I3C Slave peripheral Programmer's Model and Arch

Background on I3C, Slave arch, and details on config

Revision history

Revision	Date	Description	Author
1.3	10/23/19	Major update to Map scheme with auto-map and new MAPCTRL register.	PK
1.2d	01/27/19 05/16/19 06/03/19 06/18/19	Added new RSTACTTIME register. New optional Regs: CCCMASK, HDRCMD, VGPIO. Added ERRWARNMASK. DYNADDR now reads back with causing CCC if writable DYNADDR reg by param. HDRCMD now 2 bits to control which.	PK
1.1	03/13/18 10/5/18 10/12/18 10/20/18 10/23/18 11/09/18 12/21/18	Added Mapped address handling in DYNADDR and new MSGMAPADD register. Added SLVRST cause register to STATUS. DYNADDR has support for read-back, NACK control, and 10bit static addr (1). Also, EXTDATA now in CTRL to allow IBI data after MDB using TXFIFO. Added MAPIDX to CTRL and moved EXTDATA. Added I2CREV (ID) if i2c device id used to IDEXT register. Added IBI FIFO support regs. Show SLVRST in INTSET/INTCLR Added WDATAB1 for byte-only for DMA (no END bits)	PK
1.04c	03/17/17 4/15/17 6/29/17	Spec should now be complete to match the Slave use. Added some clarifications. Clarified some more details on errors. Added note on when EVENT status is signaled. IBIOK and the like now IBIDIS. Vendor ID and Time control freq/acc can be an MMR now. Added OFFLINE and DYNADDR optional use.	PK
0.91d	11/21/16	Added DDRMATCH status and moved CHANDLED and EVENT and EVDET. Added TXFULL and RXEMPTY bits as well as OWRITE error (over write). Clarify status raw bits that are SDR only vs. HDR. S0/S1 handling.	PK
0.9c	08/22/16	ERRWARN vs. error. CTRL bit to flush TB buff. Now just WDATAB and WDATABE and RDATAB. Removed END from DATACTRL. Moved FLUSH to 0. BAMATCH register field, FLUSHFB added. Mistake in the offset of CONFIG in title.	PK
0.8d	7/30/16	Added more status/interrupt causes. Added CCC vs. CHANDLED. Fixed INTSTAT on CHANDLED. Defined ID reg.	PK
0.3	4/4/16	Updated with new regs and config info.	PK

Contents

1. Introduction

MIPI I3C is a follow on to i2c which has major improvements in use and power, as well as providing an alternative to SPI for mid-speed. In particular:

- 2 wire multi-drop bus capable of 12MHz clock speeds with up to 11 devices
  - While using standard pads (vs. i2c special pads) with 4mA drive
  - Slave addresses are dynamically assigned – does not require a static address
    - But, slaves **may** have a static address at start
  - Slaves may use inbound clock as the peripheral clock
    - So devices may have slow/inaccurate clocks internally
  - For read from Slave, Slave normally ends the read, but Master may terminate
    - Unlike i2c and SPI with the problems of Master having to “know” length
- In-Band interrupts, allowing Slaves to notify Master
  - Can be both equivalent to a separate GPIO, but can also be directly data bearing
  - Prioritized so that if multiple Slaves wish to interrupt at the same time, the order is resolved
    - Dynamic addresses used for this, so priority controlled by Master
  - Interrupts can be started even when Master is not active on the bus, and yet no free running clock needed
  - Time-stamping option to allow resolution of initial event vs. when interrupt gets through
- Built-in Commands in separate “space” to not collide with normal Master->Slave messages
  - Controls bus behavior, modes and states, low power state, enquiries, etc.
  - Has additional room for new built-in commands to be used by other groups
- Organized forms of multi-master:
  - Slave which can request Master to allow it to message another Slave, yet not needing to generate its own clock – called Peer-to-Peer
  - Secondary Masters which can use clean handoffs between each Master
- Hot-join onto bus allows devices to come on-line later than initial bus bringup
  - May be due to late wake up (power up) or physical insertion
  - Provides clean method for notification.
- Mixed i2c and i3c capable
  - I3C has support for certain legacy i2c devices on the bus
  - I3C Slave devices capable of operating on i2c buses
  - Also support for bridging (to i2c, SPI, UART, etc.)
- High data rate modes also optionally available
  - Only Master and the specific Slave has to support – other Slaves know how to ignore
  - Has an HDR-DDR form which is about 2x the data rate of SDR (so about 20Mbps)
  - Has an HDR-TSP (ternary symbols) which are up to 3x the data rate (so about 30Mbps)
- Slaves as small as <2K gates.
  - Allowing for fully state machine driven as well as using processor
- Masters as small as 2.5K gates
  - Relies on processor to handle

The I3C peripheral supports the full feature set, but uses parameterization to allow reduction of the logic to what is needed.

2. Master vs. Slave for I3C

The I3C peripheral contains both a Master and Slave component, and can be parameterized to be only one or the other or both. However, if Master, the peripheral should generally support Slave mode to facilitate master handoff. This spec covers only the Slave component. The Master spec covers the Master with the understanding that Master+Slave would be integrated in that use.

2.1 Requirements for Master

Master mode is expected to be backed up by full software to support the various requirements of the Master (including as Secondary Master):

- Special handling of ENTDAA – assign dynamic addresses to each slave. This is a process. It is supported by the peripheral, but requires the Software to make choices.
- Building up a table of Slaves, including their capabilities. This is used to control what kinds of actions and commands may be sent to them.
- Handling requests such as In-band interrupt, Peer-to-peer slave, and Master request (handoff).
- Adjusting clock speed or write-to-read timing to match the slave's limitations. This can be done in HW in terms of dividers and uneven duty cycle, but the SW needs to make the decisions.
- Adjusting max data length.
- Being a slave after a Master handoff to another Master

Additionally, Master needs 3 pins normally, unless a very precise high strength pullup is included in the SDA pad. The 3 pin model allows one pad/pin to enable a system-provided pullup which is sized to the needs of the system.

The Master needs a reasonably accurate clock, normally capable of a frequency which is a multiple of between 11MHz and 12.5MHz. For example, 24MHz, 48MHz, 25MHz, 50MHz, 33MHz, and 66MHz are all typical speeds. Higher ratios are also quite usable.

2.2 Requirements for Slave

The I3C Slave mode is intended to allow a much simpler model. Further, the block is designed to allow the implementation to parameterize many aspects when space is a concern. So, the slave minimum requirements are very small (just I3C SDR mode and a few CCC commands). But, the slave can be scaled up quite a bit in terms of functionality, which can actually save area over software 5ost. What is important is that the Slave can be scaled in many cases by handling more in software or by handling in hardware. This decision is made in the build of the block.

2.3 I3C Slave acting as a normal I2C Slave on I3C buses

The I3C Slave, if assigned an i2c Static Address, will act as a normal I2C device when it 1st comes up. If placed on an i2c bus with an i2c Master, it will simply stay in i2c mode and operate normally. The SW knows this is the case because (a) there has not been a DACHG indicating a Dynamic address was assigned, and (b) the DYNADDR register contains 0x0.

If full i2c support for Fm and Fm+ is desired, the pads should also support a 50ns Spike filter, which must be turned off when the I3C 7'h7E broadcast address is seen (indicating an I3C Master); that turning off of the Spike filters could be done by HW using the raw net indicating that address was seen (see uarch spec for details), or it could be handled by software. Note that a 50ns Spike filter is not strictly needed to operate on an i2c bus, so this is not a requirement of the peripheral in any way.

Depending on configuration, the block supports most I2C features, with the exception of clock stretching. These features include Extended 10-bit address. DeviceID, SW reset, and HS mode (affects pads).

2.4 Understanding Offline and Hot-Join for re-joining bus

Hot-join is normally thought of as being used for Hot-insertion type uses, but it may also be used when the Slave may be powered or hard reset such that it needs to rejoin the I3C bus and will need a new Dynamic Address (DA).

The Slave provides 3 mechanisms for different cases of rejoining:

1. If DA is lost, Hot-Join is used.

2. If DA is retained in the peripheral (e.g. with SRFFs) then the CONFIG.OFFLINE bit will be used (see below).
3. If DA is retained in separate always-on memory, then the DYNADDR register as writable mechanism may be used along with the CONFIG.OFFLINE bit.

The OFFLINE bit of the CONFIG register may be set when SLVENA is set to 1. This will cause the peripheral to safely rejoin the bus. It does this by ensuring the I3C bus is not in HDR mode, using the same approach as S0/S1 exit does: wait for 1st of HDR Exit Pattern, or 60µs of SCL and SDA unchanging.

Note that after using OFFLINE, the peripheral still cannot safely use IBI until it sees a STOP (which ensures that the next START is safe to use for IBI).

If the App needs to do an IBI right away, it should wait for the 1st of STOP (see STATUS), or 200µs of SCL and SDA being High. This can be done using the STATUS and INTSET controls. If the STATUS indicates the bus is not busy and the peripheral will interrupt on START or STOP, then use of a timer to measure 200µs can be used. If the timer goes off with no START or STOP, then it is safe to use IBI. If a START causes an interrupt, the timer should be turned off and STOP waited for. If STOP causes an interrupt, it is safe to use IBI.

2.5 Reset the peripheral safely in a system

The I3C peripheral is mostly decoupled from the device-system it is part of, other than being a bus-slave addressable by the processor and DMA – so a passive peripheral. It does have some signals that go into the system (e.g. IRQ, DMA request, wake, etc), so resetting it should be done in a way that does not cause uncertainty to the system (e.g. spurious interrupts, etc). Likewise, it should be done so as to not cause problems on the I3C/I2C bus as well.

This is done by a sequence of simple steps to decouple the peripheral from the rest of the system in terms of active signals:

1. Making sure the peripheral is not actively writing to the Master (not in an i2c or I3C Read), write to the CONFIG register and disable the slave (SLVENA). This will ensure it is not responsive in terms of ACK/NACK, etc. It will also make sure states do not change while following the steps below.
2. If DMA is active, turn off the channel/channels associated with this peripheral 1st, then turn off the DMACTRL register.
3. Turn off interrupt enables in the processor for this peripheral, then mask off interrupts by writing INTCLR (to mask off all interrupts feeding the IRQ).
   The underlying sources can be cleared if wanted (but will have no effect when interrupts masked off) by writing to the STATUS and ERRWARN regs.
4. If any data is in TX or RX FIFO, use the DATCTRL register to flush.
5. If a wake system is enabled in the system, it may be necessary to mask off this peripheral from being a wake source. Although these steps would be done by an active processor, the reset can cause changes in wake signaling, which may matter to the system.
6. If an I3C Slave Reset block is used, it *can* be disabled. Once the slave is not enabled, it is possible that the I3C Master would attempt to reset (and the Slave Reset block will see this and follow the default reset mode). One option is to turn the SDA and/or SCL pads to Hi-Z GPIO mode (with no pull resistors) to decouple it (the Slave Reset block reacts only when both signals are changing per the spec).
7. If using a larger system (e.g. AXI/ACE, PCIe, OCP, etc), make sure data barriers are flushed so no transactions to this peripheral can happen while it is being reset. This peripheral does not ever use PREADY=0 (it is 1 always), so it cannot stall the bus. It can use PSLVERR (equal to HRESP) based on config, but only during an invalid write transaction.
8. The block can now be reset using a wrapper or system controlled override to PRESETn. This will reset all flops in the system, including in other clock domains. It is safe to synchronize both the

assertion and de-assertion to the PCLK (e.g. from HCLK) since flops in other domains will be inactive and not affected by D input on de-assertion.

2.6 Use of Mapping (when enabled) for multiple virtual Slaves

The I3C Slave provides mechanisms to support multiple Virtual slaves. So, along with the base primary Address (starts with legacy i2c static address if any, then moves to I3C dynamic address if on an I3C bus), the application may have control of multipole other addresses.

There are 4 basic uses of this feature, and they can be mixed and matched:

1. The ability to have more than one i2c static address, as some devices support. This could be a group address or other 2nd address.
2. The ability to have an i2c 10-bit address. Exactly 1 may be used, and it will be in Map entry 1 if used.
3. The ability to have virtual I3C slaves for various uses, each with a separate Dynamic address. This is distinct from any in-built Debug feature using a 2nd address.
4. The ability to bridge between I3C and another bus, so virtualizing the I3C addresses to match to each bridged device.

To support the above goals, the block provides 5 features:

- The block can be pre-built to support from 1 to 8 Map slots.
- The MAPCTRLn registers can be used to read and write addresses, including static addresses. Additionally, the older DYNADDR register model can be used. They can be mixed as they operate on the same information.
- The MSPMAPADDR register can be used to determine the last address that was requested, as well as a history of the previous 2 messages.
- The CRTL register may indicate which Address is sending an IBI.
- The MAPCTRLn register can be used to configure for Auto-mapping if enabled for it. If the block was pre-built for auto-map, it can do any of:
    - Participate in SETDASA uses such that Mapped static addresses are matched and converted to Dynamic addresses.
    - Participate in SETAASA uses such that all Mapped static addresses (as well as Primary) are converted to the same address as Dynammic.
    - Participate in ENTDAA using replacements for part of the PARTID, and optionally the DCR. Those replacements may be constants pre-built or may be written using the MAPCTRLn interface. Each one that is enabled will then participate in ENTDAA.

3. Registers of Digital block

The I3C block uses parameterization by the system when instantiating, so the peripheral necessarily has registers and fields which may be available or removed depending. To help clarify this, a RO register is used to indicate the blocks capabilities. Additionally, the table below shows what is affected on a broad basis.

Offset	Type	Name	Description	Configuration?
0x000	RW	reserved	Reserved for Master.	-
0x004	RW	CONFIG	Configuration fields for things setup before block is activated	FIELDS: Choices in build affect fields of these registers

Offset	Type	Name	Description	Configuration?
0x008	RO and R/W1C	STATUS	Status for Master and Slave. Not all bits used if only Slave. This document only explains the slave bits.	
0x00C	RW	CTRL	Control for active use – in particular event generation (e.g. interrupts to the Master)	
0x010	R/W1S	INTSET	Interrupt enable	REG and/or FIELDS: Choices may remove the registers or impact which interrupts are used.
0x014	W1C	INTCLR	Interrupt disable	
0x018	RO	INTMASKED	Mask of sourced interrupts (STATUS & INTSET)	
0x01C	R/W1C	ERRWARN	Error and Warning state from protocol errors and issues. Related to ERRWARN status and interrupt.	Always available
0x020	RW	DMACTRL	DMA control register, if DMA is enabled in the block.	Only if DMA selected.
0x02C	RW	DATACTRL	Allows control of data buffering and indicates current buffer state. If FIFO enabled, this is also the FIFO control.	FIFO Fields only if FIFO enabled.
0x030	WO	WDATAB	Write a byte of data, including use of a 9th bit to mark as end (last byte)	Available unless external FIFO
0x034	WO	WDATABE	Write a byte of data, which is end (last byte).	
0x038	WO	WDATAH	Write a half-word of data, including use of a 16th bit to mark as end (last byte of half-word is end).	
0x03C	WO	WDATAHE	Write a half-word of data, which is end (last byte of half-word is end).	
0x040	RO	RDATAB	Read a byte of data	
0x048	RO	RDATAH	Read a half-word of data	
0x050	WO, RO	WIBIDATA	Writes to IBI FIFO (2 entry) if EXTDATA is set in CTRL (bytes after MDB) and IBI FIFO enabled. Reads back with 1 if space to take another byte.	Only if enabled, else normal TXFIFO and so WDATAB/WDATAH are used

Offset	Type	Name	Description	Configuration?
			May write-clear as well (e.g. if master terminates IBI Data). Note: the normal Read Errors is used if Terminate, IBI Urun is used for underrun.	
0x054	RW	WDATAB1	Byte only WDATAB for DMA use. Ignores bits [31:8].	Always available since no extra cost.
0x05C	RO	CAPABILITIES2	2nd set of capabilities	Always used, but will be 0 in older versions
0x060	RO	CAPABILITIES	Indicates what is available/supported in this block, including Master and/or Slave, HDR modes, etc.	Always used – indicates how block is configured by HW
0x064	RO/RW	DYNADDR	Dynamic address once assigned, else 0.	Always used. May be RO, or may be configured to be RW
0x068	RO/RW	MAXLIMITS	Indicates the limits set by the Master (or the original requested limits).	If enabled to track this.
0x06C	RW	IDPARTNO	Allows application to write the ID part-number if configured to provide by register	Only available if configured by ID48B.
0x070	RW	IDEXT	Allows application to write the ID extension of DCR and/or BCR and/or instance if so configured.	Separate configuration for each field.
0x074	RW	VENDORID	Allows application to write the Vendor ID if so configured.	If ID is completely from MMRs/Nets and not constant.
0x078	RW	TCCLOCK	Allows application to set Time control clock and accuracy info dynamically.	If time-control is enabled and the register form is used (vs. constant).
0x07C	RO	MSGMAPADDR	Returns the index of the matching Dynamic address or Static address.	Only when mapped addresses are enabled and used.
0x100	RW	RSTACTTIME	Sets the RSTACT time response values.	Only when enabled as a register vs. parameter.
0x104	RW	VGPIO	Supports special VGPIO mechanism (outside of I3C spec)	Only if enabled
0x108	RO	HDRCMD	Holds HDR Cmd value if enabled (vs. RXFIFO)	Only if enabled to support and only if enabled by CONFIG.
0x10C	RW	CCCMASK	Used to mask on/off Unhandled CCCs.	Only if enabled, else all unhandled go to App.

Offset	Type	Name	Description	Configuration?
0x110	RW	ERRWARNMSK	Used to Mask on/off ERRWARN bits that contribute to STATUS.ERRWARN (and interrupt)	Only if enabled, else all contribute.
0x11C, 0x120, 0x124, 0x128, 0x12C, 0x130, 0x134, 0x138, 0x13C	RW	MAPCTRL0 up to MAPCTRL8	Optional MAP control registers for primary and MAP regs (vs. DYNADDR). Features based on config.	Only if MAP enabled, and number of registers based on MAP_CNT, and features based on MAP_DA_AUTO
0xFFC	RO	ID	Block ID and revision.	Controlled by parameter and could be 0.

3.1 CONFIG @ 0x004 – I3C Configuration register

The I3C configuration register allows for configuration before enabling the bus.

Bit(s)	Type	Name	Description	Config
0	RW	SLVENA	If 1, allows the Slave to operate on the i2c or I3C bus. If 0, the Slave will ignore the bus. This should not be set until registers such as this one, PARTNO, IDEXT, and the like are set 1st – if used – since they impact data to the Master. This is normally only setup once before bus comes up. If used at other times, see Hot-Join; in the case of Hot-Join, that should be set before this, so the device does not see a START or STOP incorrectly. **Default:** 0 unless configured to be 1 by default.	Always available.
1	RW	NACK	If 1, the Slave will NACK all requests to it except CCC broadcast. This should be used with caution as the Master may determine the slave is missing if overused.	
2	RW	MATCHSS	If 1, the START and STOP sticky STATUS bits will only be set if MATCHED is set. This allows START and STOP to be used to detect end of a message to/from this Slave.	
3	RW	S0IGNORE	If 1, the Slave will not detect S0 or S1 errors and so not lock up waiting on an Exit	

			Pattern. This should only be used when the bus will not use HDR.	
4	RW	DDROK	If 1, allow HDR-DDR messaging if available by setting the corresponding BCR bit to say HDR is available, and the corresponding HDRCAP HDR-DDR bit and permitting use.	Ignored if the corresponding HDR is not enabled. Also ignored if the BCR[5] bit is hard set to 1 by parameter (constant) or is 1 in the IDEXT.BCR register field.
5	RW	BTOK	If 1, allow HDR-BT messaging if available.	
8	RW	IDRAND	If 1, PARTNO is a random value. If 0 (or not configured for), PARTNO is a part number and instance.	Only if enabled for in the block along with PARTNO register.
9	RW	OFFLINE	If 1 when SLVENA set to 1, will wait for either 60us of bus quiet or HDR Exit Pattern; this ensures that bus is not in HDR mode and so can safely track SDR.	Only possible if CLK_SLOW is used, so only if IBI or MR or HJ.
11:10	RW	HDRCMD	If not 0, then HDR command (e.g. HDR-DDR Cmd) byte will be put into the HDRCMD register vs. in RXFIFO. The interpretation is: **Value** / **Meaning** 0 – HDR Cmd will be in RXFIFO for Write and Read 1 – HDR Cmd for Write and Read will be in HDRCMD reg 2 – HDR Cmd for Read will be in HDRCMD reg, RXFIFO for Write 3 – Reserved	Only if enabled to support.
23:16	RW	BAMATCH	Bus Available condition match value for current “Slow clock”. This provides the count of the slow clock to count out 1us (or more) to allow an IBI to drive SDA Low when the Master is not doing so. The max width, and so max value, is controlled by the block. **Default**: set by parameters if configured.	Only if enabled for events such as IBI or MR or HJ, and if enabled to provide this as a register. Width is limited to CLK_SLOW_BITS
31:25	RW	SADDR	If allowed by the block: Sets i2c 7 bit Static address, else should be 0.	If enabled to use one and to be provided by SW. Block may provided in HW as well.

3.2 STATUS @ 0x008 – I3C Status register

The Status register is used to indicate both sticky status for interrupts as well as “states” and “modes” related to the i3c bus. The fields are divided into current activity, interrupt maskable actions, then states and modes on the bus.

Bit(s)	Type	Name	Description	Config
Activity status				
0	RO	STNOTSTOP	Is 1 if bus is busy (activity) and 0 when in a STOP condition. Other bits below may also be set when busy. Note that this can also be true from an S0 or S1 error, which waits for an Exit Pattern.	
1	RO	STMSG	Is 1 if this bus Slave is listening to the bus traffic or responding. If STNOSTOP=1, then this will be 0 when a non-matching address seen until next repeated START or STOP.	Will include unhandled CCCs
2	RO	STCCCH	Is 1 if a CCC message is being handled automatically.	Which CCCs handled is configured.
3	RO	STREQRD	Is 1 if the REQ in process is an SDR Read from this Slave or an IBI is being pushed out. See STHDR for DDR handling.	
4	RO	STREQWR	Is 1 if the REQ in process is SDR write data from the Master to this bus Slave or all i3c Slaves, but not ENTDAA mode. See STHDR for DDR handling.	
5	RO	STDAA	Is 1 if in ENTDAA mode whether this bus Slave has a Dynamic Address or not.	
6	RO	STHDR	Is 1 if in HDR-DDR, HDR-TSP, or HDR-TSL; is 1 whether the HDR mode is supported by this block or not, and whether the message is to this block or not.	
Interrupt maskable actions. Write 1 to clear/acknowledge				
8	R/W1C	START	Set to 1 if a START or repeated START seen since last cleared. This is not usually needed, but can be used for wake events.	
9	R/W1C	MATCHED	An incoming header matched this device’s I3C Dynamic or I2C Static address (if any) since last cleared.	
10	R/W1C	STOP	A STOP state was present on the bus since last cleared. The STNOTSTOP state will also indicate if in stop. Note: A fast STOP/START combination may not trigger this status. START will always be set in that case. This bit is from a stopped state being detected.	

11	RO	RXPEND	Receiving a message from Master, which is not being handled by block (not a CCC internally processed). For all but External FIFO, this uses DATACTRL RXTRIG, which defaults to not-empty. If DMA is enabled for RX, DMA will be signaled as well. Will self-clear if data is read (FIFO and non-FIFO).	See also FIFO status in DATACRTL if FIFO is available
12	RO	TXNOTFULL	Is 1 when the To-bus buffer/FIFO can accept more data to go out. **Default**: 1. For all but External FIFO, this uses DATACTRL TXTRIG, which defaults to not-full. If DMA is enabled for TX, it will also be signaled to provide more data.	
13	R/W1C	DACHG	The Slave Dynamic Address has been assigned, re-assigned, or reset (lost) and is now in that state of being valid or none. Actual DA can be seen in the DYNADDR register. Note that this will also be used when MAP Auto feature is configured. This will be changing one or more MAP items. See DYNADDR and/or MAPCTRLn. DYNAADDR for the main DA (0) will indicate if last change was due to Auto-MAP.	
14	R/W1C	CCC	A Common-Command-Code (CCC), not handled by block, has been received. This acts differently between: ▪ Broadcasted ones, which will then also correspond with RXPEND and the 1st byte will be the CCC (command). ▪ Direct ones, which may never be directed to this device. If it is, then the TXSEND or RXPEND will be triggered with this and the RXPEND will contain the command.	For CCCs that the block is not enabled to process.
15	RO	ERRWARN	An error or warning has occurred, such as data underun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the ERRWARN register for details of the cause. Write to the ERRWARN register to clear.	Some errors only possible if configured features.
16	R/W1C	HDRMATCH	An HDR command matched this device's I3C Dynamic Address. The Command will be available as the 1st byte and RXPEND will be set, whether read or write. The MSb of that command byte also indicates if a read or a write. If a read, and there are to-bus bytes waiting, the command will be ACKed and the data sent back, else it will be NACKed. Note that when this is set, the ERRWARN bit should be checked, as the HPAR error may have be encountered after signaling this (the parity is after the destination address and CMD).	Only if HDR is supported.

<table>
<tr><td>17</td><td>R/W1C</td><td>CHANDLED</td><td>A Common-Command-Code (CCC) is being handled by the block. This is a notification only. The result may be an updated register.</td><td>For CCCs that the block is enabled to process.</td></tr>
<tr><td>18</td><td>R/W1C</td><td>EVENT</td><td>Slave: Pending IBI, MR, or Hot-Join has been sent as requested. See upper status for details.
Note that for IBI, this occurs on the ACK if no IBI byte, and after the 1 IBIDATA byte has been sent. So, if time control is used, those will go out after this EVENT is signaled.</td><td>Only if configured to support events.</td></tr>
<tr><td>19</td><td>R/W1C</td><td>SLVRST</td><td>Slave Reset to Peripheral (not whole chip). This helps application do follow up such as re-initialization.
Note : For Master MSTATUS, this is NOWMASTER bit.</td><td>If Slave Reset and RSTACT are supported</td></tr>
<tr><td>21:20</td><td>RO</td><td>EVDET</td><td>Current details of last (EVENT=1) or pending event:<table><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>Request not sent yet. Either no START yet, or waiting for Bus-Available or Bus-Idle (HJ).</td></tr><tr><td>2</td><td>Request sent and NACKed – will try again</td></tr><tr><td>3</td><td>Request sent and ACKed, so Done (unless time control data being sent still).</td></tr></table></td><td>Only if configured to support events.</td></tr>
<tr><td colspan="5">CCC States as known by the block (depending on what is enabled)</td></tr>
<tr><td>24</td><td>RO</td><td>IBIDIS</td><td>Is 1 if IBIs are disabled at this time. Note that CTRL requests will be held off while disabled.</td><td>Only if enabled for block to process.</td></tr>
<tr><td>25</td><td>RO</td><td>MRDIS</td><td>Is 1 if Master Requests are Disabled at this time. Note that CTRL requests will be held off while disabled.</td><td>Only if enabled for block to process.</td></tr>
<tr><td>26</td><td>RO</td><td>reserved</td><td>reserved</td><td>Only if enabled for block to process.</td></tr>
<tr><td>27</td><td>RO</td><td>HJDIS</td><td>Is 1 if Hot-Join is disabled at this time. Note that CTRL requests will be held off while disabled.</td><td>Only if enabled for block to process.</td></tr>
<tr><td>29:28</td><td>RO</td><td>ACTSTATE</td><td>Activity state from CCC as:<table><tr><th>Value</th><th>Meaning</th></tr></table></td><td>Only if enabled for</td></tr>
</table>

			0 – No Latency – normal bus operations 1 – 1ms of latency 2 – 100ms of latency 3 – 10s of latency	block to process.
31:30	RO	TIMECTRL	Indicates if time control is currently enabled: **Value** – **Meaning** 0 – No time control is enabled 1 – Synchronous is enabled – if supported 2 – Asynchronous standard mode (0 or 1) is enabled – if supported 3 – Both are enabled – if supported.	If Time control is enabled

3.3 CTRL @ 0x00C – I3C Control register

The Control register is used to activate various special operations for the Slave, but only if the block is configured to support. This includes events such as IBI, as well as GETSTATUS fields (except the Protocol error, which is automatically set).

Bit(s)	Type	Name	Description	Config
1:0	RW	EVENT	If set to non-0, will request an event. Once requested, STATUS.EVENT and EVDET will show the status as it progresses. Once completed, the field will automatically return to 0. Once non-0, only 0 can be written (to cancel) until done. **Value** – **Meaning** 0 – Normal mode. If set to 0 after was a non-0 value, will cancel if not already in flight. 1 – Start an IBI. This will try to push through an IBI on the bus. If data associated with the IBI, it will be drawn from the IBIDATA field. Note that if Time control is enabled, this will include any time control related bytes; further, the IBIDATA byte will have bit 7 set to 1 automatically (as is required for time control). The interrupt will occur after the 1st (Mandatory) IBIDATA, if any. 2 – Start Master-Request or Peer-to-Peer request: the meaning depending on BCR register configured in the block.	If block is configured for one or more of these. Time control is separately configured.

			3 — Start Hot-Join request. This should only be used when device is powered on after bus is already up, or is connected by hot insertion. Will wait for Bus Idle. This must be set before the Enable in the CONFIG.	
3	RW	EXTDATA	If 1, then after IBIDATA is emitted, extended data will be pulled from TXFIFO or IBIFIFO until END; this treats the extended data as an I3C Read, and an underrun error is reported if happens. This may be used with timecontrol, in which case the data follows the time info.	If block is configured for IBI data.
7:4	RW	MAPIDX	Index of Dynamic Address that IBI is for. This is 0 for the main or base Dynamic Address, or can be any valid index. See DYNADDR register for more details.	Only if Mapping is enabled
15:8	RW	IBIDATA	Data byte to go with an IBI, if enabled for it. If enabled (was in BCR), then it is required.	If block is configured for IBI data.
19:16	RW	PENDINT	Should be set to the pending interrupt that GETSTATUS CCC will return. This should be maintained by the Application if used and configured, as the Master will read this. If not configured, the GETSTATUS field will return 1 if an IBI is pending, and 0 otherwise.	Controlled with the CCC handling parameter.
21:20	RW	ACTSTATE	Should be set to the Slave's activity state that GETSTATUS CCC will return as Activity Mode. This should be maintained by the Application if used and configured, as the Master will read this. If not configured, the GETSTATUS field will always return a 0.	
31:24	RW	VENDINFO	Should be set to the Vendor Reserved field that GETSTATUS CCC will return. This should be maintained by the Application if used and configured, as the Master will read this. If not configured, the GETSTATUS field will always return a 0.	Controlled with the CCC handling parameter.

3.4 INTSET, INTCLR, INTMASKED @ 0x010, 0x014, 0x18 – Interrupt enable control registers (if configured for Interrupts)

The Interrupt registers, if enabled in the block, allow masking interrupt sources as well as checking which have activated. The normal method is to Enable an interrupt and then once it fires, it is either cleared by writing the STATUS register or cleared by action on the corresponding data register. The Interrupt is level held, meaning it stays set until the cause is cleared one way or the other. The block prevents races so that if a new event comes in, it will not be lost.

The table below shows all 3 registers, with the Type indicating INTSET (R/W1S), INTCLR (WO), and INTMASKED (RO, is STATUS&INTSET).

INTSET allows setting enables for interrupts (connecting the corresponding STATUS source to causing an IRQ to the processor).
INTCLR allows clearing interrupt enables without read-modify-write problems.

INTMASKED allows checking status bits in terms of which are holding the IRQ, if any.

Bit(s)	Type	Name	Description	Config
8	R/W1S, W1C, and RO	START	Interrupt on START and repeated START when needed (such as wakeup). See also STOP.	
9	R/W1S, W1C, and RO	MATCHED	Interrupt on Matching header for I3C Dynamic Address. Also for matching header for I2C Static Address, if configured and if no Dynamic Address set (see DYNADDR register).	Static Addr match only if configured
10	R/W1S, W1C, and RO	STOP	Interrupt on STOP state on the bus. See START as the preferred interrupt when needed. This interrupt may not trigger for quick STOP/START combination, as it relates to the state of being stopped.	
11	R/W1S, W1C, and RO	RXPEND	Interrupt when Receiving a message from Master, which is not being handled by block (excludes CCCs being handled automatically). If FIFO, then RX fullness trigger. If DMA, then message end. See also REQ in STATUS for context.	FIFO and DMA use only if configured
12	R/W1S, W1C, and RO	TXSEND	Interrupt when Request data by Master (Read). This interrupts on 1st request (header) as well as when ready for more; the Application indicates if more or END. If FIFO, triggers on TX emptiness trigger. If DMA, then message end (DMA end or terminate).	
13	R/W1S, W1C, and RO	DACHG	Interrupt on Dynamic address defined (SETDASA or ENTDAA) or lost (RSTDAA). See also DADDR register. This will not interrupt on SETNEWDA.	
14	R/W1S, W1C, and RO	CCC	For CCCs not handled by the block, RXPEND will also interrupt and the STATUS REQ field will indicate it is a CCC. Note that the handling of broadcast vs. direct write vs. direct read are all subtly different. The direct read ones likely do not provide enough time to respond, but the i3c spec allows a single retry, buying more time.	For CCCs that the block is not enabled to process.
15	R/W1S, W1C, and RO	ERRWARN	Interrupt when an error or warning has occurred, such as data 17nderrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the ERRWARN register for details of the cause.	Some errors only possible if

				configured features.
16	R/W1S, W1C, and RO	DDRMATCHED	Interrupt when DDR matched for Read or Write command.	Only if HDR enabled
17	R/W1S, W1C, and RO	CHANDLED	Interrupt when a Command-Command-Code was received and handled by the block (is done). STATUS will show new results. This can be used to track when Activity states and masks on events (e.g. IBIs) occur.	For CCCs that the block is enabled to process.
18	R/W1S, W1C, and RO	EVENT	Slave: Interrupt when Pending IBI, MR, or Hot-Join has been sent as requested. See STATUS for details (EVDET).	Only if configured to support events.
19	R/W1S, W1C, and RO	SLVRST	Slave: SlaveReset pattern detected and action was set to reset peripheral. Application should reset peripheral or verify it is correct. This should not be unmasked.	Only if SLVRST is configured

3.5 ERRWARN @ 0x01C – I3C Error and Waning register

The ERRWARN register contains I3C protocol errors and issues detected on the line. This includes internal issues such as overrun and underrun, as well as detected errors and condition such as parity errors, CRC errors, and termination of a Read by the Master.

Bit(s)	Type	Name	Description
General Errors and Warnings detected by slave engine			
0	R/W1C	ORUN	The internal from-bus buffer/FIFO was overrun (too many chars coming in and not drained by the app fast enough).
1	R/W1C	URUN	The internal to-bus buffer/FIFO was underrun during data read (app did not provide the data fast enough). The END bit or register should be used if that was the last one.
2	R/W1C	URUNNACK	The internal to-bus buffer/FIFO was underrun in the read header and so the block NACKed the header.
3	R/W1C	TERM	The Master terminated a read from Slave when an END was not set (on same or previous).
4	R/W1C	INVSTART	Invalid start with SCL falling while SDA=1 in STOP condition.
5	R/W1C	IBIURUN	If EXTDATA is used in CTRL, this will signal if the TX or IBI FIFO underruns before an END mark.
Errors in data integrity			
8	R/W1C	SPAR	SDR Parity error on message from Master. This will also set the GETSTATUS Protocol Error sticky bit (cleared on GETSTATUS read).

			For Read, this will be set if a Read Abort (timeout) occurs due to Master not driving clock for >100µs during an I3C SDR Read.
9	R/W1C	HPAR	HDR Parity error or framing error on message from Master. Note that the corresponding CMD or Data that had the error will normally be in the RX buffer (read via RDATAB).
10	R/W1C	HCRC	HDR-DDR CRC error on message from Master. This calls into question the data from the whole DDR command frame. Note that this includes an HDR Restart or Exit being issued before an HDR-DDR message from Master is complete.
11	R/W1C	S0S1	S0 or S1 error has occurred and the Slave is locked waiting on an HDR Exit Pattern. Writing 1 to this will cause it to release the lock, but that should be used with great care. Normally, it will clear automatically when an Exit Pattern is detected. So, writing 1 should only be used under controlled circumstances to avoid problems. It will then wait for a START (or repeated START) or STOP to operate normally.
Application errors (misuse)			
16	R/W1C	OREAD	The RDATAB register was read for more bytes than were available by app.
17	R/W1C	OWRITE	The WDATAB/BE register was written when FULL

3.6 DMACTRL @ 0x020 – DMA Control register (if configured for DMA)

The DMA Control register allows for DMA to be used for inbound messages and outbound messages. This is only available if configured for it.

DMA is limited in value for Slave use. This is because the Slave has to be reactive to what happens. The two common use models:

1. From-bus collection to avoid being overrun. The CONFIG MATCHSS bit is set, and then the processor enables the interrupts for START, and STOP, as well as enabling the DMA to collect the data. The START or STOP interrupt will only occur after a message directed to the Slave (MATCHED bit set), and the DMA copied data can then be examined.
2. To-bus for larger reads. Since I3C and I2C reads are preceded by a write which indicates what will be read (or in response to an IBI from the Slave), the DMA can be used to push through the data.
   Note that for I3C, the last value needs to be handled by the processor unless the DMA moves wider words to be able to set the END bit (i.e. 16-bit values when byte mode or 32-bit values when in half-word)
   Note that for I2C, the last value is determined by the Master, so the DMA may end early or may run out and the Master still wants more.

Bit(s)	Type	Name	Description	Config
1:0	RW	DMAFB	DMA Read (From-bus) trigger. If enabled with 1 or 2, will request DMA on RX trigger (see DATACTL). It will request until empty unless DMA setup as trigger. Note: will cancel on ERRORWARN (since cannot be as planned). **Value** | **Meaning**	If configured for DMA and not external FIFO.

			0 DMA not used 1 DMA enabled for 1 Frame – auto clears on STOP or repeated START (see MATCHSS in CONFIG). 2 DMA enabled until turned off.	
3:2	RW	DMATB	DMA Write (To-bus) trigger. If enabled with 1 or 2, will start request DMA on TX trigger (see DATACTL). It will request until full unless DMA setup as trigger. Note: will cancel on ERRORWARN (since cannot be as planned). **Value** / **Meaning** 0 DMA not used 1 DMA enabled for 1 Frame (ended by DMA or Terminated) – then auto-clears on STOP or START (see MATCHSS in CONFIG). 2 DMA enabled until turned off. Normally should only be used with Master Message mode.	
5:4	RW	DMAWIDTH	Width of DMA operations, if configured to allow half-word data access. **Value** / **Meaning** 0, 1 Byte. **Default**=1 2 Half word (16b). If supported by HW. This will make sure 2 bytes free/available in FIFO. 3 reserved	

3.7 DATACTRL @ 0x02C – Data control register (and FIFO if configured for FIFO)

The Data control register simply assists in data control when no FIFO, and assists in FIFO when the FIFO is available (regardless of size). This allows some control over the FIFO behavior. In particular, it allows control of when to interrupt on fullness/emptiness and also controls behavior related to width, when not 1 byte wide.

Bit(s)	Type	Name	Description	Config
0	WO	FLUSHTB	Flush the to-bus buffer/FIFO. Used when Master terminates a to-bus (read) message prematurely.	
1	WO	FLUSHFB	Flushes the from-bus buffer/FIFO. Not normally used.	

<table>
<tr><td>3</td><td>WO</td><td>UNLOCK</td><td>If this bit is not written 1, the register bits from 7 to 4 are not changed on write.</td><td></td></tr>
<tr><td>5:4</td><td>RW</td><td>TXTRIG</td><td>Trigger level for TX emptiness when FIFOed. Affects interrupt and DMA (if enabled). The defaults is 3<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Trigger on empty</td></tr><tr><td>1</td><td>Trigger on ¼ full or less</td></tr><tr><td>2</td><td>Trigger on ½ full or less</td></tr><tr><td>3</td><td>Trigger on 1 less than full or less. Default</td></tr></table></td><td>If Internal FIFO enabled for to-bus, has full meaning. If just 2 or 3 byte (ping pong) buffer, then only 0 is special to mean “empty”. That is, 0 will cause TXNOTFULL to be true only if TX FIFO is empty, so Half word write can be used; 1, 2, and 3 will set TXNOTFULL when not full.</td></tr>
<tr><td>7:6</td><td>RW</td><td>RXTRIG</td><td>Trigger level for RX fullness when FIFOed. Affects interrupt and DMA (if enabled).<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Trigger on not empty</td></tr><tr><td>1</td><td>Trigger ¼ or more full</td></tr><tr><td>2</td><td>Trigger ½ or more full</td></tr><tr><td>3</td><td>Trigger ¾ or more full</td></tr></table></td><td>If Internal FIFO available for from-bus, has full meaning. If just 2 or 3 byte (ping pong) buffer, then only 3 is special to mean at least 2 bytes in). That is, 3 will set RXPEND only if 2 or 3 bytes are in RX FIFO, so Half word read can be used; 0, 1, and 2 will set RXPEND if not empty.</td></tr>
<tr><td>20:16</td><td>RO</td><td>TXCOUNT</td><td colspan="2">Count of bytes in TX</td></tr>
<tr><td>28:24</td><td>RO</td><td>RXCOUNT</td><td colspan="2">Count of bytes in RX</td></tr>
<tr><td>30</td><td>RO</td><td>TXFULL</td><td colspan="2">Is 1 if TX is full</td></tr>
<tr><td>31</td><td>RO</td><td>RXEMPTY</td><td colspan="2">Is 1 if RX is empty. Default 1.</td></tr>
</table>

3.8 WDATAB @ 0x030 – Write Byte Data (to-bus) register

This register allows writing a byte to the bus (to Master) unless external FIFO is used. This takes a byte and an end-of-data (last) marker bit.
A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the STATUS register.

Bit(s)	Type	Name	Description	Config
7:0	WO	DATA	Byte to send to master	If not external FIFO
8, 16	WO	END	If 1, this marks the last byte of the message. If 0, it is assumed there are more bytes. This is required to be used in I3C and is	

			optional in i2c. Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) since DDR is byte-pairs.	

3.9 WDATABE @ 0x034 – Write Byte Data as End (to-bus) register

Write a byte just like WDATAB, but mark as end-of-data (last byte). Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) since DDR is byte-pairs.

A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the STATUS register.

Bit(s)	Type	Name	Description	Config
7:0	WO	DATA	Byte to send to master	If not external FIFO

3.10 MWDATAH @ 0x038 – Write Half-word Data (to-bus) register

This register allows writing a half-word (pair of bytes) to the bus unless external FIFO is used. This takes a half-word, which will send out the Low byte and then the High byte. An end-of-data (last) marker bit is allowed (or must be 0).
A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the DATACTRL register.

Bit(s)	Type	Name	Description	Config
7:0	WO	DATA0	1st Byte to send to Master	If not external FIFO and if configured for half-word data
15:8	WO	DATA1	2nd Byte to send to Master	
16	WO	END	If 1, this marks the last byte of the message. If 0, it is assumed there are more bytes/half-words. For this register, this always marks DATA1 as the end. This is required to be used in I3C and is optional in i2c. Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) since DDR is byte-pairs.	

3.11 MWDATAHE @ 0x03C – Write Half-word Data as End (to-bus) register

Write a half-word (byte pair) just like MWDATAH, but mark the 2nd bytes as end-of-data (last byte). Note that for HDR-DDR, the byte with the END must be an even (2nd, 4th, 6th, etc) since DDR is byte-pairs.

A half-word should not be written unless there is room for both, as indicated by use of TX FIFO level trigger or TXCOUNT available space in the DATACTRL register.

Bit(s)	Type	Name	Description	Config
7:0	WO	DATA0	1st Byte to send to Master	If not external FIFO and if configured for half-word data
15:8	WO	DATA1	2nd Byte to send to Master	

3.12 RDATAB @ 0x040 – Read Byte Data (from-bus) register

This register allows reading a byte from the bus (Master) unless external FIFO is used.

A byte should not be read unless there is data waiting, as indicated by the RXPEND bit being set in the STATUS register.

Bit(s)	Type	Name	Description	Config
7:0	RO	DATA0	Byte read from Master	If not external FIFO

3.13 RDATAH @ 0x048 – Read Half-word Data (from-bus) register

This register allows reading a Half-word (byte pair) from the bus (Master) unless external FIFO is used.

A Half-word should not be read unless there is at least 2 bytes of data waiting, as indicated the RX FIFO level trigger or RXCOUNT available space in the DATACTRL register.

Bit(s)	Type	Name	Description	Config
7:0	RO	LSB	1st byte read from Master	If not external FIFO and if configured for half-word data
15:8	RO	MSB	2nd byte read from Master	

3.1 WIBIDATA @ 0x050 – Write IBI Extended Byte Data register

Write a byte just like WDATAB, but writes to IBI EXTDATA (see CTRL reg) if configured and enabled.

Reads back to indicate if can take another byte.

Write END to mark END of IBI data (or will get an IBI Underrun error if FIFO goes empty)

Write CLEAR to clear the FIFO.

A byte should not be written unless there is room, as indicated by reading this register.

Any errors during IBI data processing, including terminate-by-master and under-run use the normal ERRWARN. Underrun uses the IBIURUN, terminate by master uses the normal terminate by master.

Bit(s)	Type	Name	Description	Config
0	RO	FREE	If reads 1, the DATA buffer can take another byte, and may be written	If configured for IBI EXT DATA and IBI FIFO
7:0	WO	DATA	Byte to send to master	
8, 16	WO	END	If 1, this marks the last byte of the message. If 0, it is assumed there are more bytes.	
31	WO	CLEAR	If 1, this clears the FIFO and the lower bits are ignored.	

3.1 WDATAB1 @ 0x054 – Byte-only Write Byte Data (to-bus) register

This register allows writing a byte to the bus (to Master) such that only bits 7:0 are looked at (no END bits). This is intended to be used by DMAs which do not format the upper part of the APB word.

A byte should not be written unless there is room, as indicated by the TXNOTFULL bit being set in the STATUS register – for DMA when dma_tb_req is asserted.

Bit(s)	Type	Name	Description	Config

7:0	WO	DATA	Byte to send to master	If not external FIFO
31:8	-	-	Ignored	

3.2 CAPABILITIES(2) @ 0x05C 0x060 – Capabilities info register

The Capabilities read-only registers indicate what is configured in the block, so that SW can be sure it is matched. CAPABILITIES is at 0x60 and CAPABILITIES2 is at 0x5C.

CAPABILITIES at 0x060			
Bit(s)	**Type**	**Name**	**Description**
1:0	RO	IDENA	Indicates how the ID 48b value is to be handled. • 0 = application (VID and PARTNO registers) • 1 = HW • 2 = HW but instance provided • 3 = PARTNO register used
5:2	RO	IDREG	Bits indicate what is in Regs vs. in HW: • Bit 0 = 1: ID Instance is a register. Only if no PARTNO reg • Bit 1 = 1: IDRAND field is available • Bit 2 = 1: DCR register is available • Bit 3 = 1: BCR register is available

8:6	RO	HDRSUPP	Indicates which HDR is supported:
			• Bit 0 = 1: DDR
			• Bit 2 and 1 = 0 if no BT, else
			  ○ 1 = BT 1 lane only
			  ○ 2 = BT 1 and 2 lane
			  ○ 3 = BT 1, 2, and 4 lane

9	RO	MASTER	1 if Master capability included.
11:10	RO	SADDR	Indicates how static address is handled: • 0 = no static address • 1 = static address fixed in HW • 2 = HW controls dynamically (e.g. from pin strap) • 3 = CONFIG register supplies the static address
15:12	RO	CCCHANDLE	Indicates who handles CCCs between block and app: • Bit 0 = 1: Block handles events, activities, status, HDR, and if enabled for it, ID and static address related • Bit 1 = 1: Block handles max read and write length as well as max data speed.

			• Bit 2 = 1: GETSTATUS CCC will return the CTRL register's PENDINT and ACTSTATE fields. • Bit 3 = 1: GETSTATUS CCC will return the CTRL register's VENDINFO (vendor info) bits.
20:16	RO	IBI_MR_HJ	Indicates which events are to be allowed: • Bit 0 = 1: Supports application generating an IBI • Bit 1 = 1: when Bit 0=1, means IBI has data from register • Bit 2 = 1: Supports application generating Master request for 2ndary master or Peer to Peer. • Bit 3 = 1: Supports application generating Hot Join • Bit 4 = 1: Use BAMATCH register for Bus Available timing.
21	RO	TIMECTRL	If 1, indicates that at least one time control type is supported.
22	-	-	reserved
25:23	RO	EXTFIFO	Indicates External FIFOs is enabled. If not, check FIFOTX and FIFORX for internal FIFO. • 0 = no external FIFO. • 1 = standard available/free external FIFO • 2 = request track external FIFO • 3= reserved
27:26	RO	FIFOTX	Indicates if TX (to-bus) is enabled and what size it is: • 0 = default 2 byte TX FIFO. • 1 = 4 byte TX FIFO • 2 = 8 byte TX FIFO • 3 = 16 byte TX FIFO
29:28	RO	FIFORX	Indicates if RX (from-bus) is enabled and what size it is: • 0 = default 2 (or 3) byte RX FIFO. • 1 = 4 byte RX FIFO • 2 = 8 byte RX FIFO • 3 = 16 byte RX FIFO
30	RO	INT	1 if Interrupts supported
31	RO	DMA	1 if DMA supported

CAPABILITIES2 at 0x05C			
Bit(s)	**Type**	**Name**	**Description**
3:0	RO	MAPCNT	Is 0 if no mapping (DYNADDR used to add more Static and/or Dynamic addresses to act as virtual slaves). Else, indicates how many are allowed.

4	RO	I2C10B	Is 1 if I2C 10-bit address is allowed in MAP 1 (so MAPCNT must be 1 or greater).
5	RO	I2CRST	Is 1 if I2C SW Reset is supported
6	RO	I2CDEVID	Is 1 if I2C Device ID is supported
7	-	-	reserved
8	RO	IBIEXT	Supports CTRL.EXTDATA to allow data past MDB
9	RO	IBIFIFO	Supports 2 entry FIFO for IBI EXTDATA using WIBIDATA register
15:10	-	-	reserved
16	RO	V1_1	Is 1 if Supports I3C v1.1 GETCAPS
17	RO	SLVRST	Is 1 if Supports v1.1 SlaveReset
19:18	RO	GROUP	Is 0 if does not supports v1.1 Group addressing, else indicates number of groups as 1, 2, 3. Groups use mapping, so MAPCNT must be same or greater.
20	-	-	reserved
21	RO	AASA	Supports SETAASA
22	RO	SSTSUB	Slave-Slave(s)-Tunnel subscriber capable
23	RO	SSTWR	Slave-Slave(s)-Tunnel write capable
31:24	-	-	reserved

3.3 DYNADDR @ 0x064 – Dynamic Address register (and Map control)

The Dynamic Address register is filled in with the assigned address once the Master has assigned it via SETDASA or ENTDAA CCC commands. It will clear if the RESETDAA CCC is used. The current validity state is also indicated via the STATUS.DACHG bit, which can be used to interrupt the processor. The DACAUSE field can be used to determine why it changed, if enabled.

NOTE: Uses for Mapped addresses, if enabled, are preferably now accessed via the MAPCTRLn registers.

DA [0] may be written whether mapping is enabled or not: If configured to allow write, this is normally only used to restore the DA after a power-down (ultra low power state that loses power to peripheral but retains the DA somewhere else). This is not needed if state-retention flops are used for the DA. This mechanism only allows write when slave is disabled (will be ignored otherwise); note that the KEY is used to permit the write. If the Master uses RSTDAA or SETNEWDA, it will over-ride this mechanism and cede to the Master assigned DA. Note that when enabling the slave, the CONFIG.OFFLINE bit should be set as well. This will wait for evidence that the bus is not in I3C HDR mode – will exit when an HDR Exit pattern is see or when 60us has expired. This makes it safe to monitor START and STOP. If the App needs to do an IBI, it should either wait for a STOP (see STATUS) or make sure that 200 micro-seconds have gone by with no activity (no START or STOP) before emitting the IBI.

Mapping: The MAPIDX/MAPSA model allows writing additional DAs (and SAs) into a list – number in list is pre-configured; any with DAVALID=0 are never matched. The additional DAs may be based on the SETBRGTGT CCC, or done by move (read current DA and then copy into upper map, then invalidate the 0

map) when matching ENTDAA over and over. Any mapped location can be invalidated by writing the MAPIDX and DAVALID=0; note that the KEY must be set to permit this. The mapped ones are not reset by the RSTDA CCC. See also the MSGMAPADDR register.

Note that to copy the base DA to the mapped set, the configuration has to be setup to allow it, else they are distinct mechanisms. If used for the copy model, then a special reset feature is allowed. In this case, the DYNADDR register is written with KEY=0xCB19 and the rest 0. This will clear the DA and then self-clear. Note that if in I3C mode, a base DA is required, so the last one should not be cleared or writing one with DAVALID=1 is necessary.

NACK/ACK of Mapped DAs/SAs: the register also permits specific DAs (above the base) to NACK or ACK writes or reads, such as for bridges when the endpoint is busy. The NACK should not be set for too long, or the Master may think the slave is in an error state. The model is to write with KEY of 0xA731 and the MAPIDX and DAVALID to select ACK or NACK.

Mapping: if write with MAPIDX!=0 and KEY=0, then next Read will return details on the Static or Dynamic address at that location. If the read does not have MAPIDX !=0, then was not an acceptable location.

Bit(s)	Type	Name	Description	Config
0	RO or RW	DAVALID	Is 1 if Dynamic Address is Assigned when reading or setting DADDR. Is 1 for ACK, 0 for NACK when using the 0xA731 KEY.	Writable only if enabled for modify, likewise for DCAUSE. MAP scheme is only if configured for MAPped addresses.
7:1	RO or RW	DADDR	Assigned Dynamic Address when DAVALID is 1. Will be a static address if MAPSA is 1.	
10:8	RO (for MAPIDX=0)	DCAUSE	When just reading the main DA (e.g. after DACHG), this indicates the cause: 0 = No info. Is 0 if not configured to write DA. 1 = Set using ENTDAA 2 = Set using SETDASA or SETAASA or SETNEWDA 3 = Cleared using RSTDAA 4 = Auto MAP change happened last	
11:8	WO/RO	MAPIDX	Selects which mapped DA to write to, with 0 meaning not mapped (normal use of write DYNADDR if allowed). This allows for a list of matching DAs or SAs (i2c static addresses). When written with non-0 and KEY, this selects the slot to write into with a DA or SA or to control the ACK/NACK behavior. When written with non-0 and KEY=0, read back returns Map DADDR, if valid, MAPSA, SA10B, and [16]=1 if NACK. This field will indicate the index.	
12	WO	MAPSA	If 1 on write with MAPIDX!=0, sets a static address into list, else a dynamic address.	

15:13	WO	SA10B	If not 0 when MAPSA is set, will store a 10bit static address composed of {SA10B,DADDR}. Only one static address may be stored this way, and it must be MAPIDX==1.	
31:16	WO and RO	KEY	Must set to 0xA4D9 to write DADDR (and set DAVALID to 1). Base is only writable when slave is not enabled (for restoral) unless mapping is enabled. The mapped locations and base may be written when slave is enabled, but care should be taken to not be when there are transactions on the I3C bus. Reads back as 1 if overwritten, else is 0 if from Master assigned, including when master changes it. If Mapping is allowed, then writing with key of 0xCB19 is used to clear the base DA. If Mapping is allowed, then writing with key of 0xA731 means that MAPIDX will indicate which mapped address to set to ACK or NACK, such that DAVALID =1 if ACK, =0 if NACK; this allows having the address refuse write/read requests when NACK. When using read back from map index (write with MAPIDX!=0, KEY=0, read) bit [16] =1 if NACK, else 0.	

3.4 MAXLIMITS @ 0x068 – Maximum limits set by I3C Master (if configured for Limits)

The Max limits may or may not be enabled in the HW, including max read and write length. If they are, the current setting (including default request) shows up in this register.

Bit(s)	Type	Name	Description
11:0	RW	MAXRD	Maximum read length. Must be between 16 and saturation at 4095 (means is set to that or higher). App should not set higher than Master sets, only smaller. **Default**: non-0 if set by parameters in HW config.
27:16	RW	MAXWR	Maximum write length. Must be between 8 and saturation at 4095 (means is set to that or higher). App should not set higher than Master sets, only smaller. **Default**: non-0 if set by parameters in HW config.

3.5 PARTNO @ 0x06C – Register to allow Application to set I3C Part number ID (if configured for Part-number from app)

The Part number is normally hard coded into the hardware, but if the application needs to set it, this allows that. It is only available if configured for a register based part number. Note that part number includes instance as well. The application must write a value into this field since 0 is not normally valid.

Bit(s)	Type	Name	Description
31:0	RW	PARTNO	May be set to the part number.

3.6 IDEXT @ 0x070 – Register to allow Application to set I3C ID components (if configured for ID parts from app)

The ID extension fields of Instance (from Part number), DCR, and BCR are normally set in the HW. But. If configured, the Application must set them instead.

Bit(s)	Type	Name	Description
3:0	RW	INSTANCE	Set the Instance of the ID if configured. Would not be used if PARTNO is provided by the application.
6:4	RW	I2CREV	If the I2C DeviceID is used, then this supports the revision. These 3 bits will be ORed onto the constant ID.
15:8	RW	DCR	Set the DCR (device type) register if configured for it.
23:16	RW	BCR	Set the BCR (Bus capabilities) register if configured for it. This controls features like Peer-to-peer or 2ndary master, Slow speed requirements, etc.

3.7 VENDORID @ 0x074 – Register to allow Application to set I3C Vendor ID (if configured for VID from app)

The MIPI Vendor ID is normally hard coded into the hardware, but if the application needs to set it, this allows that. It is only available if configured for a register based VID. The default will be set from the constant field and so usually will be the chip vendor. If using the chip vendor ID, the PARTNO must then not collide with other uses.
MIPI Vendor ID is available to all companies, MIPI member or not; it only takes a request via the mipi.org website.

Bit(s)	Type	Name	Description	Config
14:0	RW	VID	Should be set to the 15-bit MIPI Vendor ID of developer of the firmware on this part. **Default** will be chip vendor's ID if supplied.	Only if provided, else is hard coded to ID of chip vendor.

3.8 TCCLOCK @ 0x078 – Register to allow Application to indicate Frequency and accuracy of time-control clock (if enabled)

The Clock frequency and accuracy is normally a constant set by the HW. But, if the clock can be adjusted (e.g. divided) or the accuracy could vary with knowable information, then it may be set via the register. This should be updated whenever the clock source is changed.

Bit(s)	Type	Name	Description	Config
7:0	RW	ACCURACY	Clock accuracy in 1/10ths of %. So, 1.5% is 15.	If time control used and if Reg form is selected.

			Default: set by parameters if configured.	
15:8	RW	FREQ	Clock frequency in 0.5MHz steps (e.g. 10MHz is 20). **Default**: set by parameters if configured.	If time control used and if Reg form is selected.

3.9 MSGMAPADDR @ 0x07C – Indicates matching DA or SA from last messages

When the DYNADDR register is used to build a list of extra DAs or SAs to match, this register allows the software to determine which address was matched when the STATUS.MATCHED bit is set. It holds the last 3 matches.

Bit(s)	Type	Name	Description	Config
3:0	RO	MAPLAST	Matched address index for current or last matched message. 0 for the base address.	If mapped address list is enabled.
4	RO	LASTSTATIC	1 if last matched address was an i2c Static address vs. an I3C Dynamic Address	
11:8	RO	MAPLASTM1	Previous match index. 0 for the base address.	
19:16	RO	MAPLASTM2	2 Previous match index. 0 for the base address.	

3.10 RSTACTTIME @ 0x100 – Set timing rules for SlaveReset recovery

When the Master issues a SlaveReset, some part of the device may be reset, whether peripheral, whole chip, or some subset of the chip. The Master needs to know how long the device will take to recover and be ready for new I3C messages. This may be hard-coded or may be settable using this register.

Bit(s)	Type	Name	Description	Config
7:0	RW	PERRSTTIM	Time to recover, in milliseconds from the I3C peripheral being reset.	If enabled for MMR for this info
15:8	RW	SYSRSTTIM	Time to recover, in seconds, from a full system/chip reset.	
23:16	RW	CUSRSTTIM	Time to recover, in milli-seconds, from a custom Reset, if enabled. This will be a custom RSTACT type.	

3.11 VGPIO @ 0x104 – Control VGPIO mechanism

This register supports a VGPIO mechanism, which is outside of the I3C spec. It uses one of two methods: a 2nd Dynamic Address to support such messages as direct private, or a Vendor CCC. This register selects which to use, and if the Vendor CCC, it sets the CCC address to use.

Bit(s)	Type	Name	Description	Config
0	RW	VCCCSEL	If 1, selects the CCC model for VGPIO, else the DA model is used (which requires DYNADDR register to manage the DA).	If enabled for VGPIO

15:8	RW	VCCCVAL	Value of CCC to use. Default is 0xE4 (1st open)	

3.12 HDRCMD @ 0x108 – Hold HDR Cmd byte vs. into RXFIFO

This register is only used if the feature is configured and if enabled in CONFIG.HDRCMD. If so, this register will hold the last Cmd byte when an HDR message is used (read or write). The NEWCMD bit will be set each time a new Command (will coincide with the DDR interrupt/STATUS). It will auto-clear on read of this register. This register will set the OVFLW bit if a previous value was overwritten without being read by the application. It too will auto-clear once the register has been read.

Bit(s)	Type	Name	Description	Config
7:0	RO	CMD0	Will contain the last Cmd byte when enabled and used. Note that for DDR, the top bit will be 1 if Read, 0 if Write. NOTE: should be read when new to avoid picking up a value in transition.	If enabled at all and if CONFIG.HDRCMD=1
30	RO	OVFLW	Will be 1 if a previous Cmd was overwritten before being read by the application. Will auto clear on next read of the register.	
31	RO	NEWCMD	Will be 1 if a new Cmd. Auto clears on read of the register.	

3.13 CCCMASK @ 0x10C – Mask for Unhandled CCCs

This register allows masking on or off Unhandled CCCs when enabled. Unhandled means not handled by HW.

Bit(s)	Type	Name	Description	Config
0	RW	BASE	If 1, any unhandled CCC in the range of 0 to 0x1F and 0x80 to 0x8F are passed to app, else they are suppressed.	If enabled for CCC masking
1	RW	BASEBX	If 1, any unhandled CCC in the range of 0x20 to 0x48 are passed to app, else they are suppressed.	
2	RW	BASEDX	If 1, any unhandled CCC in range 0x90 to 0xBF are passed to app, else they are suppressed.	
3	RW	MEXTB	If 1, any unhandled CCC in range 0x49 to 0x64 are passed to app, else they are suppressed.	
4	RW	MEXTD	If 1, any unhandled CCC in range 0xC0 to 0xE3 are passed to app, else they are suppressed.	
5	RW	VENDB	If 1, any unhandled CCC in range 0x65 to 0x7F are passed to app, else they are suppressed.	
6	RW	VENDD	If 1, any unhandled CCC in range 0xE4 to 0xFE are passed to app, else they are suppressed.	

3.14 ERRWARNMASK @ 0x110 – Mask for ERRWARN reg bits to form status bit

The ERRWARN register contains I3C protocol errors and issues detected on the line. If any are set, they set the STATYS.ERRWARN bit (and interrupt if enabled). This register allows masking them off so only some form the status bit.

Bit(s)	Type	Name	Description	Config
General Errors and Warnings detected by slave engine				Only if enabled for masking of ERRWARN
0	R/W	ORUN	If 1, will allow this ERRWARN bit to contribute. **Reset to 1.**	
1	R/W	URUN		
2	R/W	URUNNACK		
3	R/W	TERM		
4	R/W	INVSTART		
5	R/W	IBIURUN		
Errors in data integrity				
8	R/W	SPAR	If 1, will allow this ERRWARN bit to contribute. **Reset to 1.**	
9	R/W	HPAR		
10	R/W	HCRC		
11	R/W	S0S1		

3.15 MAPCRTLn @ 0x11C to 0x13C – Control of MAP features, if enabled

The MAPCTRLn registers are named MAPCTRL0, MAPCTRL1, etc based on number of mapped addresses. MAPCTRL0 represents the Primary DA/SA, with MAPCTRL1 onwards being the Mapped addresses.

The features of the control reg depend on configurations. Note that MAPCTRL0 acts somewhat differently, as shown below.

In general, this mechanism is intended to replace the DYNADDR register for all MAP related uses.

Note that when using the Auto-MAP and DASA or AASA, the slot will be changed from Static address to Dynamic. If the Master then issues RSTDAA, the application would need to rewrite the static addresses and enable them, as they will be marked disabled.

Table 1. MAPCTRL0 only

Bit(s)	Type	Name	Description	Config
0	RO	ENA	Is 1 if Primary Dynamic address is enabled	With MAP enabled
7:1	RO	DA	Primary Dynamic address if ENA=1. Note that if ENA=0, then static address would be used (but not shown here).	
10:8	RO	CAUSE	Is how DA was last assigned; used with DACHG STATUS bit/interrupt: 0 = No info. Is 0 if not configured to write DA.	

			1 = Set using ENTDAA 2 = Set using SETDASA or SETAASA or SETNEWDA 3 = Cleared using RSTDAA 4 = Auto MAP change happened last. This may have changed this DA as well (e.g. ENTDAA, SETAASA), but at least one MAP entry auto changed **after**.	
16	RO	OVERRIDE	If 1, the DA has been overridden via the DYNADDR register.	

Table 2. MAPCTRL1 up to MAPCTRL8

Bit(s)	Type	Name	Description	Config
0	RW	ENA	Is 1 if MAP entry is enabled. This could be a static or dynamic address.	With MAP enabled
7:1	RW	ADDR	If ENA=1, is Static or Dynamic Address. See MAPSA to see which	
8	RW	MAPSA	IF ENA=1, is 1 if Static address (i2c style), else is I3C DA.	
11:9	RW	SA10B	If MAPCTRL1, this is SA10B for 10-bit extended address. Is 0, if normal address, else is 10-bit extension.	If enabled for 10bit
12	RW	NACK	If 1, messages to this address will be NACKed every time.	If Enabled
Below only if Auto MAP for DAA and register use (vs. constant). Not related to Auto DASA/AASA.				
13	RW	AUTO	Is 1 if this slot is Enabled for Auto DAA. Will only apply if ENA=0	If Enabled for Auto DAA using MMR.
23:14 Or 31:14	RW	PID	If 1 or more PID bits are to be replaced by register, this provides them. Only uses as many bits as configured for – up to 10 if DCR is used, up to 18 if no DCR.	
31:24	RW	DCR	If DCR is to be replaced by register, this provides the DCR value.	

3.16 ID @ 0xFFC – Optional BlockID

The BlockID, if enabled, allows software to detect the block and its version info.

Bit(s)	Type	Name	Description
31:0	RO	ID	Meaning is specific to each use. **Default**: Parameter sets this value.