Hello, and welcome to this presentation of the I3C peripheral embedded in the STM32H5.
This applies to STM32H56x/573 and STM32H503.

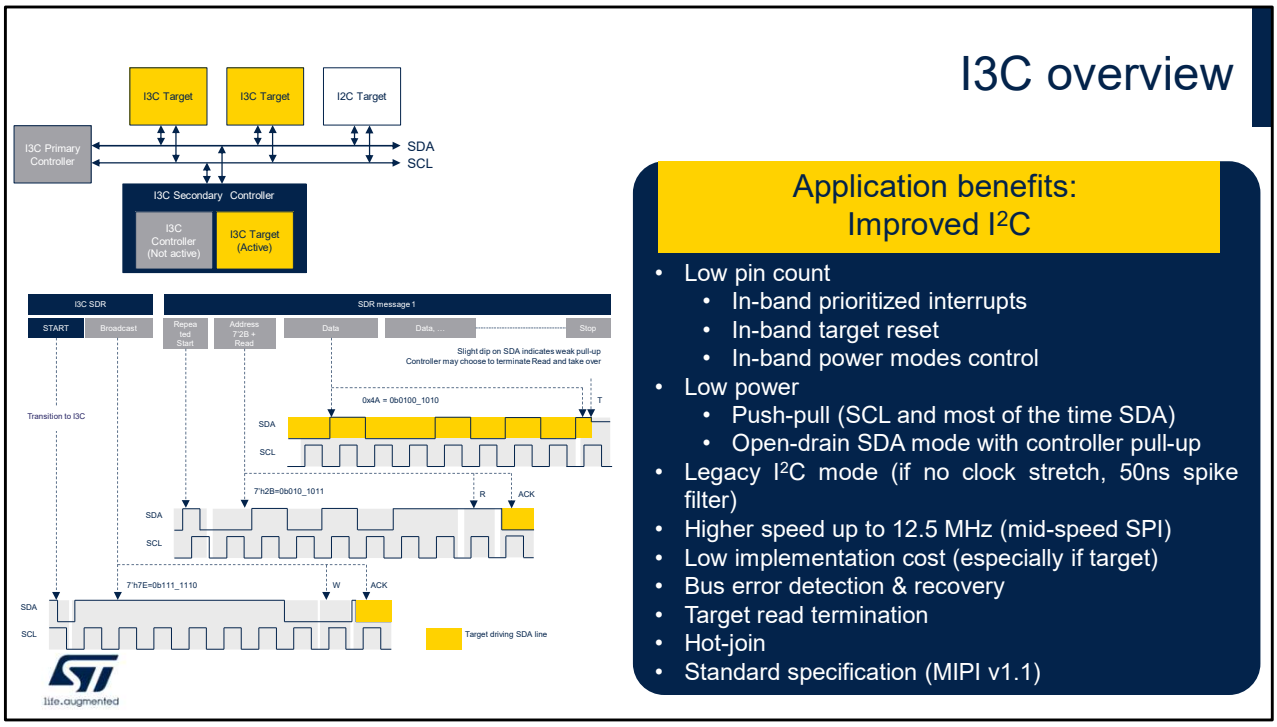
Outline

life.augmented

2

This presentation starts with a quick overview, key features, MIPI specification support, product-dependent features.
It is followed by describing the following items:

- I3C peripheral state and its programming sequence
- I3C MIPI supported Common Command Codes (CCCs)
- A few examples which illustrate I3C bus transfers and registers setting
- A few examples clarifying the I3C FIFOs management
- The list of interrupts and underlying functional events
- Wakeup from stop.



I3C defines the electrical connection between the chips to be a two wire, shared (multidrop), serial data bus, one wire (SCL) being used as a clock to define the sampling times, the other wire (SDA) being used as a data line.

The standard defines a signaling protocol in which multiple chips can control communication and thereby act as the bus controller.

The I3C specification uses the same electrical connections as, and allows some backward compatibility with, the I²C bus.

In the upper figure, a host controller is connected to both I2C and I3C devices. The primary controller is in charge of the I3C bus initialization, while a secondary

master initially acts as a slave, but can also accept mastership from any current master.
Note that an I2C device must use a separate signal to generate an interrupt request to the host controller, while I3C supports a native in-band interrupt mechanism, known as IBI.
The bottom figure represents an I3C typical communication.

- First, the master issues a start and then broadcasts address (=7'h7E) followed by RnW (=0). Then the master turns on a pull-up resistor and goes to open drain while all slaves ack by pulling the SDA line low
- In the second part of the transaction, the master then issues a repeated start, then the address of the slave (=0x2B) it wants to read, followed by RnW=1. The master then turns on a pull-up resistor and goes to open drain, allowing the slave to acknowledge by pulling the SDA line low.
- At this point, the master continues to toggle the SCL line and release the SDA line, allowing the slave to drive SDA to send one byte of data (0x4a) followed by 'T'. T= 1 informs the master that there is additional data, whereas T = 0 signals the end. Here there is additional data, so the slave drives SDA high until SCL goes high, at which time it releases SDA.

In addition to IBI, I3C contributes to reduce the pin count by supporting an in-band reset and in-band power modes control.
The I3C protocol supports a Hot-Join mechanism, to

allow Slaves to join the I3C Bus after it is already configured.
The error detection and recovery methods are provided in order to avoid fatal conditions when errors occur.
The I3C unit present in STM32H5 is compatible with I3C revision 1.1 specification.
Only single data rate (SDR) mode is supported.

STM32H5 I3C peripheral key features 1/2

- MIPI I3C specification v1.1
  - SDR-only primary controller
  - SDR-only secondary controller
  - SDR-only target
- Programmable I3C bus timing
  - When controller
    - SCL high and low time
    - SCL stall time
    - Bus free condition time
  - When target
    - SDA hold time
    - Bus available & idle condition time
- Queued data transfers
  - Transmit FIFO (TX-FIFO)
  - Receive FIFO (RX-FIFO)
- Queued control and status transfers, when controller
  - Control FIFO (C-FIFO)
  - Optional status FIFO (S-FIFO)
- For each FIFO, optional DMA mode with a dedicated DMA channel
- Target-initiated requests
  - In-band interrupts, with payload (up to 4 bytes)
  - Up to 4 simultaneous targets, when controller
  - Bus control & hot-join request

life.augmented

4

This slide and the following one list the I3C peripheral features.
The I3C peripheral can be configured to be:
- An SDR-only primary controller
- An SDR-only secondary controller
- An SDR-only target

The following timings are configurable:
- SCL high and low times, SCL stall time and bus free condition when operating as a controller
- SDA hold time and bus available and idle condition time when operating as a slave.

To facilitate the software management, the I3C peripheral implements the following queues:
- Data transfer queues: Transmit FIFO for data

bytes/words to be transmitted on the I3C bus and Receive FIFO for data bytes/words received from I3C bus

- A control FIFO called C-FIFO for control words to be sent on the I3C bus
- An optional status FIFO called S-FIFO for status words received from the I3C bus.

For each FIFO, an optional DMA mode can offload the Cortex-M33 core by moving data to and from buffers allocated in memory.

The target device supports the following requests:

- In-band interrupts, with up to 4-byte payload
- Bus control & hot-join requests.

The I3C peripheral, when acting as a controller, supports up to 4 simultaneous requests initiated by target devices.

STM32H5 I3C peripheral key features 2/2

- Frame-level messages management
- Individual event-based management
- Bus error detection and recovery
- Multi-clock domain management
  - APB clock, I3C kernel clock and I3C bus clocks
- Automatic SDA push-pull & open-drain modes by I3C hardware
- Configured GPIOs for SCL & SDA
  - Alternate function
  - With no pull
- Wakeup from Stop (with SVOS3)
  - When controller, on an acknowledged and received target request
    - hot-join
    - IBI without MDB
    - controller-role
  - When target
    - On a reset pattern detection
    - On a missed start detection

life.augmented

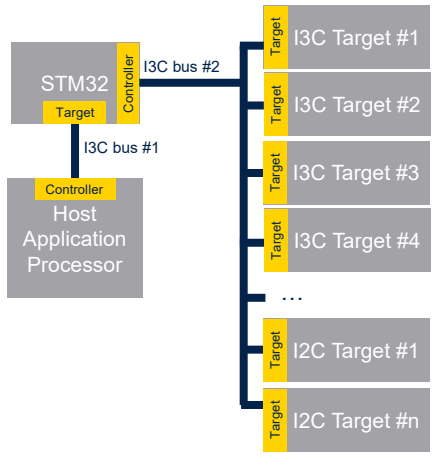
5

The software manages messages at the I3C frame level. Management is based on individual programmable events, including bus error detection and recovery.
The I3C peripheral is implemented with several clock domains: APB clock for register access, I3C kernel clock and I3C bus clock.
The I3C hardware automatically switches from/to SDA push-pull & open-drain modes. Push-pull is used when communication is unidirectional, and there is no chance of another device communicating at the same time.
The allocated GPIOs for SCL & SDA must be configured with the alternate function and no pull.
The I3C peripheral can work in autonomous mode, enabling transfers during low-power modes with DMA.

I3C can wakeup the system from low power modes on the following conditions:
- Acknowledged target request completion and DMA-based triggered frame completion when operating as a controller
- Missed start detection and reset pattern detection when operating as a target device.

STM32H503 I3C peripheral features

Implementation-dependent feature	STM32H5
Number of I3C instances	2
I3C hardware triggers	None
I3C wakeup	from Stop mode with SVOS3



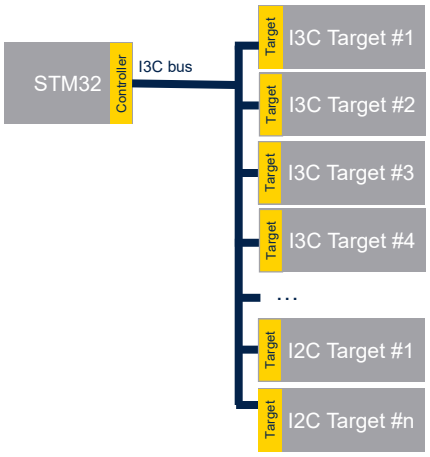
life.augmented

6

The STM32H503 includes two I3C peripheral instances, so that the device can support target operation on one I3C peripheral and operate as a controller on the other I3C peripheral.
The figure represents this possible configuration.
The I3C peripheral does not support native hardware triggers, because the ones implemented in the DMA controller can be used instead.
When the microcontroller is in stop mode, the I3C can work in autonomous mode and cause the wakeup of the CPU. In this case, the system stop mode voltage scale 3 mode shall be chosen.

STM32H562/563/573 I3C peripheral features

Implementation-dependent feature	STM32H5
Number of I3C instances	1
I3C hardware triggers	None
I3C wakeup	from Stop mode with SVOS3



life.augmented

7

The STM32H56X and STM32H573 have a unique I3C peripheral instance.
The figure represents the I3C peripheral configured as a controller in the microcontroller.
The features of the I3C peripheral are identical to the ones in the STM32H503.

MIPI specification v1.1 support

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		Controller	Target	
I3C SDR message	Y	Y	Y	
Legacy I²C message	Y	Y		Mandatory as controller when there is an I²C target Optional in MIPI v1.1 when target
HDR message	Y			Optional in MIPI v1.1
Dynamic address	Y	Y	Y	
Static address	Y	Y		I3C peripheral can not be a target/slave on an I²C bus
Grouped addressing	Y	Y		Optional in MIPI v1.1
CCCs	Y	Y	Y	All mandatory CCCs + some optional CCCs are supported
Error detection & recovery	Y	Y	Y	
IBI	Y	Y	Y	
Secondary controller	Y	Y	Y	
Hot-join	Y	Y	Y	
Target reset	Y	Y	Y	
Asynchronous timing control 0	Y	Y	Y	As controller, with limitation on the timestamp precision computation
Synchronous & asynchronous timing control 1, 2, 3	Y			Optional in MIPI v1.1
Device to device tunneling	Y	Y		Optional in MIPI v1.1
Multi-lane	Y			Optional in MIPI v1.1
Monitoring device early termination	Y			Optional in MIPI v1.1

life.augmented

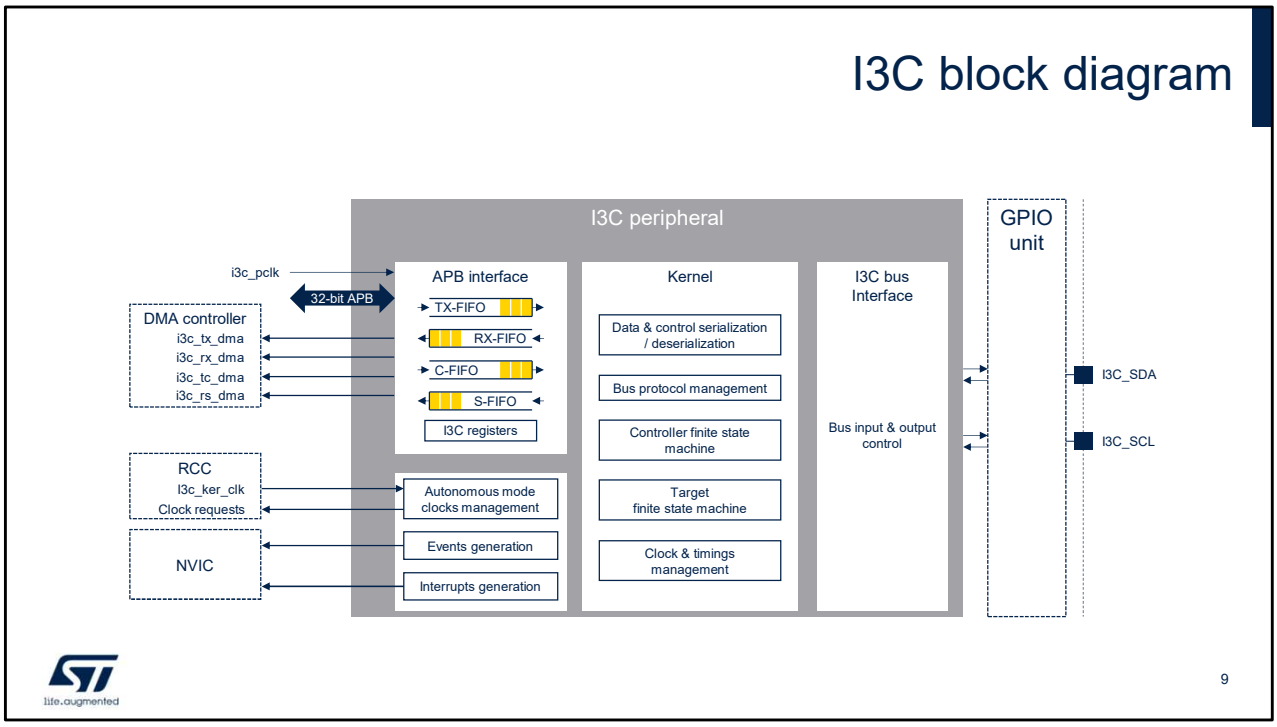
8

This table indicates which features described in the I3C MIPI specification revision 1.1, are actually implemented in the STM32H5 I3C peripheral unit.
In brief, the I3C peripheral supports all the mandatory features of the MIPI Specification v1.1.
This includes: single-data rate message, dynamic address assignment, grouped addressing, error detection and recovery, IBI, secondary controller, hot-join, target reset, asynchronous timing control mode 0.
Synchronous and asynchronous timestamping improves the accuracy of applications that fuse signals from various peripherals.
When operating as a controller, the following features are also supported: I2C static addresses, I2C message

handling and device to device tunneling.
The following optional features are not implemented:

- High data rate signaling
- Synchronous and asynchronous timing control modes 1, 2 and 3.
- Multi-lane operation
- Device early termination monitoring.

The list of supported CCC commands is detailed in the next slides.



the I3C peripheral is implemented with several clock domains:

- The SCL bus signals: for the I3C bus interface
- The I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines
- The APB clock: for the APB interface, DMA interface, events and interrupt generation.

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC, reset & clock controller block.

The I3C kernel clock and the APB clock frequencies are to be set at a minimum value, versus the intended operating

SCL bus clock frequency.
Refer to the reference manual for the list of the clock requirements and constraints.
To save dynamic power consumption, the I3C peripheral hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.
Two interrupt lines are connected to the NVIC:
- Error interrupt line
- Event interrupt line.

The four FIFOs can be programmed to generate DMA requests when they need to be serviced.

I3C controller state & programming 1/2

I3C state = DISABLED

1	**Initialize I3C bus and I3C as (primary) controller** ➢ Keep I3C_CFGR.EN = 0	Set I3C peripheral as controller: Configure I3C bus timings: Enable/disable SDA high keeper: Set own device dynamic address:	Write and set I3C_CFGR. CRINIT=1 Write I3C_TIMINGR0, I3C_TIMINGR1, I3C_TIMINGR2 Write I3C_CFGR.HKSDAEN Write I3C_DEVR0.DA[6:0]
2	**Configure the management of any target i, for i=1 to nb_targets (nb_targets <=4):**	Enable/disable IBI request: Enable/disable IBI data payload Enable/disable controller-role request: Set dynamic address: Enable/disable pending read notification:	Write I3C_DEVRI.IBIACK Write I3C_DEVRI.IBIDEN Write I3C_DEVRI.CRACK Write I3C_DEVRI.DA[6:0] Write I3C_DEVRI.SUSP
3	**Configure the execution mode of a frame/target-requested transfer:** ➢ Write I3C_CFGR fields	Enable/disable DMA mode for FIFOs: Configure TX-FIFO byte/word threshold: Configure RX-FIFO byte/word threshold: Enable/disable TX-FIFO and C-FIFO preload: Enable/disable S-FIFO: Enable/disable Hot-join Ack: Enable/disable HDR exit pattern: Enable/disable HDR reset pattern: Enable/disable arbitrable header:	TXDMAEN, CDMAEN, RXDMAEN, SDMAEN TXTHRES RXTHRES TMODE SMODE HJACK EXITPTRN RSTPTRN NOARBH
4	**Enable I3C**	write and set I3C_CFGR.EN=1	

See next slide

life.augmented

This slide and the next one illustrate the overall programming sequence of the I3C peripheral when acting as primary controller

After reset, the I3C peripheral is in the default disabled state.

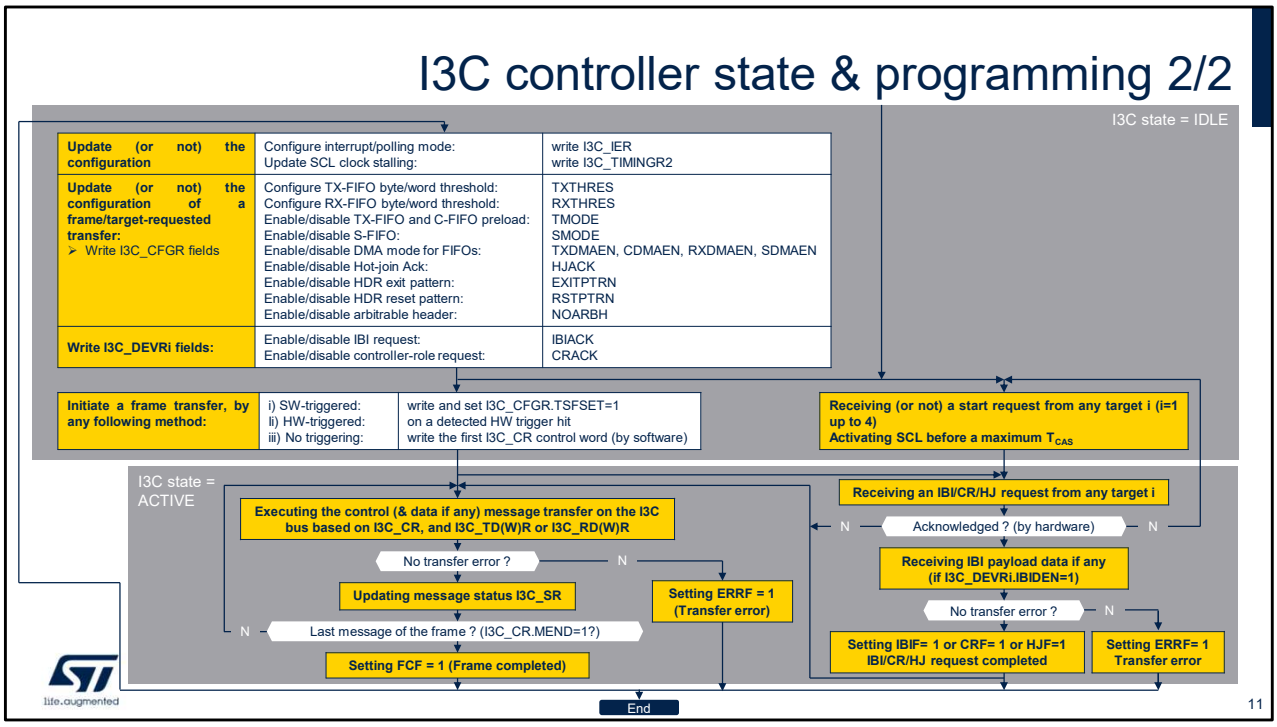
In this state, the software has to execute the sequence described in this figure prior to enabling the I3C peripheral.

The three initialization steps are detailed:

- Initialize I3C bus and select controller operation
- Configure the management of all targets
- Configure the execution mode of a frame and target-requested transfer

Then when software writes 1 into the controller enable bit of the I3C_CFGR register, the I3C peripheral takes into

account the value of the different configuration registers and switches to the enabled and idle state.



Once the I3C controller is in idle state, it is able to initiate a frame transfer, or concurrently receive a start request and/or a target initiated transfer (IBI), Hot join or controller role request from an allowed target.

In the first case, the I3C peripheral switches to the active state and executes the frame transfer on the I3C bus until the completion of the last message, unless a transfer error error is signaled.

See the left part of the Active state box in the figure.

The software is notified by the corresponding event and/or interrupt and the I3C switches back to IDLE state where it may be re-configured for a next frame transfer.

In case of target initiated transfer, the hardware receives and logs the request from the target in its registers and an

event is generated to notify the software on the completion of the frame or on a detected error.

I3C target state & programming 1/2

I3C state = DISABLED

1	**Initialize I3C as target:** ➢ Keep I3C_CFGR.EN = 0	Set I3C peripheral as target: Set I3C bus available/idle/hand-off condition timings: Enable/disable IBI request: Enable/disable controller-role request: Enabled/disable hot-join: Set/initalize target characteristics and capabilities:	write and clear I3C_CFGR. CRINIT=0 write I3C_TIMINGR1.AVAL[7:0] write I3C_DEVR0.IBIEN write I3C_DEVR0.CREN write I3C_DEVR0.HJEN write I3C_BCR, I3C_DCR, I3C_MAXRLR, I3C_MAXWLR, I3C_GETCAPR, I3C_CRCAP, I3C_GETMXDSR, I3C_EPIDR
2	**Configure (the execution mode of) a transfer via I3C_CFGR:**	Configure TX-FIFO byte/word threshold: Configure RX-FIFO byte/word threshold: Enable/disable DMA mode for FIFOs: Disable S-FIFO for I3C_SR:	TXTHRES RXTHRES TXDMAEN, RXDMAEN SMODE=0
3	**Enable I3C**	write and set I3C_CFGR.EN=1	

See next slide

life.augmented

This slide and the next one illustrate the overall programming sequence of the I3C peripheral when acting as target.

When target is in disabled state, the software performs the following initializations:

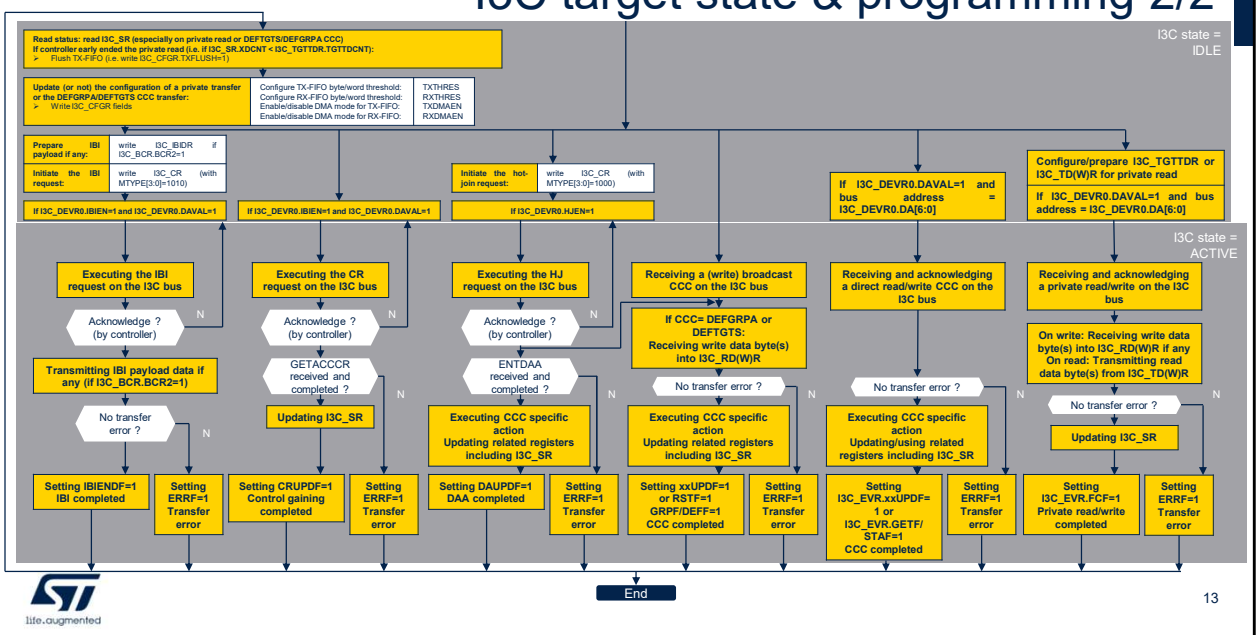
- Set I3C peripheral as target
- Set I3C bus timings
- Configure target-initiated requests
- Initialize target characteristics and capabilities
- Configure the execution mode of a transfer
- Configure interrupt generation or polling mode from any event.

Then, the software can enable the I3C peripheral.

The following target-initiated requests are supported:

- In-band interrupt
- Controller-role request
- Hot-joing request.

I3C target state & programming 2/2



When the target is in idle state, the target is ready to receive a communication message on the I3C bus from the controller and is ready to switch to the active state.
All sub-states of the ACTIVE state are represented in the bottom part of the figure.
From left to right, the various communication messages that are handled by the target are:

- In-band Interrupt
- Controller-role request
- Hot-Join
- Receipt of a write broadcast CCC
- Receipt of a direct read or write CCC
- Receipt of private read / write request.

Common Command Codes (CCCs) are I3C's standardized

commands.
A Broadcast Write CCC is seen by all I3C Slaves.
A Direct Read/Write CCC may alternatively read or write data from/to one or more specific I3C Slaves, one Slave at a time, selected by the Slave Dynamic Address.
A private read / write is a memory-mapped access to a register in the target.
Back in idle state, the software may update the configuration of the I3C target before a next transfer, refer to the upper left part of the IDLE state description.

I3C command set – Broadcast CCCs

CCC name		CCC value	Rd/ Wr	With or Without one defining or one sub-command byte	With or without optional data byte(s)	Used as controller	Used as target, raised I3C_EVR event
ENEC	Enable Events Command	0x00	Wr	N	Y (1)	X	X, INTUPDF
DISEC	Disable Events Command	0x01				X	X, INTUPDF
ENTASx x=0…3	Enter Activity State 0…3	0x02..0x05			N	X	X, ASUPDF
RSTDAA	Reset Dynamic Address Assignment	0x06			-	X	X, DAUPDF
ENTDAA	Enter Dynamic Address Assignment	0x07			-	X	X, DAUPDF
DEFTGTS	Define List of Targets	0x08			Y (n)	X	X, DEFF
SETMWL	Set Max Write Length	0x09			Y (2)	X	X, MWLUPDF
SETMRL	Set Max Read Length	0x0A			Y (2 or 3)	X	X, MRLUPDF
ENTTM	Enter Test Mode	0x0B			Y (1)	X	-
SETXTIME	Exchange Timing Information	0x28		Y	N and Y (n)	X	
SETAASA	Set All Addresses to Static Addresses	0x29		N	N	X	
RSTACT	Target Reset Action	0x2A		Y		X	X, RSTF
DEFGRPA	Define List of Group Address	0x2B		N	Y (n)	X	X, GRPF
RSTGRPA	Reset Group Address	0x2C			N	X	-

life.augmented

14

This table lists all supported broadcast CCCs.
Depending on the particular CCC format, a sub-command byte and data bytes may be provided by the controller.
Number of data bytes is indicated between parenthesis.
The ENEC and DISEC CCCs allow the Controller to control when Target-initiated traffic is allowed vs. is not allowed on the I3C Bus.
Dynamic Addressing is required by the I3C protocol, while supporting Static Addressing for Legacy I2C Devices.
As an optional feature, multiple I3C Target Devices can share a single Group Address, allowing a Controller Device to send a given I3C Message to all Target Devices in the Group at once rather than one at a time.
The following CCCs are related to addressing:

- ENDAA
- SETAASA
- DEFGRPA
- RSTGRPA.

The ENTAS0-3 CCCs allow the Active Controller to inform one or all Target Devices that it will not be active on the I3C Bus for an approximated amount of time, so that the Target Devices may use a lower power state during that period.
The DEFTGTS CCC tells Secondary Controller Devices what Targets (and Groups) are present on the I3C Bus, via four consecutive Data Bytes per Target (or Group).
The SETMWL/SETMRL CCCs allow the I3C Controller to set the maximum data write length and respectively maximum read length for all Target Devices.
The ENTTM CCC informs all I3C Devices that the Controller is entering a specified Test Mode during manufacturing or Device test.
The SETXTIME CCC provides the framework for Controller(s) and Target(s) to exchange event timing information for purposes such as synchronizing controls, collecting or reconstructing timestamps, and specifying the timing data procedure.
The RSTACT CCC is used to configure the next Target Reset action.

I3C command set – Direct CCCs 1/2

CCC name		CCC value	Rd/ Wr	With or Without defining or sub-command byte	With or without optional data byte(s)	Used as controller	Used as target, raised I3C_EVR event
ENEC	Enable Events Command	0x80	Wr	N	Y (1)	X	X, INTUPDF
DISEC	Disable Events Command	0x81				X	
ENTASx x=0…3	Enter Activity State 0…3	0x82..0x85			N	X	X, ASUPDF
SETDASA	Set Dynamic Address from Static Address	0x87				X	
SETNEWDA	Set New Dynamic Address	0x88			Y (1)	X	X, DAUPDF
SETMWL	Set Max Write Length	0x89			Y (2)	X	X, MWLUPDF
SETMRL	Set Max Read Length	0x8A			Y (2,3)	X	X, MRLUPDF
GETMWL	Get Max Write Length	0x8B	Rd		Y (2)	X	X, GETF
GETMRL	Get Max Read Length	0x8C			Y (2,3)	X	
GETPID	Get Provisioned ID	0x8D			Y (6)	X	

life.augmented

15

This slide and the next one list all supported direct CCCs. The ones that can also be transported through a broadcast CCC have been explained in the previous slide.
The SETDASA allows the Controller to assign a Dynamic Address to one Target using the Target's Static Address. This is faster than the ENTDAA Dynamic Address Assignment procedure.
The SETDASA CCC should be used before the ENTDAA CCC is used; all Targets without assigned Dynamic Addresses will respond to the ENTDAA CCC.
The SETNEWDA allows the I3C Controller to assign a new Dynamic Address to one I3C Target or Secondary Controller.
The GETMWL and GETMRL CCCs allow the I3C

Controller to Get the maximum data write and respectively maximum read length in bytes for one target Device.
The GETPID CCC is a Get request for one I3C Target Device to return its 48-bit Provisioned ID to the Controller. This unique ID is used by the dynamic address assignment procedure.

I3C command set – Direct CCCs 2/2

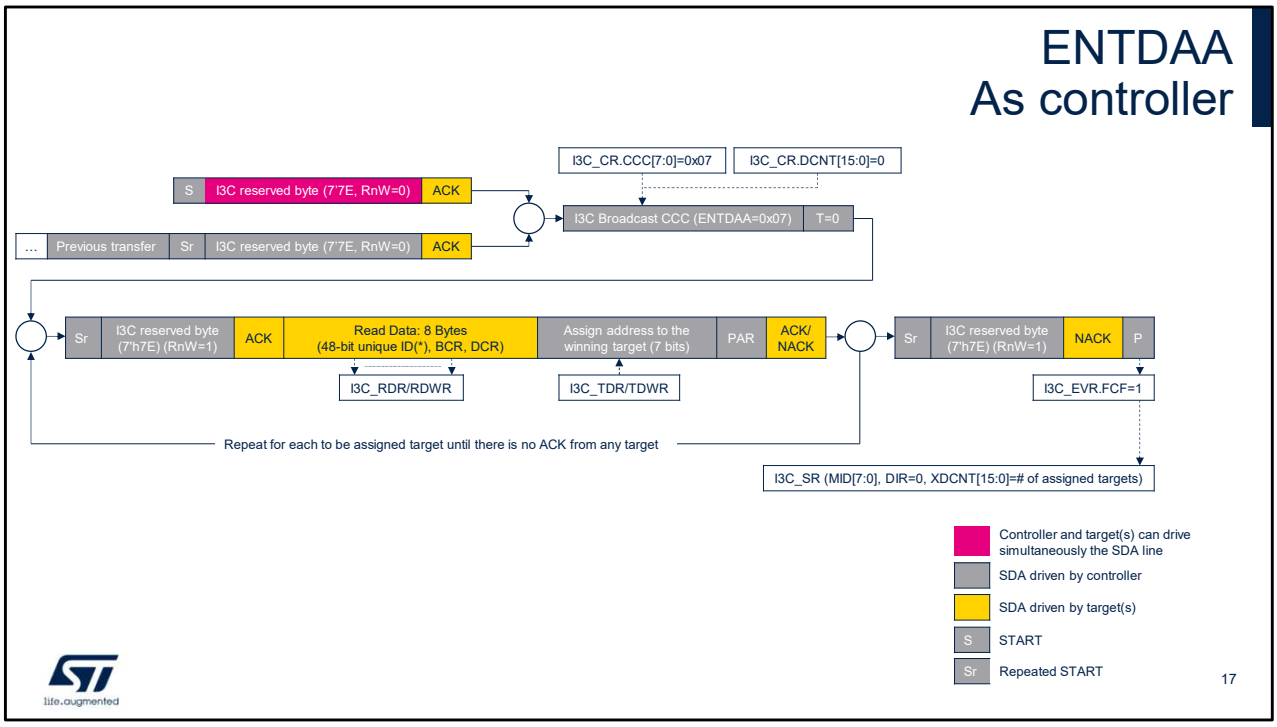
CCC name		CCC value	Rd/ Wr	With or Without defining or sub-command byte	With or without optional data byte(s)	Used as controller	Used as target, raised I3C_EVR event
GETBCR	Get Bus Characteristics Register	0x8E	Rd	N	Y (1)	X	X, GETF
GETDCR	Get Device Characteristics Register	0x8F				X	
GETSTATUS	Get Device Status	0x90		N and Y	Y (2)	X	X, STAF if format 1 X, GETF if format 2
GETACCCR	Get Accept Controller Role	0x91		N	Y (1)	X	X, CRUPDF
GETMXDS	Get Max Data Speed	0x94		N and Y	Y (1, 2, 5)	X	X, GETF
GETCAPS	Get Optional Feature Capabilities	0x95			Y (2, 3)	X	
D2DXFER	Device to Device(s) Tunneling Control	0x97	Wr	Y	Y (1)	X	
SETXTIME	Set Exchange Timing Information	0x98				X	
GETXTIME	Get Exchange Timing Information	0x99	Rd	N	N	X	

life.augmented

16

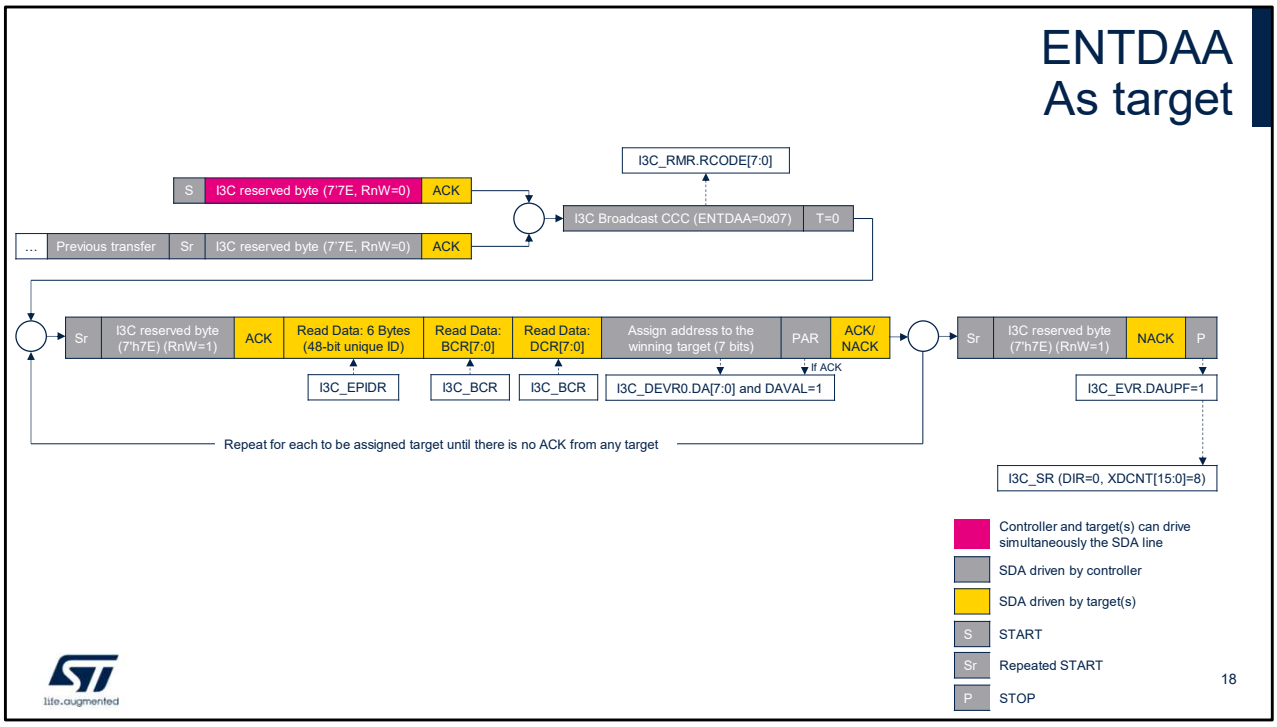
The GETBCR and GETDCR CCC are a Get request for one I3C Target Device to return its Bus Characteristics Register (BCR), respectively Device Characteristics Register (DCR) to the Controller.
The GETSTATUS CCC is a Get request for one I3C Target Device to return its current Status to the controller. The status information includes activity mode, error status and pending interrupt status.
The GETACCR CCC is used both to verify a Controller Role Request, if received earlier; and to allow the Active Controller to offer (i.e., to pass) the Controller Role to an I3C Secondary Controller, even if it was not previously requested.
The Controller uses the GETMXDS to determine the SDR

Mode data speed limitations of one Target Device.
The GETCAPS CCC allows the Controller to query an I3C Target to determine what optional I3C features that Target supports.
The D2DXFER CCC initiates and controls the Device to Device(s) Tunneling procedure.
The SETXTIME CCC provides the framework for Controller(s) and Target(s) to exchange event timing information for purposes such as synchronizing controls, collecting or reconstructing timestamps, and specifying the timing data procedure.
The GETXTIME CCC provides the framework for the Controller to query the Exchange Timing capabilities supported by the I3C Targets.



This figure illustrates the I3C broadcast ENTDAA CCC, as it is communicated on the I3C bus and as it is programmed when acting as controller.
After ENTDAA is presented, followed by START or repeated START with the reserved byte, the Acknowledge is driven from the addressed targets.
The target then sends its 48-bit unique and provisioned ID in open-drain, which is subject to arbitration.
The target which wins arbitration then continues, providing its BCR and DCR.
The controller transfers the 7-bit dynamic address for the winning device in open drain, plus parity bit.
If parity bit is OK, the target shall acknowledge.
The procedure is iterated until the controller detects a

NACK condition after having transferred the reserved byte. This means that all targets have been assigned a dynamic address.
Note that the figure mentions the registers of the I3C peripheral that the controller's software has to access during the sequence.



This figure illustrates the I3C broadcast ENTDAA CCC, as it is communicated on the I3C bus and as it is programmed when acting as target.

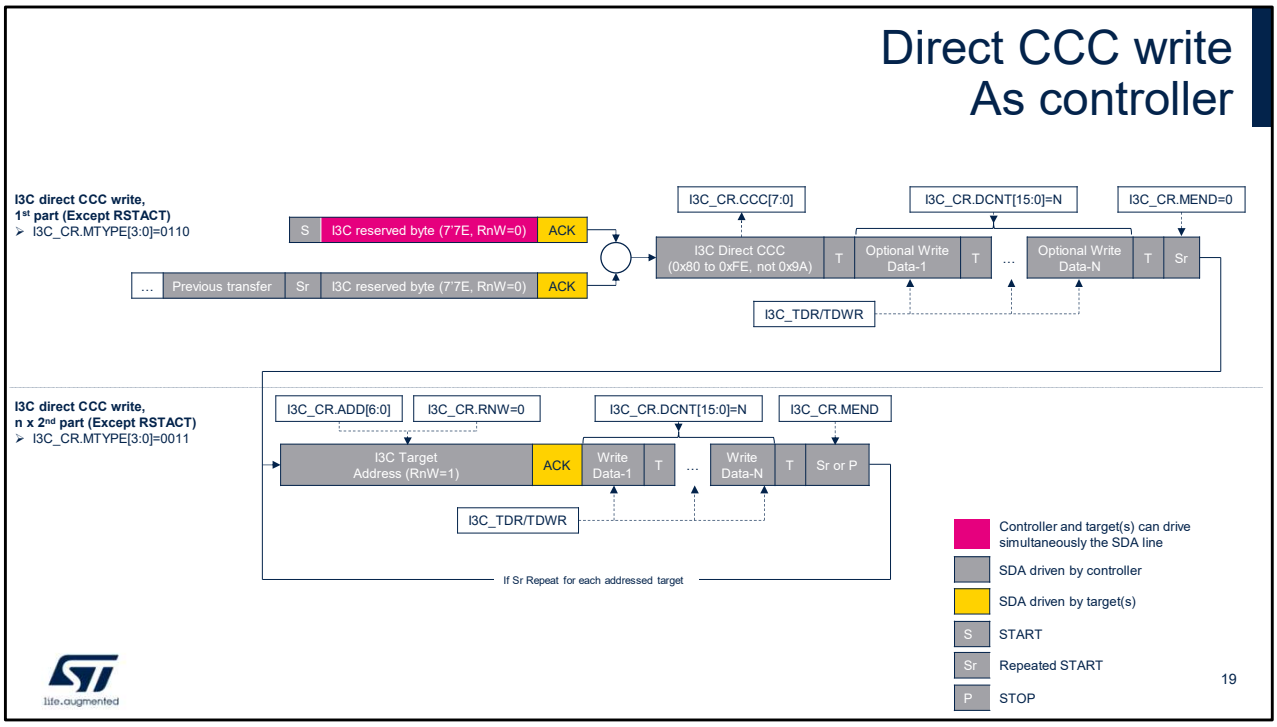
The assigned address is captured in the I3C_DEVR0 register.

Here is the format of the provisioned ID:

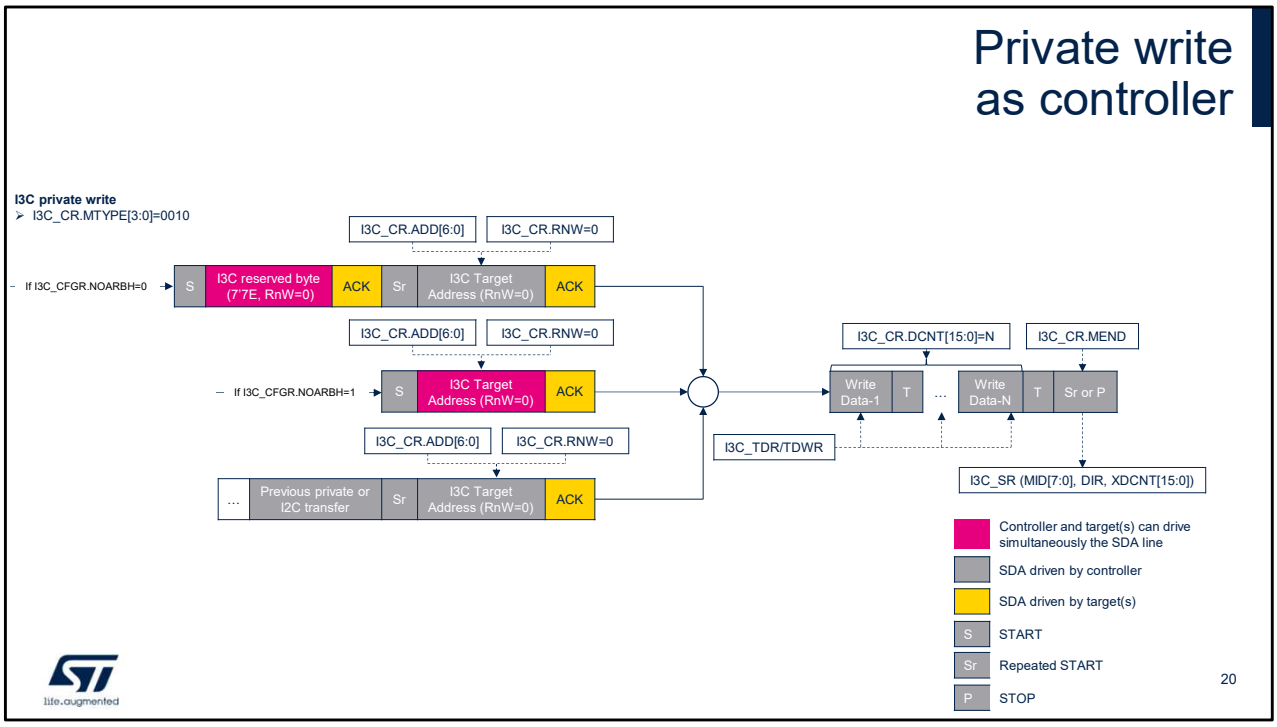
- bits[47:33] of the 48-bit provisioned ID = 15-bit STMicroelectronics MIPI ID (0x0104)
- bit[32] of the 48-bit provisioned ID = 0 = vendor fixed value
- bits[31:16] of the provisioned ID =0
- bits[15:12] of the 48-bit provisioned ID= MIPIID field of the I3C_EPIDR register

This MIPIID field is written by software to set and identify

individually each instance of this I3C IP with a unique number on a single I3C bus.



This figure illustrates the I3C direct CCC write transfer (except RSTACT), as it is communicated on the I3C bus and as it is programmed when acting as controller.
The controller first issues the I3C reserved byte with RnW set to 0.
Once properly acknowledged, the controller transmits the CCC code and subsequent optional bytes, if CCC defines byte, sub-command byte and data byte(s)
As MEND=0, this message is followed by a repeated start, before another message is to be emitted.
In the second message, the controller transmits the target address followed by N bytes.
When MEND=1, this message from controller ends with a stop (P), being the last message of a frame.



This figure illustrates an I3C private write transfer, as it is communicated on the I3C bus and as it is programmed when acting as controller.
The MTYPE field of the control register selects the message type, binary 00010 for private message.
This private write transfer can be scheduled when one of the following conditions is satisfied:

- An arbitrable header containing the reserved byte and RnW=0 is emitted after a start and before the I3C private write message
- The target address is emitted directly after a start
- The I3C write message follows a previous I2C or private I3C transfer.

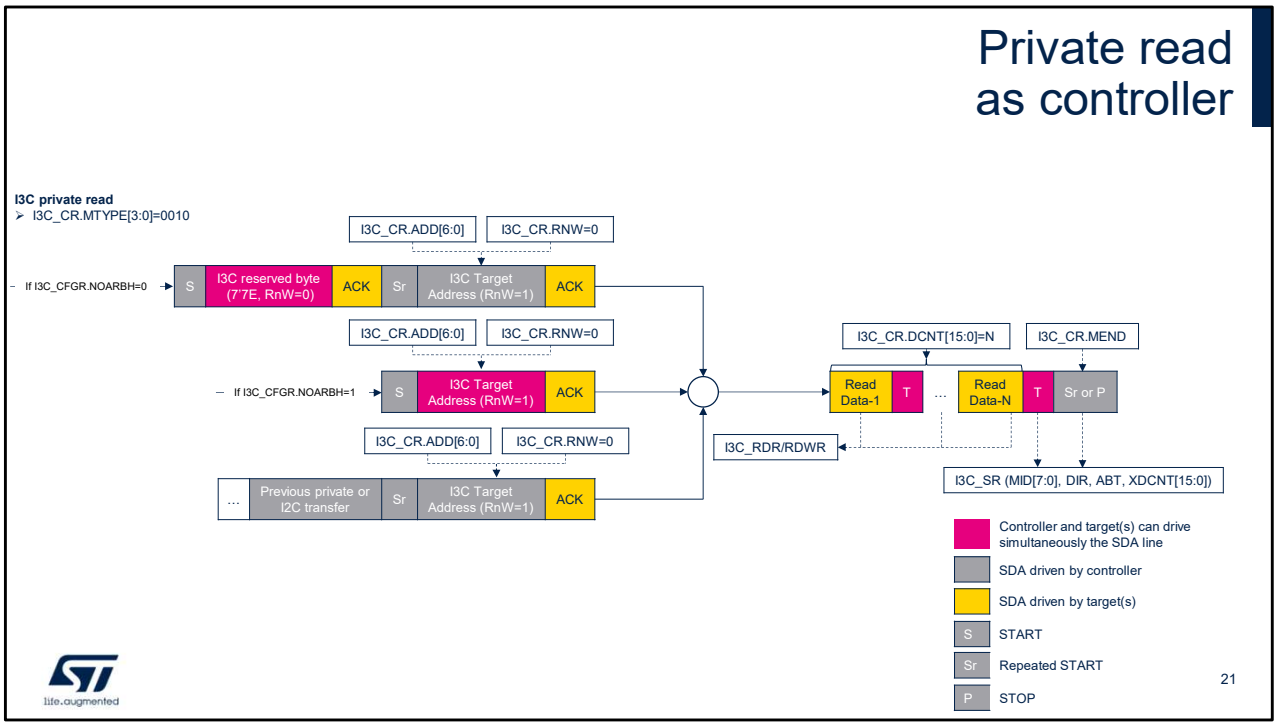
The controller selects the target by transferring its address

and indication of a write data transfer direction.
Then data bytes are sent by the controller, separated by transition bit.
The transfer ends with a STOP or repeated START condition.
Configuration fields that have to be initialized by the controller:

- Address of the target
- Direction of the data transfer
- Number of bytes to be transferred.

The status register is updated with:

- The control word message ID programmed in the control register to which the status information refers
- The direction of the transfer
- The number of transmitted data bytes.



This figure illustrates an I3C private read transfer, as it is communicated on the I3C bus and as it is programmed when acting as controller.

This private read transfer can be scheduled when one of the following conditions is satisfied:

- An arbitrable header containing the reserved byte and RnW=0 is emitted after a start and before the I3C private read message
- The target address is emitted directly after a start
- The I3C read message follows a previous I2C or private I3C transfer.

The controller selects the target by transferring its address and indication of a read data transfer direction.

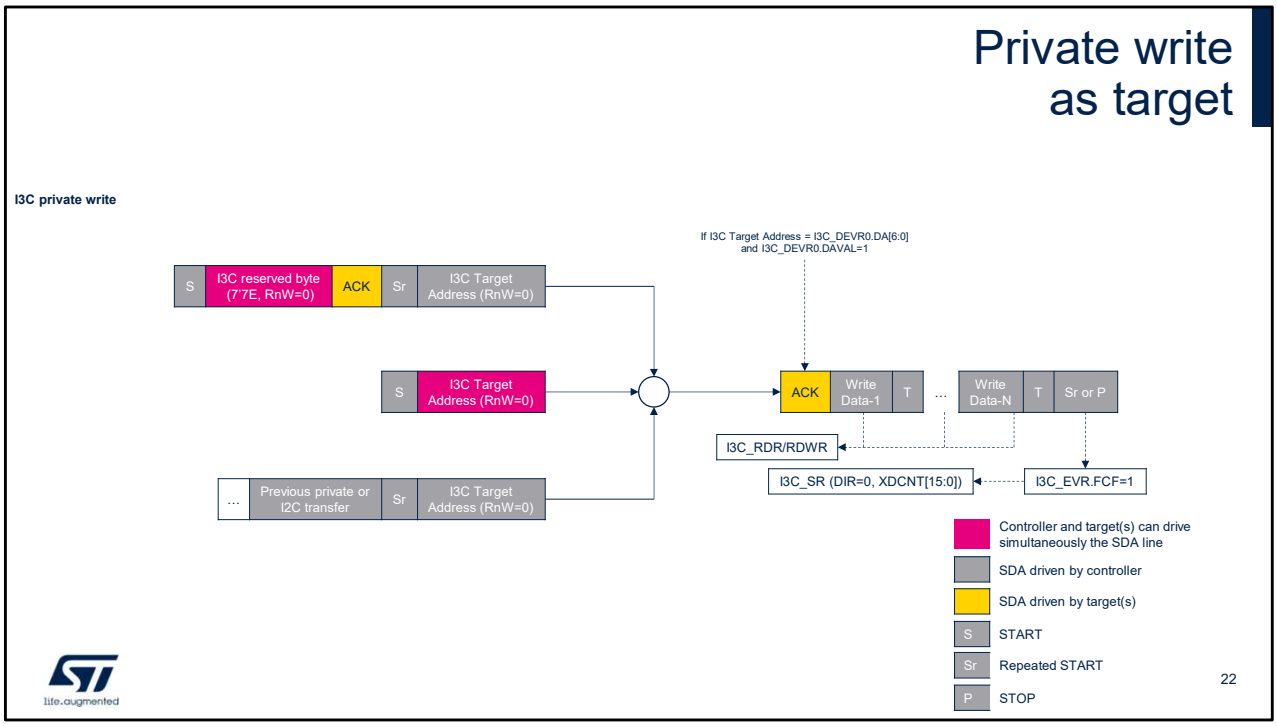
Then data bytes are sent by the target, separated by

transition bit.
The transfer ends with a STOP or repeated START condition.
Configuration fields that have to be initialized by the controller:

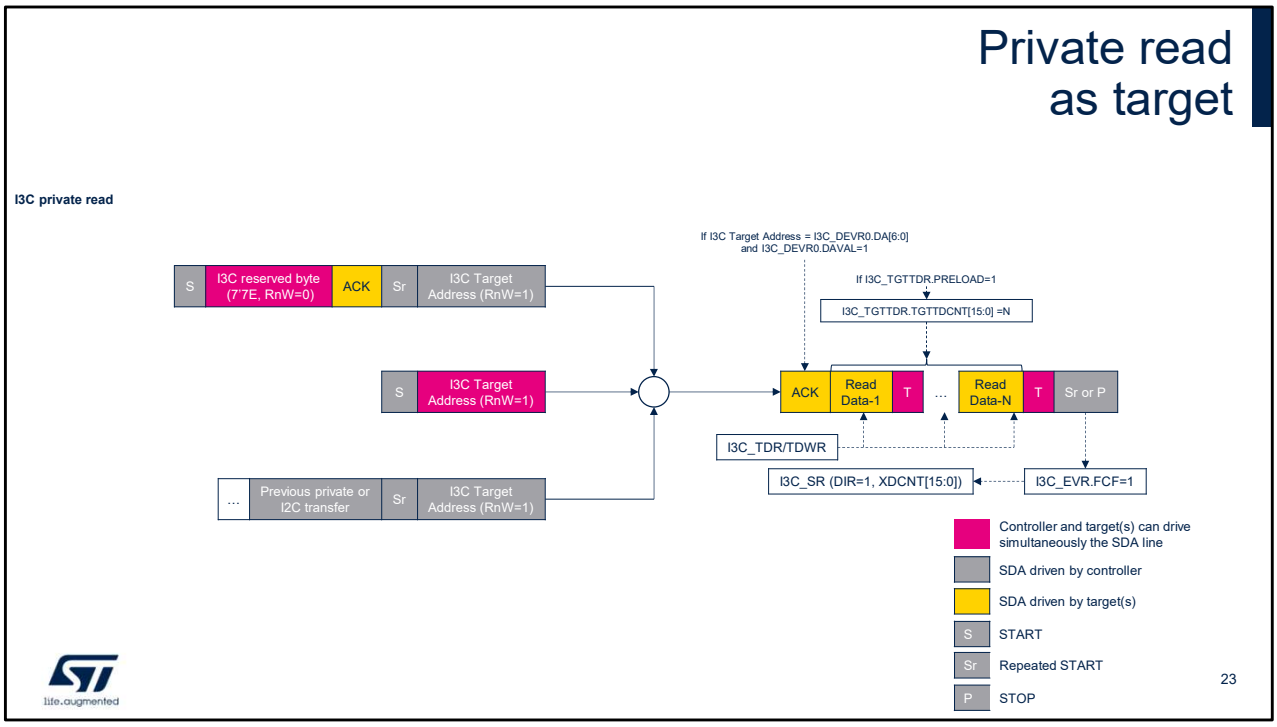
- Address of the target
- Direction of the data transfer
- Number of bytes to be received.

The status register is updated with:

- The control word message ID programmed in the control register to which the status information refers
- The direction of the transfer
- The number of transmitted data bytes
- An abort flag is set in case of early completion from the target.

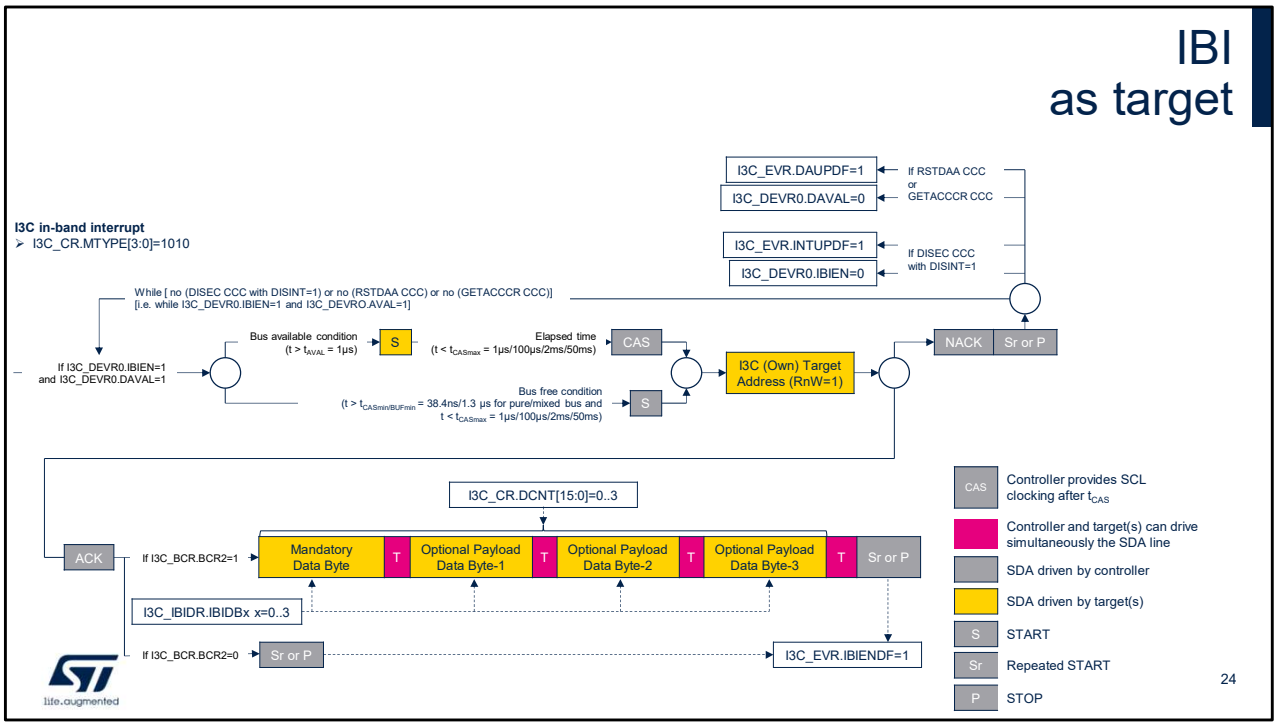


This figure illustrates an I3C private write transfer, as it is communicated on the I3C bus and as it is programmed when acting as target.
Each target compares the target address with its assigned address. DAVAL bit set indicates that dynamic address is valid. When a match occurs, the target acknowledges the request and prepares to receive data bytes.
The Frame Complete Flag (FCF) is asserted by hardware to indicate that a message addressed to this target has been normally completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.
The status register is updated with the direction of the transfer and the number of received bytes.



This figure illustrates an I3C private read transfer, as it is communicated on the I3C bus and as it is programmed when acting as target.
Each target compares the target address with its assigned address.
When a match occurs, the target acknowledges the request and prepares to transmit data bytes.
The Frame Complete Flag (FCF) is asserted by hardware to indicate that a message addressed to this target has been normally completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.
The status register is updated with the direction of the transfer and the number of transmitted bytes.

Bytes to be transmitted may have been preloaded into the TxFIFO.



This figure illustrates IBI (in-band interrupt) transfer, as it is communicated on the I3C bus and as it is programmed when acting as target.

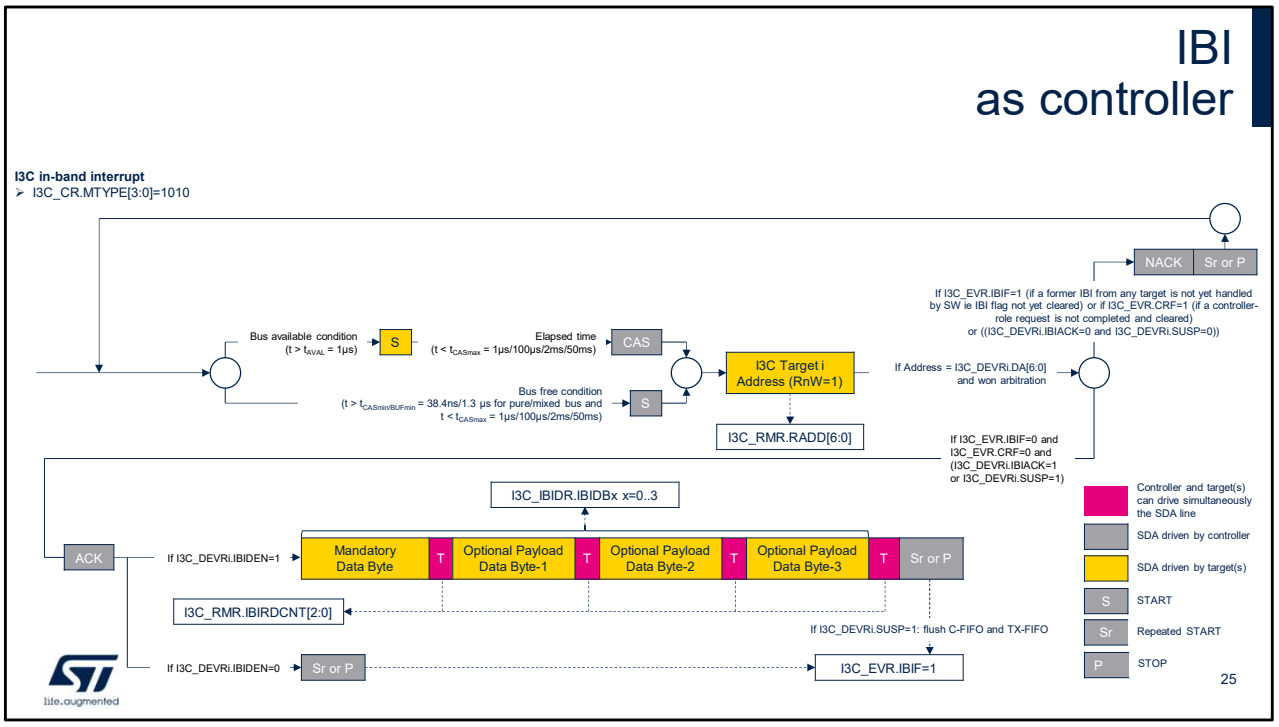
IBI is enabled when IBIEN bit is set to one and the target was assigned a dynamic address, which is confirmed by DAVAL bit set to one.

In the upper right part of the figure, the IBI is masked due to either the IBI target event disabling or dynamic address resetting.

In order to request an interrupt, an I3C Slave shall emit its Address into the arbitrated Address header following a START (but not following a Repeated START).

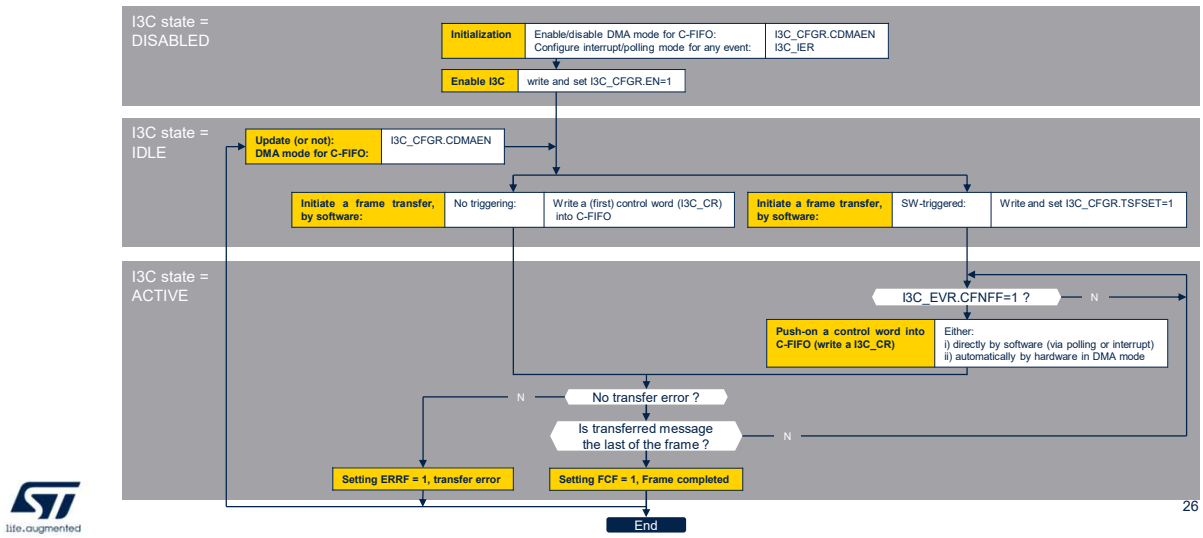
If no START is forthcoming within the Bus Available Condition, then the I3C Slave may issue a START by

pulling the SDA line Low, and then wait for the Current Master to pull the SCL Low.
If the Current Master sets the SCL line Low, thus completing the START, and then provides clocks on the SCL line, then the Slave shall drive the SDA line with its own Address, followed by an RnW bit with value 1.
If more than one Slave has issued an IBI request after the same START, then the Current Master shall process those IBIs in Priority Level order.
This arbitration loss case corresponds to the arrow right to left from NACK condition back to the beginning of the sequence.
The Slave that lost the Arbitration may issue another IBI request but shall not do so until after the Bus Available Condition.
The slave which has won the arbitration transfers mandatory and optional payload bytes, as represented in the bottom part of the figure. These bytes identify the cause of the interrupt request.



This figure illustrates IBI (in-band interrupt) transfer, as it is communicated on the I3C bus and as it is programmed when received as controller.
The address of the target that has won the arbitration is captured into the Received Message Register (RMR).
The IBIF flag is asserted by hardware to indicate that an IBI request has been received.
This flag is cleared when software writes 1 into that bit, to indicate that the current IBI has been handled.
When an IBI is received and the previous one is not yet handled by the controller, this second IBI is ignored.
The I3C_IBIDR register is used to receive the IBI data payload and IBIRDCNT field logs the number of data bytes effectively received in the I3C_IBIDR register.

I3C FIFOs management
C-FIFO as controller



The software should initialize the C-FIFO management to be then written either:

- Directly by the software
- Or by the allocated DMA channel.

As soon as the transfer is initiated, a transition to active state occurs.

When a DMA channel is used, the DMA automatically pushes-on/writes control words into the I3C_CR from its memory source buffer, until the frame completes.

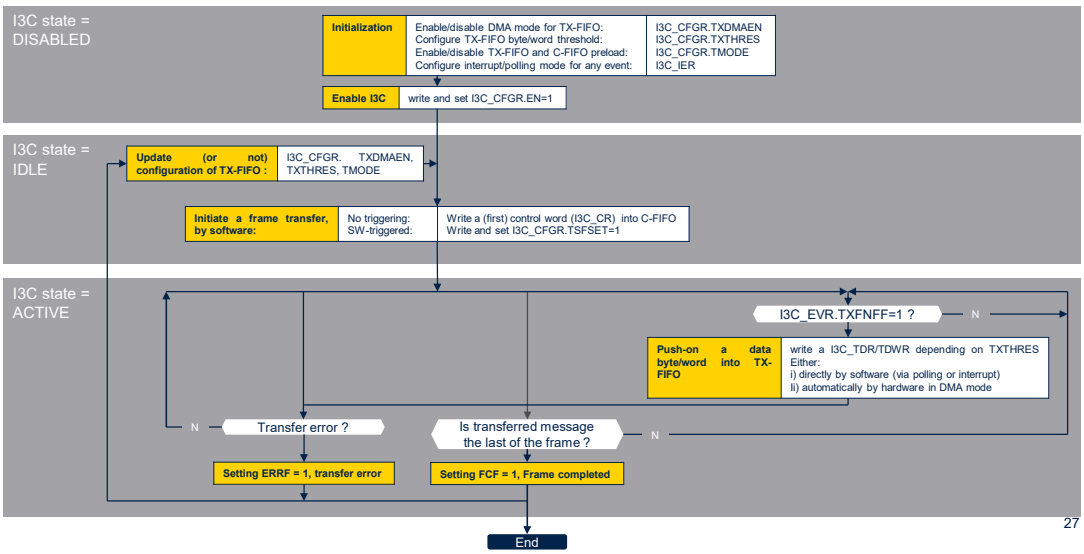
Software management is based on the C-FIFO not full flag, CFNFF.

Software C-FIFO refill can be based on polling or interrupt.

Polling mode waits for a next control word is requested by the hardware, which is indicated by the CFNFF flag set to

one, before an explicit write to the I3C_CR register.
When interrupt mode is chosen, an interrupt is requested when the CFNFF flag is set to one by the hardware.
The message continues until either an error occurs, or it is the last of the frame.
The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.
If a transfer error occurs, then the C-FIFO is automatically flushed by the hardware.
If C-FIFO is empty and a restart has to be emitted with a new control word, a C- FIFO underrun is reported.

I3C FIFOs management
TX-FIFO as controller



The software should initialize the TX-FIFO management, through the following programmable settings:

- Enabling/disabling DMA mode for TX-FIFO
- Configuring TX-FIFO threshold
- Enabling/disabling TX-FIFO preload
- Enabling/disabling interrupt.

The threshold defines when the Tx-FIFO Not Full Flag TXFNFF flag is set.

When threshold control bit is equal to 0: 1-byte threshold TXFNFF is set when 1 byte is to be written in TX-FIFO through the I3C_TDR register.

When threshold control bit is equal to 1: one-word/four-byte threshold TXFNFF is set when one word / four bytes is/are to be written in TX-FIFO through the I3C_TDWR

register.
As soon as the transfer is initiated, a transition to active state occurs.
When a DMA channel is used, the DMA automatically pushes-on/writes data bytes/words to the I3C_TDR or I3C_TDWR register from its memory source buffer, until the frame completes.
Software TX-FIFO refill can be based on polling or interrupt.
The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.
If a transfer error occurs, then the TX-FIFO is automatically flushed by the hardware.
If TX-FIFO is empty and a data byte has to be transmitted on the I3C bus, a TX-FIFO underrun is reported.

I3C interrupt 1/2

Interrupt acronym		Interrupt event	As controller	As target
I3C	EVT	A control word is requested	Y	
		A status word is available	Y	
		A data to be transmitted is requested	Y	Y
		A received data is available	Y	Y
		A frame transfer is completed, as controller A private transfer is completed, as target	Y	Y
		A private read transfer is prematurely ended by the target (and I3C_CFGR.SMODE=0)	Y	
		An IBI request is received	Y	
		A controller-role request is received	Y	
		A hot-join request is received	Y	
	ERR	An error occured	Y	Y

life.augmented

28

The table in this slide and the next one lists all interrupt sources present in the I3C peripheral.
All event interrupt sources are Ored together and reported to the NVIC by using the i3c_evt_it line.
All error interrupt sources are Ored together and reported to the NVIC by using the i3c_err_it line. Refer to the reference manual for the complete list of error types.
Each interrupt source can be individually masked or enabled.
The four events at the beginning of the table can be used as DMA requests to inform the DMA that data has to be transferred between memory and I3C peripheral's FIFOs.

I3C interrupt 2/2

Interrupt acronym		Interrupt event	As controller	As target
I3C	EVT	IBI request is completed		Y
		I3C bus start is missed		Y
		GETACCCR CCC is received		Y
		GETSTATUS CCC is received		Y
		Any GETxxx CCC (except GETSTATUS) is received		Y
		ENTDAA/RSTDAA/SETNEWDA CCC is received		Y
		SETMWL CCC is received		Y
		SETMRL CCC is received		Y
		A reset pattern is detected		Y
		ENTASx CCC is received		Y
		ENEC/DISEC CCC is received		Y
		DEFTGTS CCC is received		Y
		DEFGRPA CCC is received		Y

life.augmented

29

Events in this slide are related to the I3C peripheral acting as a target.
Most of them correspond to the receipt of a particular CCC.
In addition, completion of an IBI, a reset pattern detection and a missed start are signaled.

I3C wakeup from stop with SVOS3

As controller
Wakeup on a hot-join request (HJF)
Wakeup on a controller-role request (CRF)
Wakeup on a IBI without Mandatory Data Byte (IBIF)

As target
Wakeup on a target reset pattern (RSTF)
Wakeup on a missed start (WKPF)

life.augmented

30

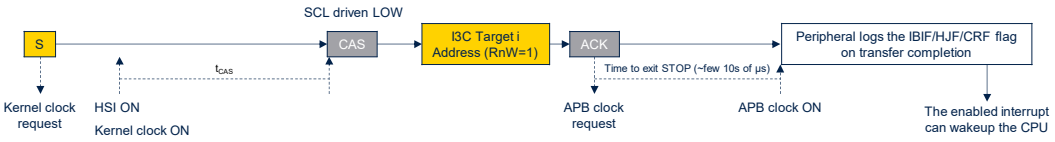
When entering Stop, the VCORE domain is supplied, by default any clock oscillator is disabled, and in any case neither the system clock nor a peripheral clock is running in the VCORE domain

A controller can wake-up on an IBI without MDB, a hot-join request or a controller-role request.

A target can wake-up on a target reset pattern, or on a missed start, as explained in more details in the next slides.

I3C wakeup from stop with SVOS3 as controller

- Exit from Stop due to Hot-Join, Controller-role request and In-Band Interrupt



life.augmented

31

When the peripheral acts as controller, before the product enters a low-power mode, the
software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that
the I3C controller is not expected to exit from idle state neither to communicate on the I3C
bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay
defines the TCAS delay for the controller to set the SCL bus clock low and running, after a
start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx
with the value x=1, 2, or 3.

The I3C peripheral is able able to wake-up the system from Stop mode, this is only possible when SVOS3 is selected before entering Stop mode.
This slide describes the wakeup sequence when the I3C peripheral acts as a controller.

1. First the I3C peripheral requests its kernel clock to the system:
– On a detected start condition on the I3C bus (SDA line is detected to be driven low
while SCL is high). As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets
ready (after a few microseconds from HSI). The user may keep the source oscillator ON
in Stop mode to reduce this startup latency, at the expense of power consumption.
Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding TCAS time to
elapse, then drives low SCL and continues toggling SCL to let the target perform
its hot-join/in-band interrupt/controller-role request on the I3C bus.
– The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.

3. as controller the I3C peripheral requests its APB clock to the system:

– On the ACKed address of a received IBI request and it maintains the
APB clock request until that IBIF is cleared.
If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the
APB clock is not yet provided.
– On the ACKed address of a received controller-role request and it
maintains the APB clock request until that CRF is cleared.
– On the ACKed address of a hot-join request and it maintains the
APB clock request until that HJF is cleared.

4. The system is notified of the I3C APB clock request, and the power management unit
of the PWR module is awaken.

5. An additional delay may be needed for the regulator, if it must increase the voltage for
Run mode.

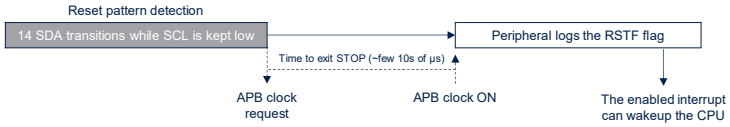
6. The system enables the system clock that drives the APB clock. There is an additional
delay if the selected oscillator source for the system and APB clocks is not the same as
the one driving the I3C kernel clock. If so, the system enables the source oscillator of
the system clock, which gets ready after a delay.

7. With the APB clock running, the peripheral can log the I3C transfer in its status and

data registers. When the bus transfer is completed, the peripheral generates the
corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

I3C wakeup from stop with SVOS3 as target

- Exit from Stop due to target reset pattern



The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

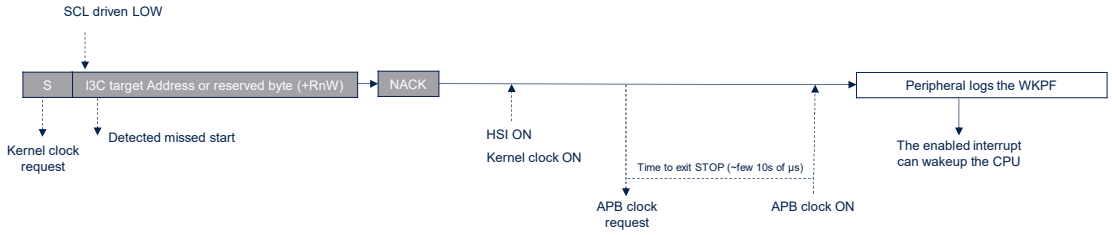
As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit

of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for
Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after
few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR
register, and the enabled interrupt can wake up the CPU.

I3C wakeup from stop with SVOS3 as target

- Exit from Stop due to missed start



As target, the wake-up sequence on a missed start is the following:

1. The peripheral requests its kernel clock to the system:

– On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high)

– As target, as initiated by an external I3C device, either controller or target.

– If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that an I3C bus start is missed and waits for the kernel clock to be provided. It can be noted that an I3C controller message intended to be addressed to it may have been missed (and NAcked). Or may have been missed any I3C CCC or private message from the controller to another addressed target or an In-Band Interrupt, Controller-Role Request, Hot-Join start

request from another target.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HIS is the clock source). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
Especially if the controller is in an Activity State of x = 0 (with a 1 µs TCAS latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag conditioned by non-user text or:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a few microseconds.
7. With the APB clock running:
– The missed start flag (WKPF of the I3C_EVR register) is raised conditioned by non-user text: if it occurred, and the enabled interrupt can wake up the CPU. It is then known that an I3C bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout in order to return to Stop mode, if it is not addressed by the controller again within this interval.

Thank you

life.augmented

Refer to the Application Note AN5879, that provides some I3C examples based on the STM32CubeMX, to cover most of the I3C communication modes, available on the STM32 microcontrollers, and to provide recommendations for the correct use of the I3C peripheral.

In addition to this presentation, you can refer to the following presentations:

- Reset and Clock Control
- Nested Vectored Interrupt Controller
- General purpose direct memory access controller
- Power management.