`asf_info` [in]

Pointer to ASF information structure.

Description

Retrieves ASF information from I3C controller.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid

7.97. checkOperationMode

Function Declaration

```
uint32_t ();

;
```

Parameter List

`pd` [in]

Pointer to driver's private data object.

`opMode` [out]

Pointer to OperationMode structure.

Description

Checks controller operation mode.

Return Value

CDN_EOK on success

CDN_EINVAL if input parameters are invalid