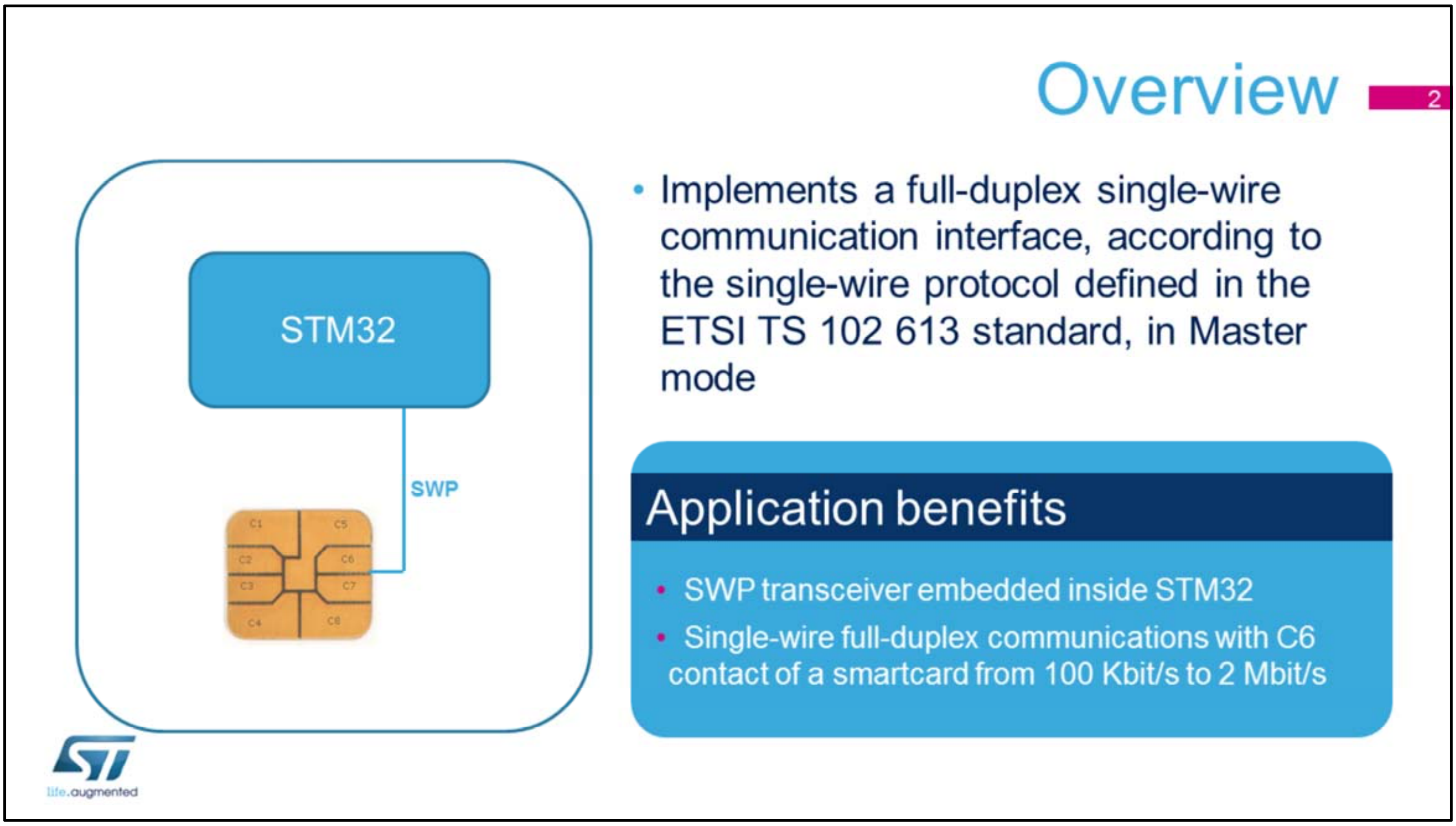
STM32L4 - SWPMI

Single Wire Protocol Master Interface

Revision 2.0

Hello, and welcome to this presentation of the STM32L4 single-wire protocol master interface or SWPMI. It covers the main features of this interface, which is used to connect a smartcard to the microcontroller.



Overview

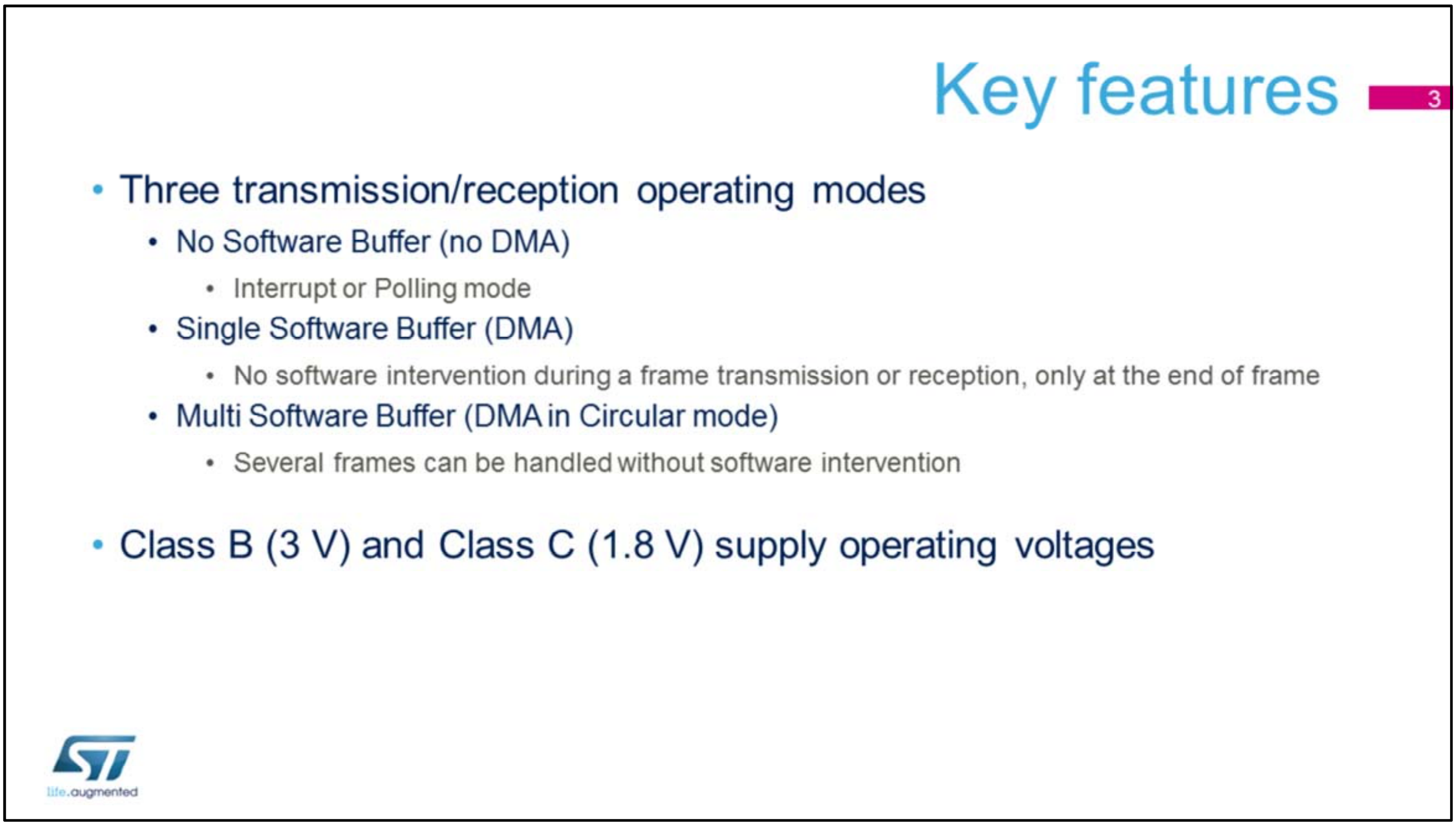
- Implements a full-duplex single-wire communication interface, according to the single-wire protocol defined in the ETSI TS 102 613 standard, in Master mode

Application benefits

- SWP transceiver embedded inside STM32
- Single-wire full-duplex communications with C6 contact of a smartcard from 100 Kbit/s to 2 Mbit/s

The SWPMI integrated inside STM32 products implements a full-duplex single-wire communication interface, in compliance with the single-wire protocol defined in the ETSI TS 102 613 standard, in Master mode.

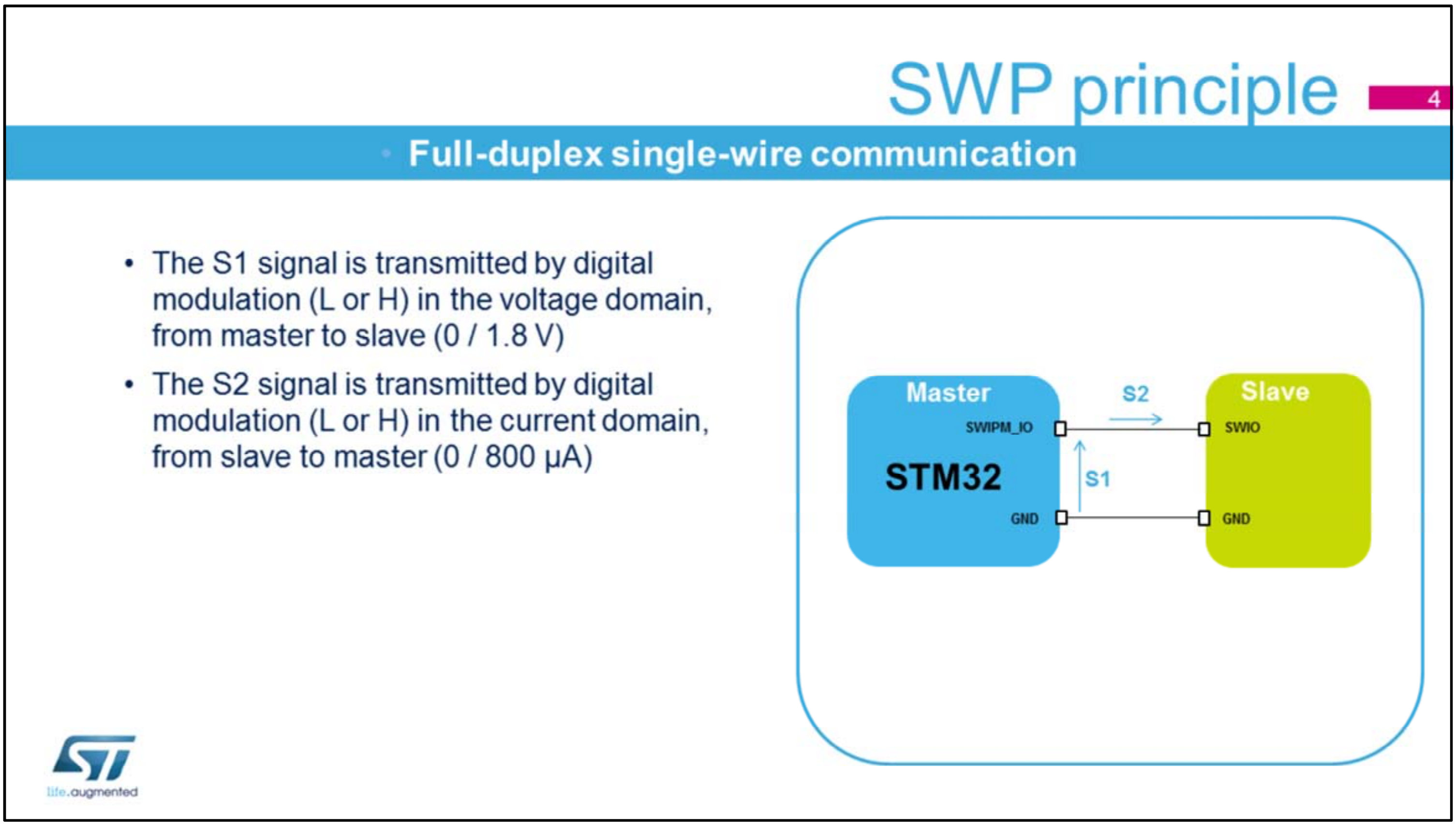
The STM32 embeds the SWP transceiver. Applications benefit from the easy single pin connection to a smartcard for full-duplex communications up to 2 Mbit/s.



Key features

- Three transmission/reception operating modes
  - No Software Buffer (no DMA)
    - Interrupt or Polling mode
  - Single Software Buffer (DMA)
    - No software intervention during a frame transmission or reception, only at the end of frame
  - Multi Software Buffer (DMA in Circular mode)
    - Several frames can be handled without software intervention
- Class B (3 V) and Class C (1.8 V) supply operating voltages

The SWPMI inside STM32 products offers three operating modes, with or without DMA, which are explained in detail later on. The STM32 supports both Class B and Class C supply operating voltages.



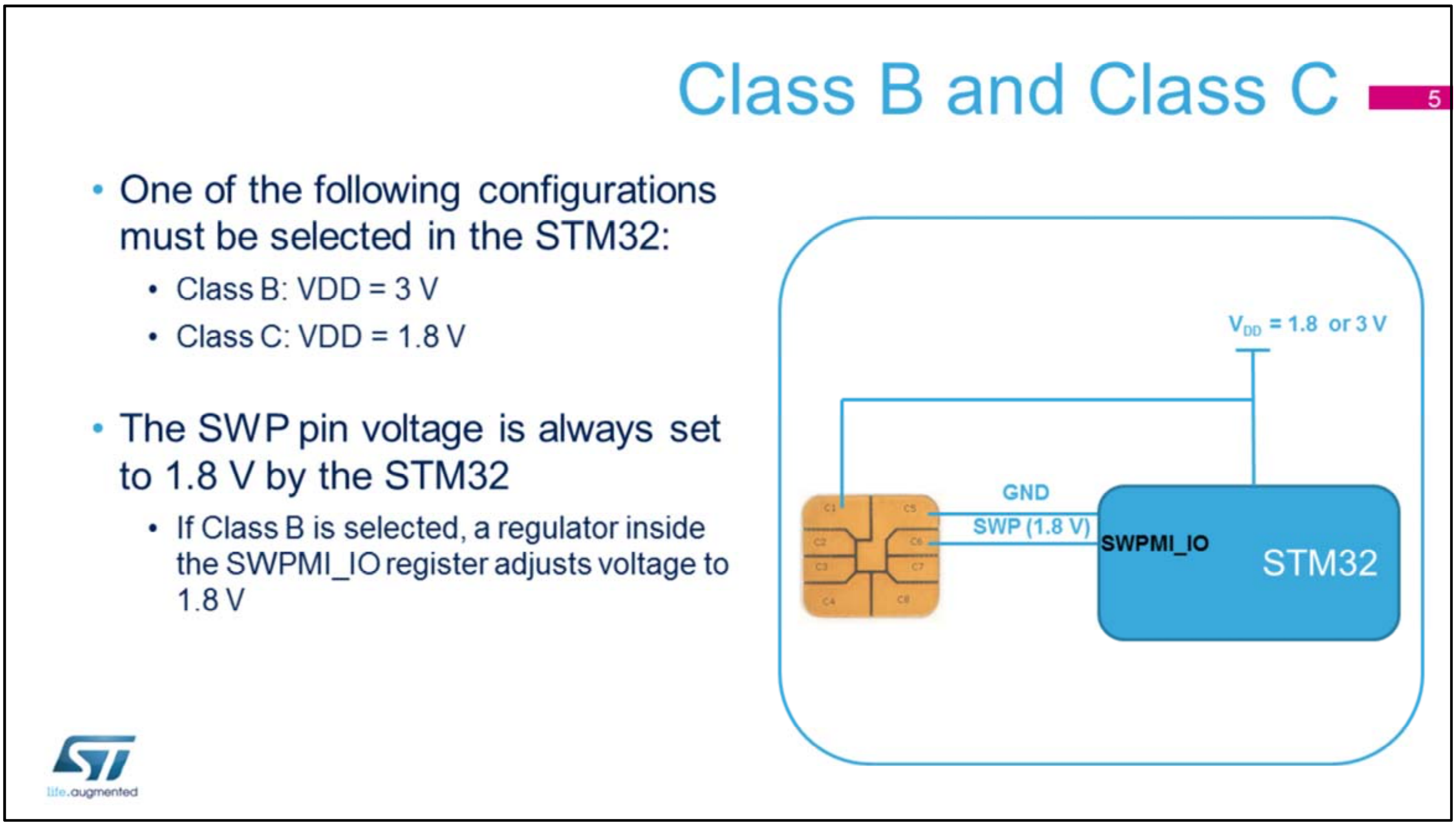
SWP principle

4

Full-duplex single-wire communication

- The S1 signal is transmitted by digital modulation (L or H) in the voltage domain, from master to slave (0 / 1.8 V)
- The S2 signal is transmitted by digital modulation (L or H) in the current domain, from slave to master (0 / 800 µA)

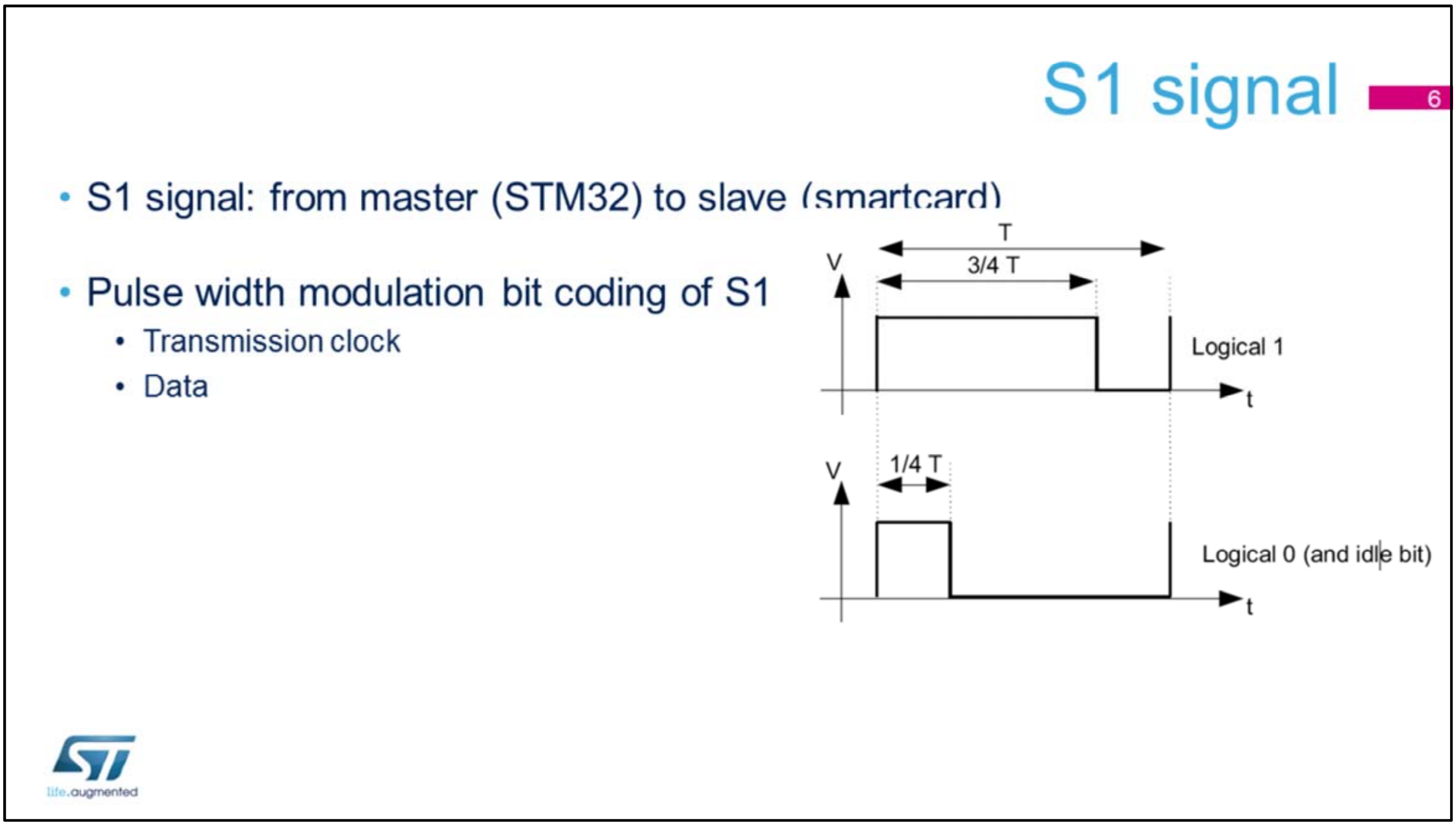
The SWP is full-duplex on a single wire thanks to the following principle. The S1 signal is transmitted in the voltage domain from master to slave. The S2 signal is transmitted in the current domain from slave to master.



Class B and Class C

- One of the following configurations must be selected in the STM32:
  - Class B: VDD = 3 V
  - Class C: VDD = 1.8 V
- The SWP pin voltage is always set to 1.8 V by the STM32
  - If Class B is selected, a regulator inside the SWPMI_IO register adjusts voltage to 1.8 V

The supply voltage or class must be selected in the STM32 during software initialization. a dedicated 1.8V voltage regulator inside the STM32 SWPMI_IO is used to adjust the SWP voltage if VDD is 3V.

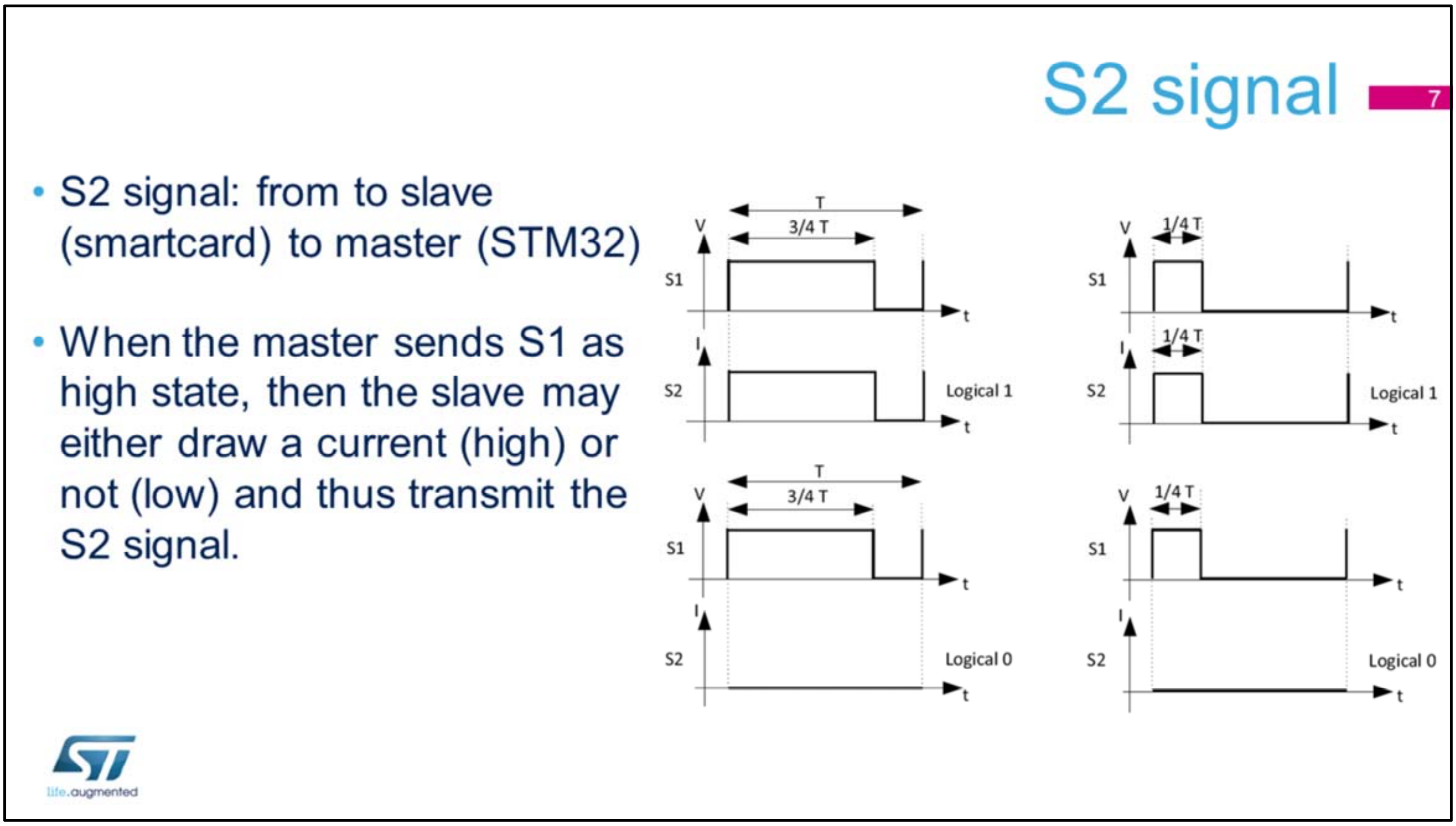


S1 signal

6

- S1 signal: from master (STM32) to slave (smartcard)
- Pulse width modulation bit coding of S1
  - Transmission clock
  - Data

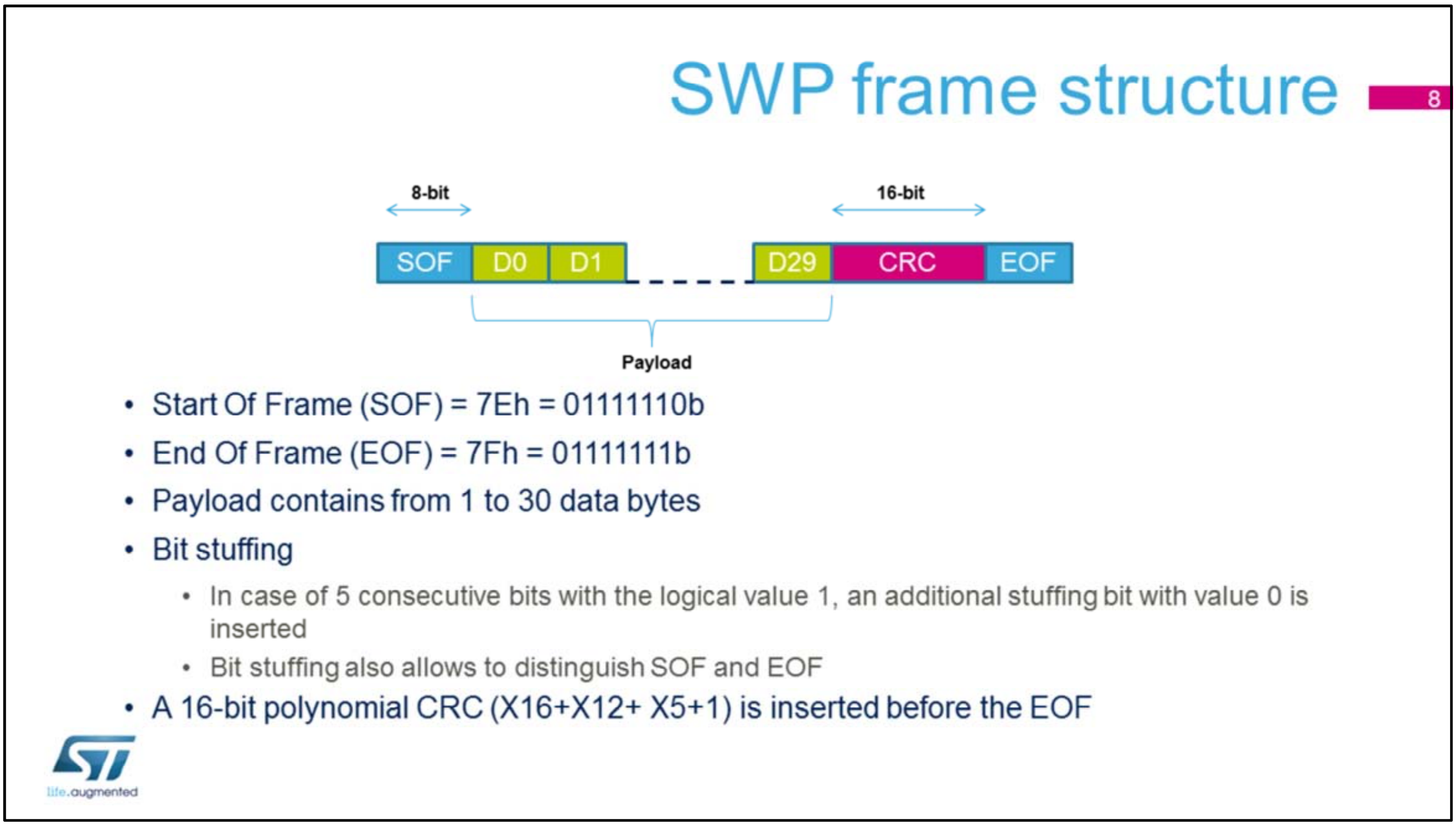
The S1 signal is transmitted by the STM32, the master, to the smartcard, the slave. A duty cycle of 25% on S1 codes a logical 0 (and an idle bit), while a duty cycle of 75% on S1 is codes a logical 1. The S1 signal frequency determines the transmission clock.



S2 signal

- S2 signal: from to slave (smartcard) to master (STM32)
- When the master sends S1 as high state, then the slave may either draw a current (high) or not (low) and thus transmit the S2 signal.

The S2 signal is transmitted by the slave, the smartcard, to the master, the STM32. The slave draws a current while S1 is high to send a logical 1. If the slave does not draw any current while S1 is high, it is a logical 0.



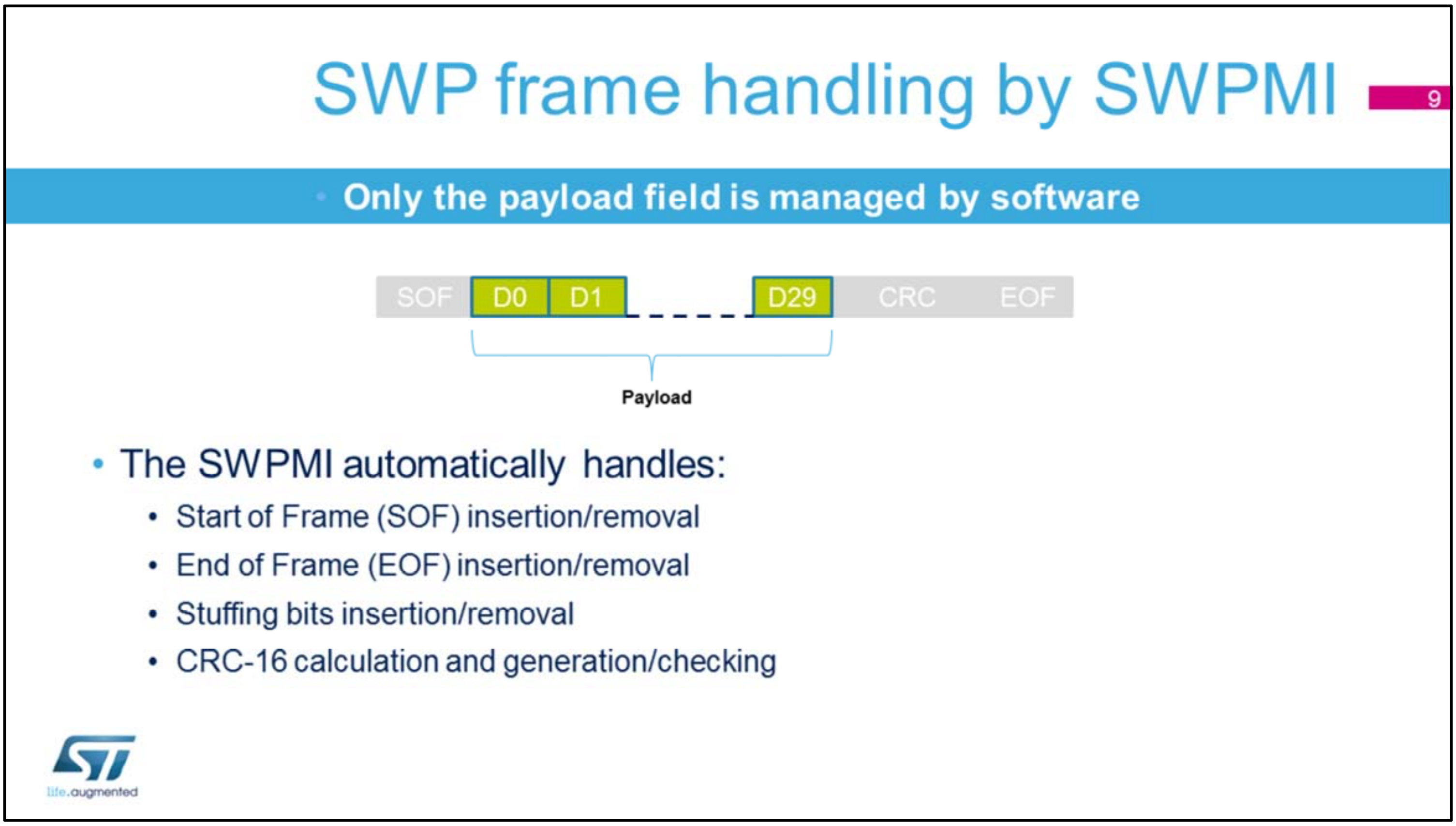
SWP frame structure

8-bit
16-bit
SOF
D0
D1
D29
CRC
EOF
Payload

- Start Of Frame (SOF) = 7Eh = 01111110b
- End Of Frame (EOF) = 7Fh = 01111111b
- Payload contains from 1 to 30 data bytes
- Bit stuffing
  - In case of 5 consecutive bits with the logical value 1, an additional stuffing bit with value 0 is inserted
  - Bit stuffing also allows to distinguish SOF and EOF
- A 16-bit polynomial CRC (X16+X12+ X5+1) is inserted before the EOF

life.augmented

SWP frames start with a Start of Frame field, coded by a 7E byte in hexadecimal format, and ends with an End of Frame field, coded by a 7F byte in hexadecimal format. The payload contains between 1 and 30 bytes of data. The protocol also implements bit stuffing. An extra-bit is inserted in case of 5 consecutive bits at 1. This guarantees that the Start and End of Frame fields are distinguished from the payload bytes. Data integrity is guaranteed by a 16-bit polynomial cyclic redundancy check (CRC).



SWP frame handling by SWPMI

- Only the payload field is managed by software

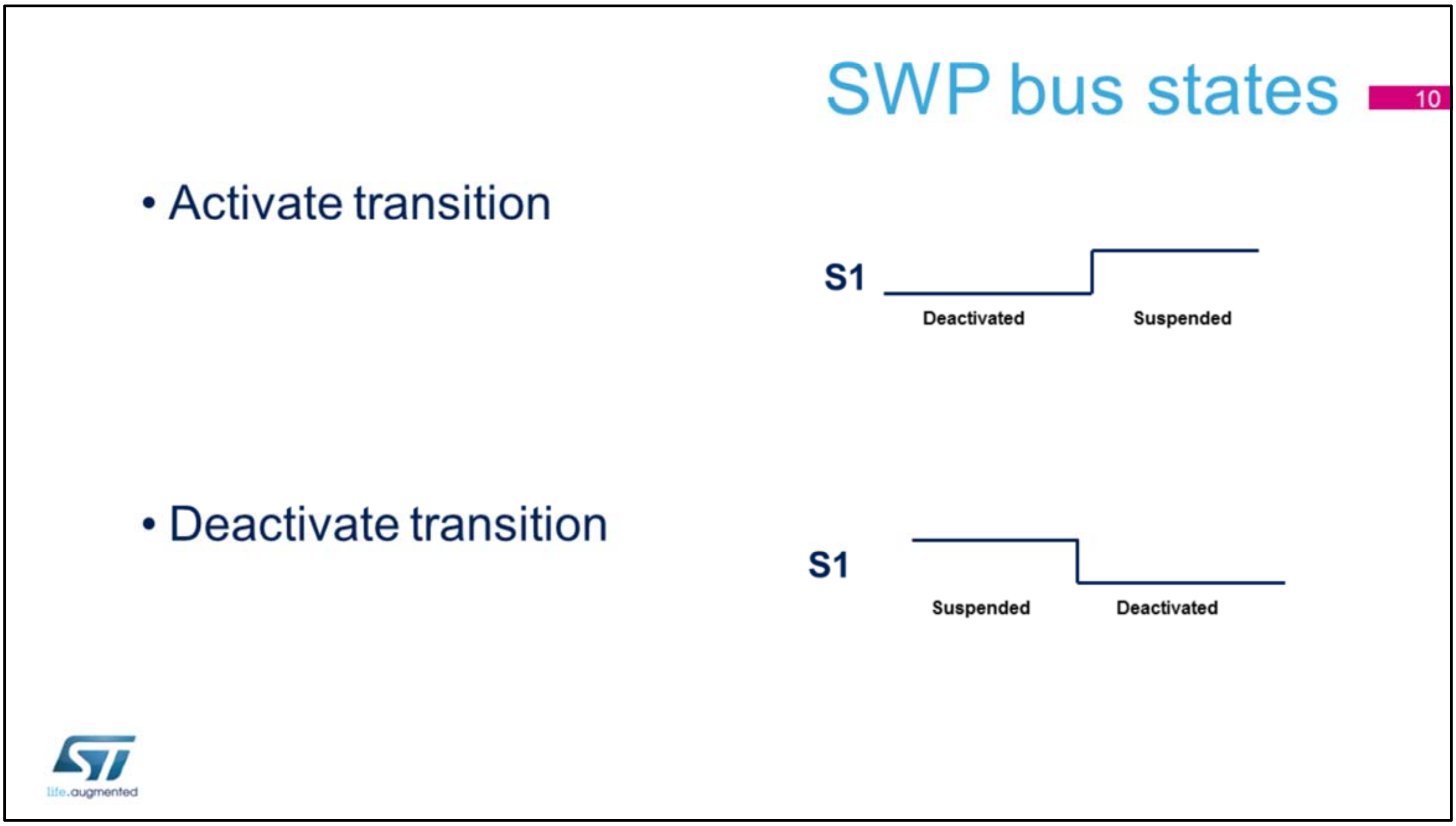
SOF D0 D1 D29 CRC EOF

Payload

- The SWPMI automatically handles:
  - Start of Frame (SOF) insertion/removal
  - End of Frame (EOF) insertion/removal
  - Stuffing bits insertion/removal
  - CRC-16 calculation and generation/checking

life.augmented

The SWPMI automatically handles the Start and End of Frame fields, stuffing bits and the CRC. In this way, software just has to manage payload data.



SWP bus states

- Activate transition

S1

Deactivated

Suspended

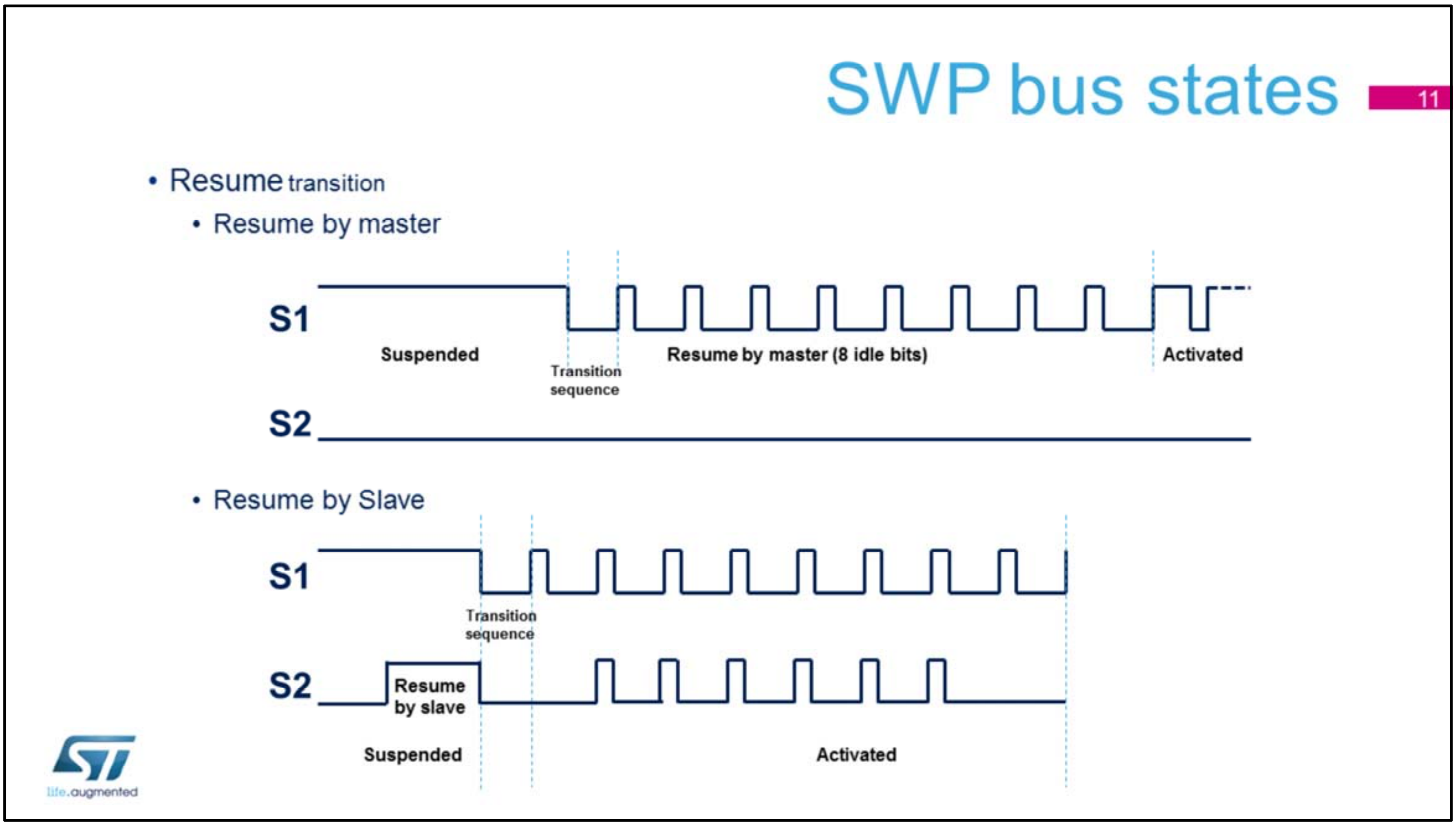
- Deactivate transition

S1

Suspended

Deactivated

Several states are defined for the SWP bus. In Deactivated state, the S1 signal is at low level. Before starting any communication, the master must raise the S1 signal to high level to set the SWP in Suspended state. Once communication is no longer required, the SWP can be deactivated by the master.

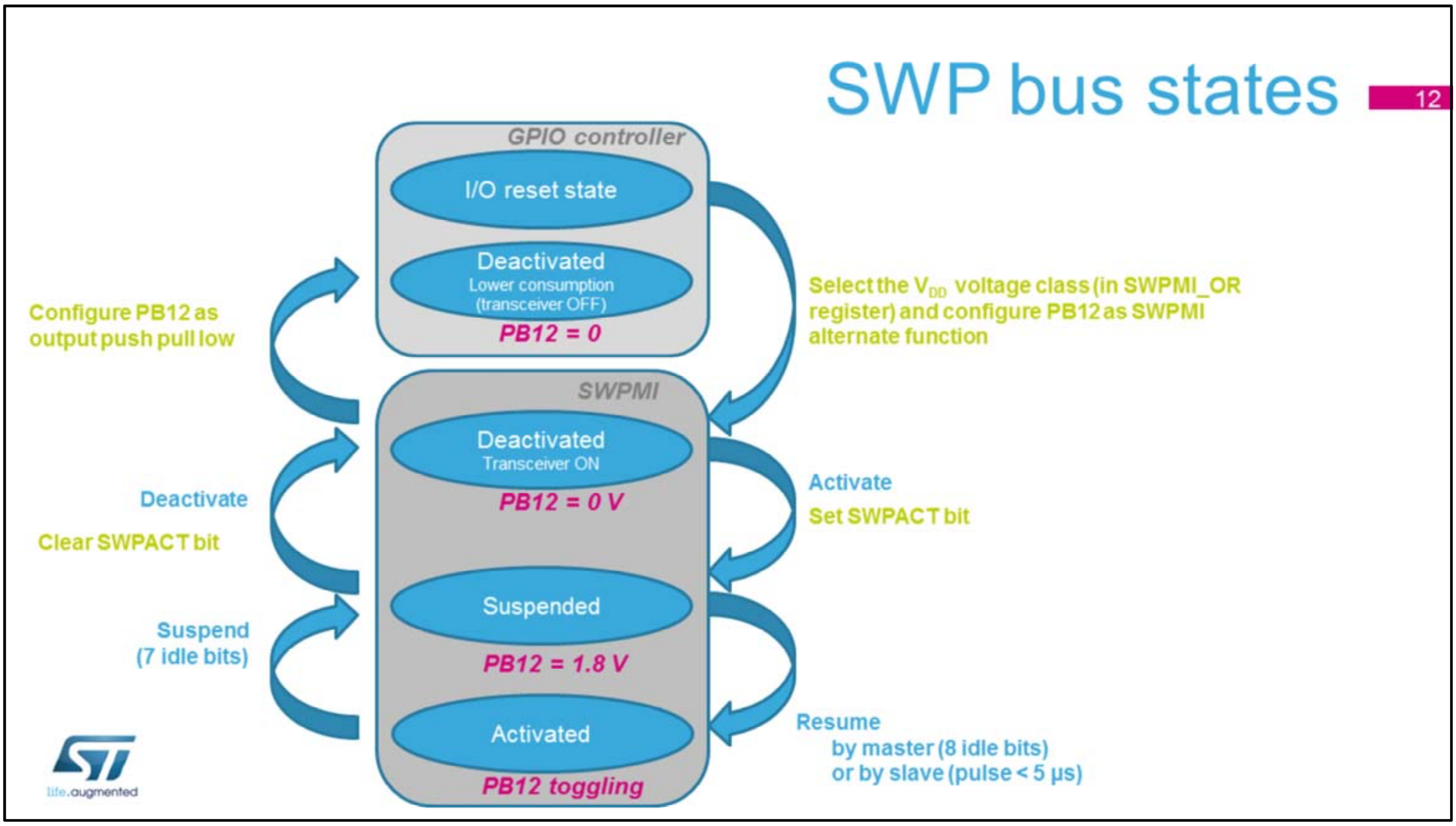


SWP bus states

11

- Resume transition
  - Resume by master
  - Resume by Slave

Now, either the master or the slave can initiate a communication, by sending a Resume sequence. A Resume sequence by the master consists of a transition sequence and 8 idle bits, whereas a Resume signal by the slave consists of drawing current until the master detects it and as a consequence starts to toggle the S1 signal to allow the slave to start transmitting data.



SWP bus states

GPIO controller

I/O reset state

Deactivated
Lower consumption
(transceiver OFF)

PB12 = 0

SWPMI

Deactivated
Transceiver ON

PB12 = 0 V

Suspended

PB12 = 1.8 V

Activated

PB12 toggling

Configure PB12 as output push pull low

Select the V_{DD} voltage class (in SWPMI_OR register) and configure PB12 as SWPMI alternate function

Deactivate

Clear SWPACT bit

Activate

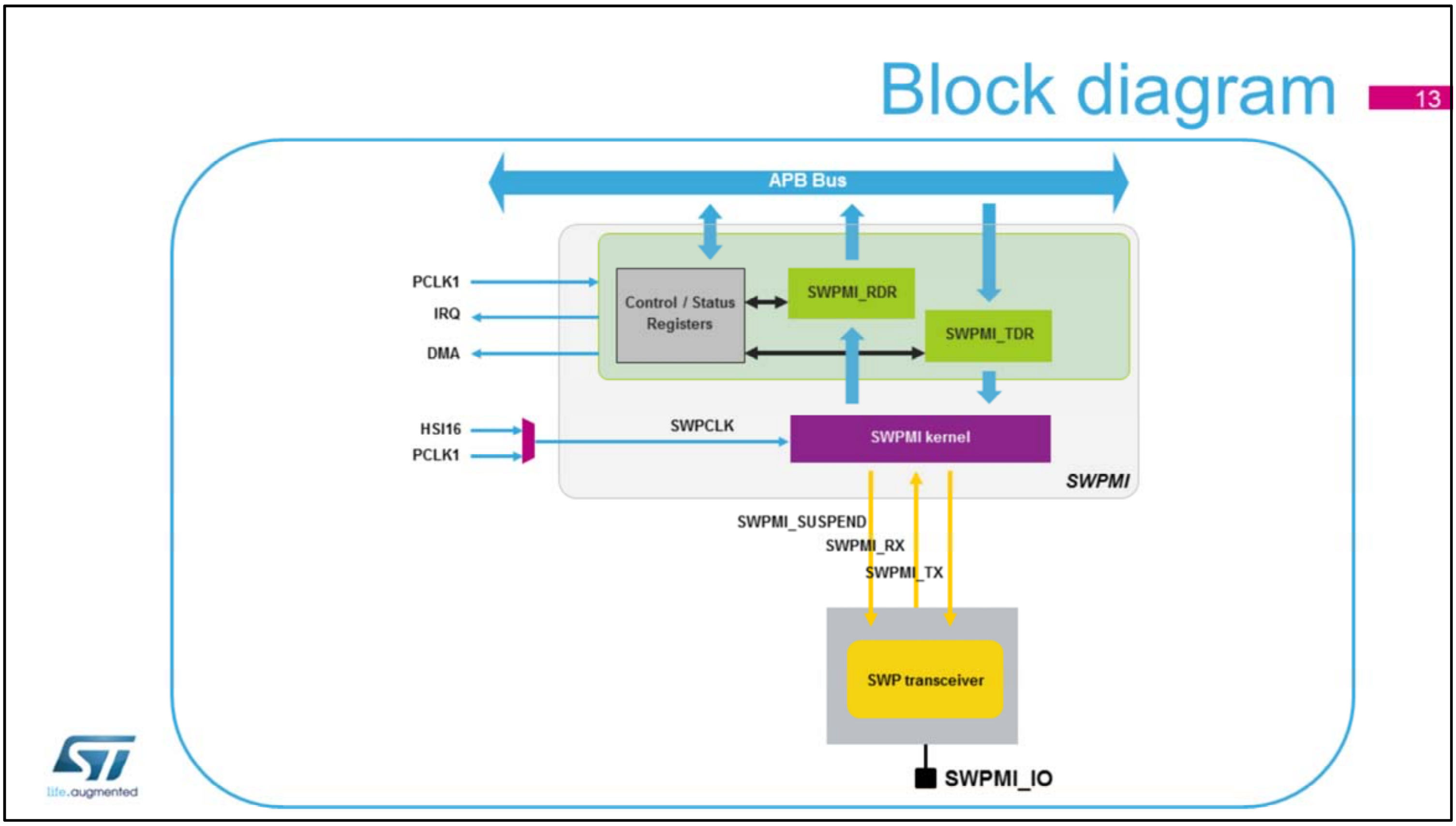
Set SWPACT bit

Suspend
(7 idle bits)

Resume
by master (8 idle bits)
or by slave (pulse < 5 µs)

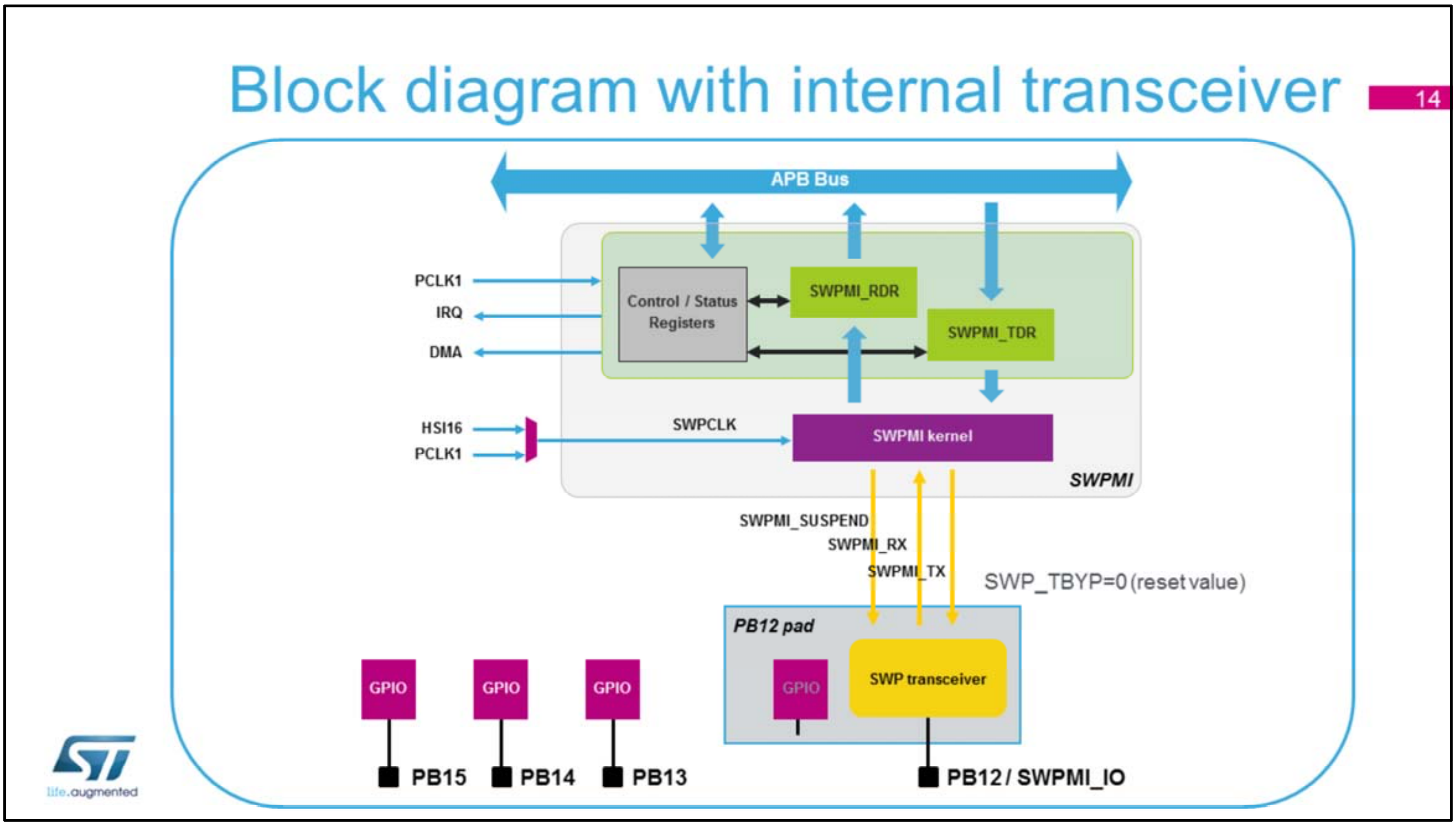
life.augmented

Here is an overview of how the SWP bus states are managed by the STM32. You can refer to the reference manual for more details about the initialization and activation procedures.



Block diagram

Here is the block diagram of the SWPMI peripheral. The kernel part is clocked either by the **HSI16**, internal RC oscillator, or by PCLK1, which is the APB bus clock. The interface with the APB bus allows access to the SWPMI registers by the CPU. There are also connections to the NVIC and the DMA. The SWP transceiver is embedded in the STM32 which interfaces with the external pin through the SWPMI_IO signal.



Block diagram with internal transceiver

APB Bus

PCLK1

IRQ

DMA

Control / Status Registers

SWPMI_RDR

SWPMI_TDR

HSI16

PCLK1

SWPCLK

SWPMI kernel

SWPMI

SWPMI_SUSPEND

SWPMI_RX

SWPMI_TX

SWP_TBYP=0 (reset value)

PB12 pad

GPIO

SWP transceiver

GPIO

GPIO

GPIO

PB15

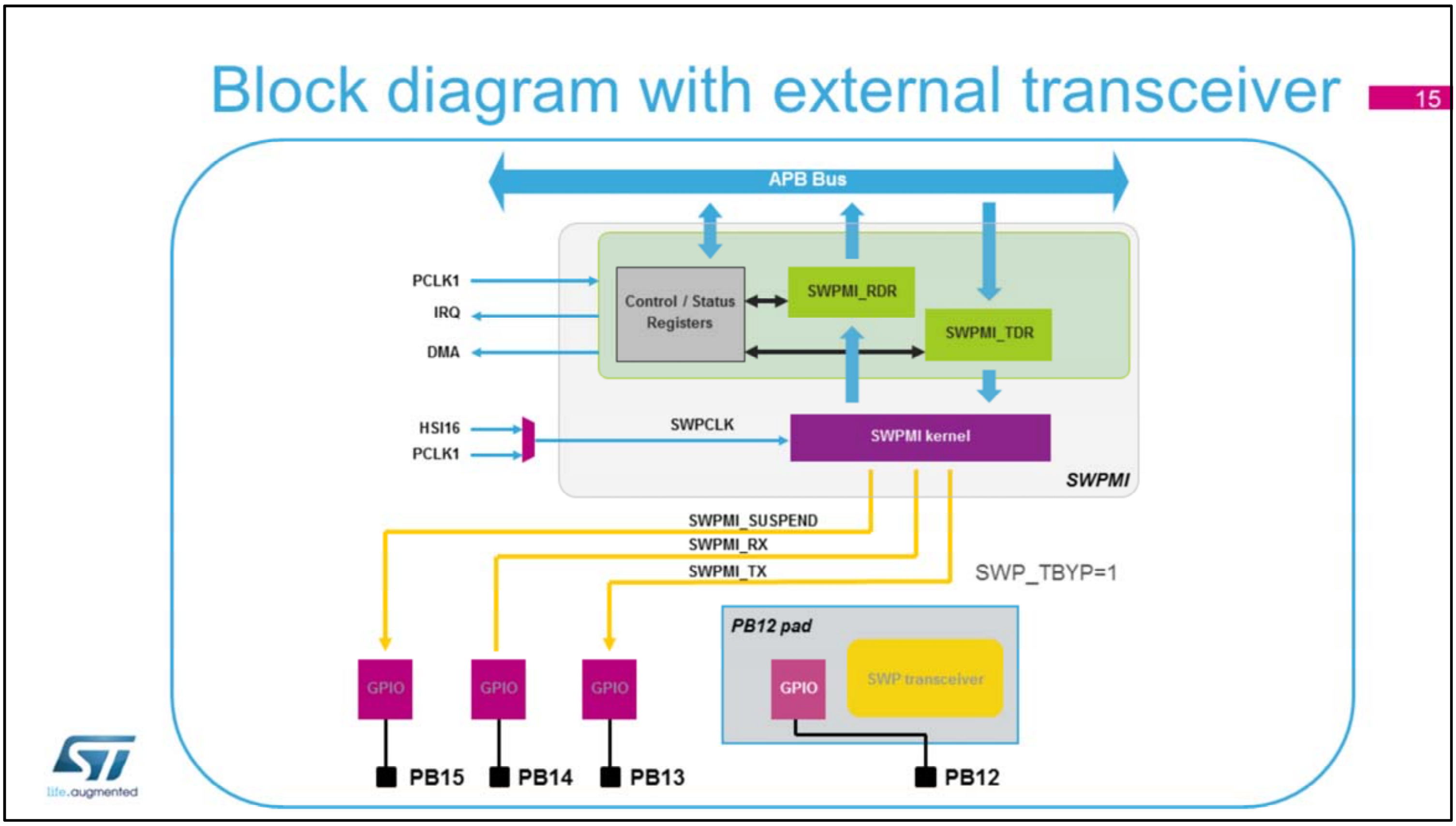
PB14

PB13

PB12 / SWPMI_IO

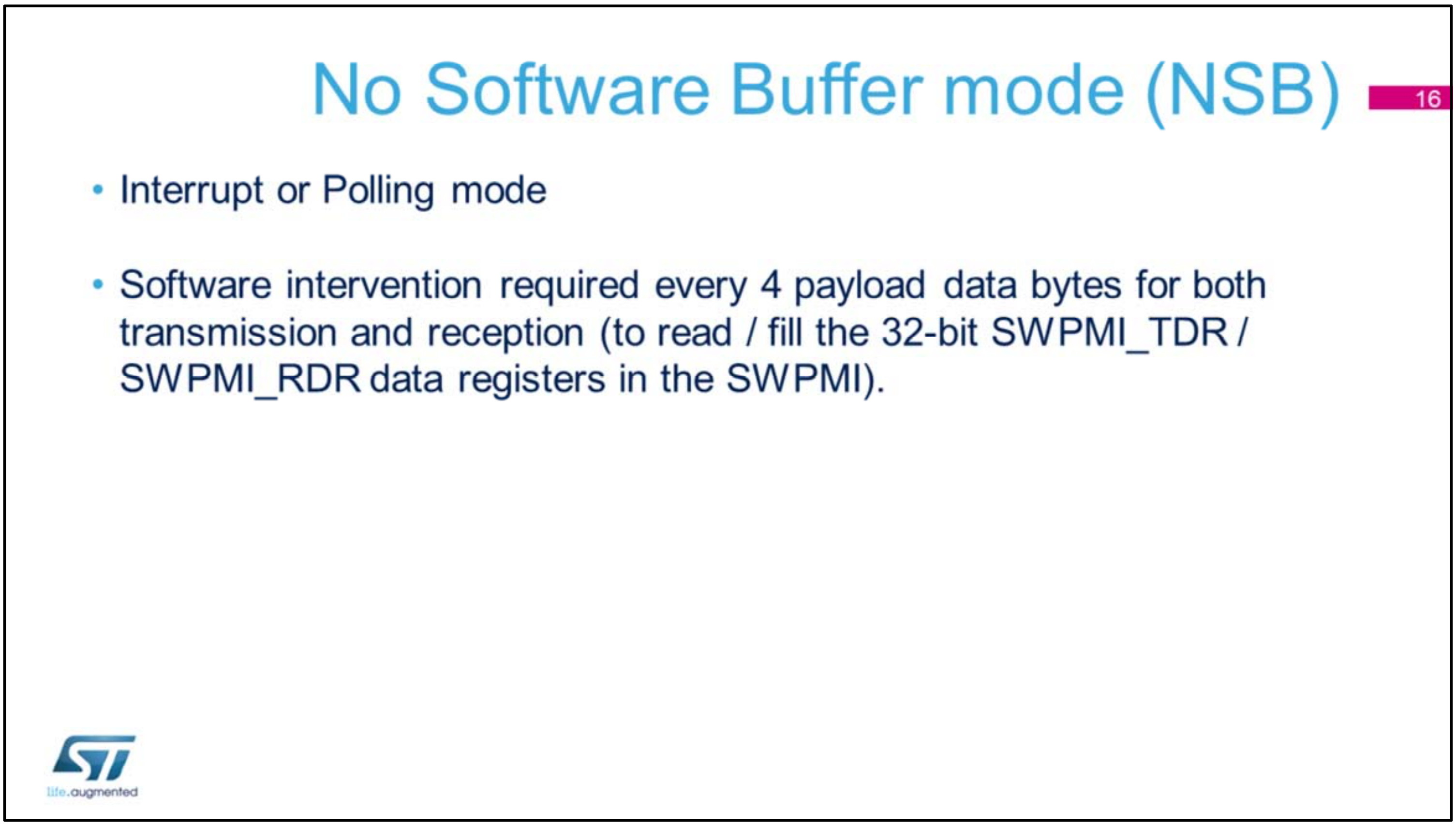
life.augmented

Here is the default configuration using the internal transceiver. The SWPMI_IO signal is available on the PB12 pin.



Block diagram with external transceiver

It’s also possible to connect an external transceiver using a configuration bit in the SWPMI registers. In this case, the Suspend, Receive and Transmit signals are available on pins PB15, PB14 and PB13. Pin PB12 can then be used as a standard GPIO.



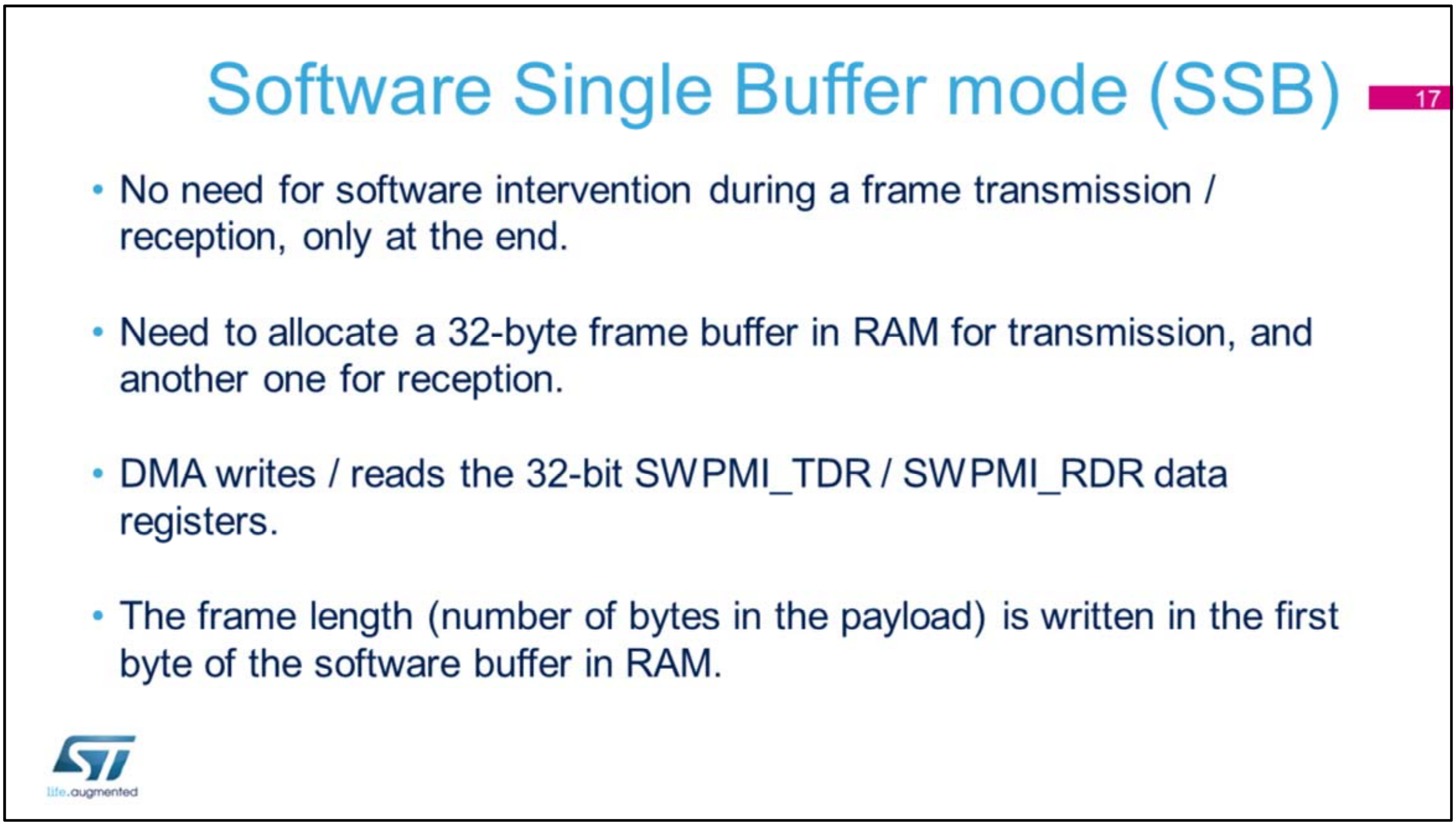
No Software Buffer mode (NSB)

16

- Interrupt or Polling mode
- Software intervention required every 4 payload data bytes for both transmission and reception (to read / fill the 32-bit SWPMI_TDR / SWPMI_RDR data registers in the SWPMI).

life.augmented

Let’s look at the different operating modes, starting with No Software Buffer mode (NSB). In this mode, data is received and transmitted in Polling or Interrupt mode or by checking the SWPMI flags. Software intervention is required each time the Receive Data register becomes full or when the Transmit Data register becomes empty; that is to say, every 4 data bytes in the payload.



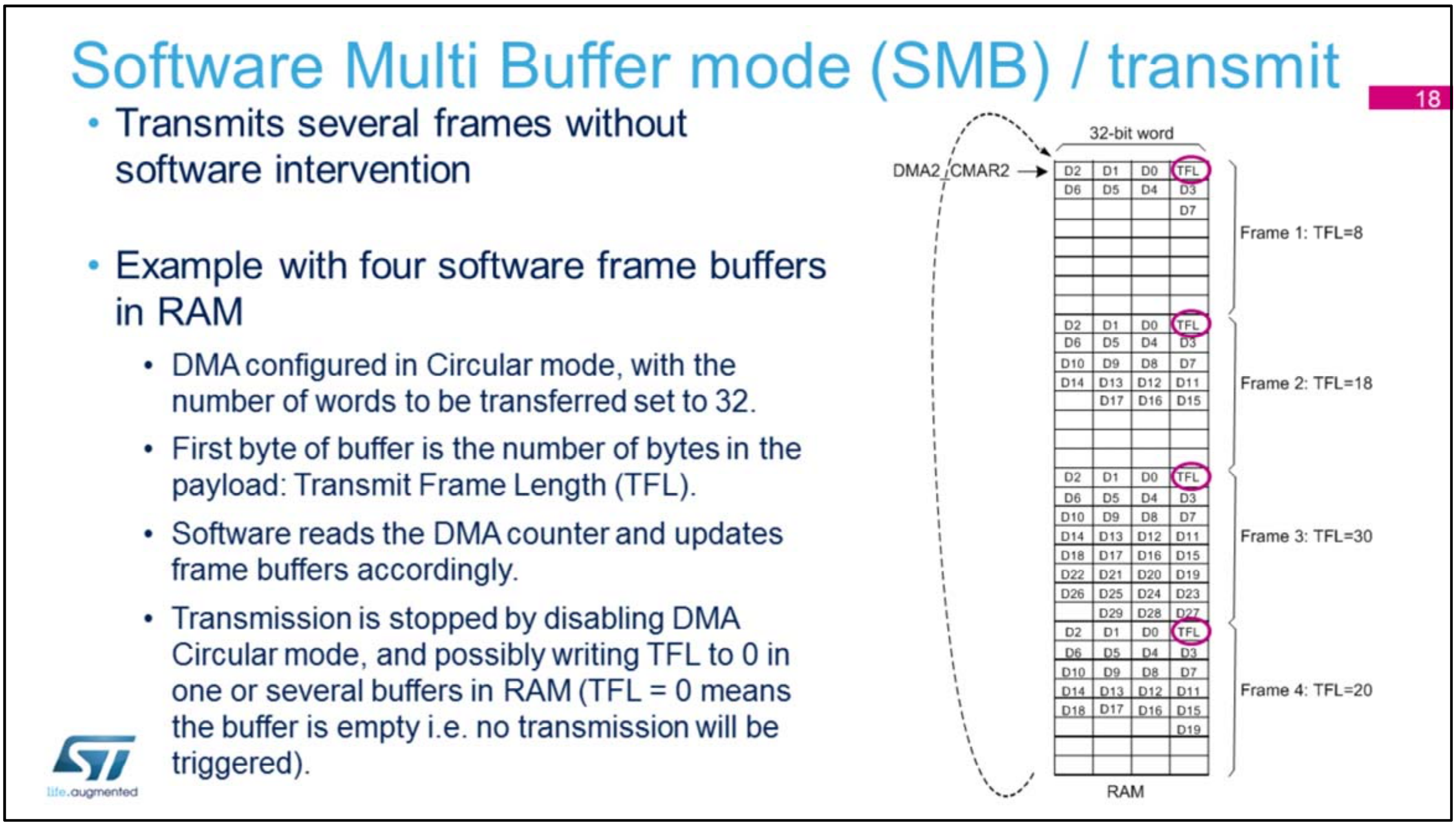
Software Single Buffer mode (SSB)

17

- No need for software intervention during a frame transmission / reception, only at the end.
- Need to allocate a 32-byte frame buffer in RAM for transmission, and another one for reception.
- DMA writes / reads the 32-bit SWPMI_TDR / SWPMI_RDR data registers.
- The frame length (number of bytes in the payload) is written in the first byte of the software buffer in RAM.

life.augmented

Software Single Buffer mode (SSB) is used to transmit or receive an entire SWP frame without software intervention. A 32-byte software buffer for frame transmission is defined in RAM, and the SWPMI automatically reloads the SWPMI_TDR register through the DMA until the End of Frame is received. For reception, a 32-byte software buffer is defined in RAM for frame reception, and the SWPMI_RDR register content is transferred to the RAM by the DMA. The first byte in the RAM buffer is used to code the number of bytes in the frame payload.

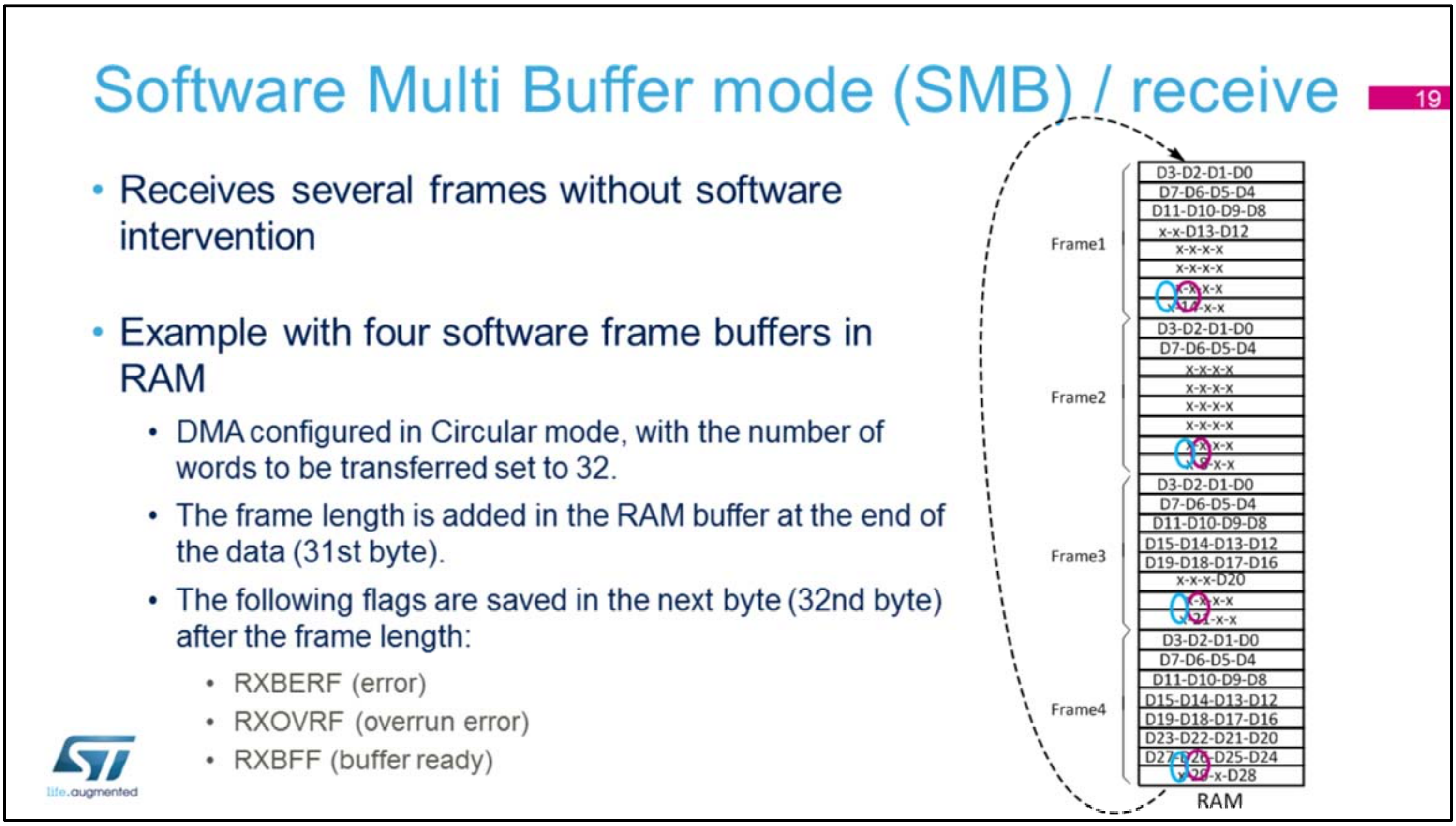


Software Multi Buffer mode (SMB) / transmit

- Transmits several frames without software intervention
- Example with four software frame buffers in RAM
  - DMA configured in Circular mode, with the number of words to be transferred set to 32.
  - First byte of buffer is the number of bytes in the payload: Transmit Frame Length (TFL).
  - Software reads the DMA counter and updates frame buffers accordingly.
  - Transmission is stopped by disabling DMA Circular mode, and possibly writing TFL to 0 in one or several buffers in RAM (TFL = 0 means the buffer is empty i.e. no transmission will be triggered).

D2	D1	D0	TFL
D6	D5	D4	D3
			D7
D2	D1	D0	TFL
D6	D5	D4	D3
D10	D9	D8	D7
D14	D13	D12	D11
	D17	D16	D15
D2	D1	D0	TFL
D6	D5	D4	D3
D10	D9	D8	D7
D14	D13	D12	D11
D18	D17	D16	D15
D22	D21	D20	D19
D26	D25	D24	D23
	D29	D28	D27
D2	D1	D0	TFL
D6	D5	D4	D3
D10	D9	D8	D7
D14	D13	D12	D11
D18	D17	D16	D15
			D19

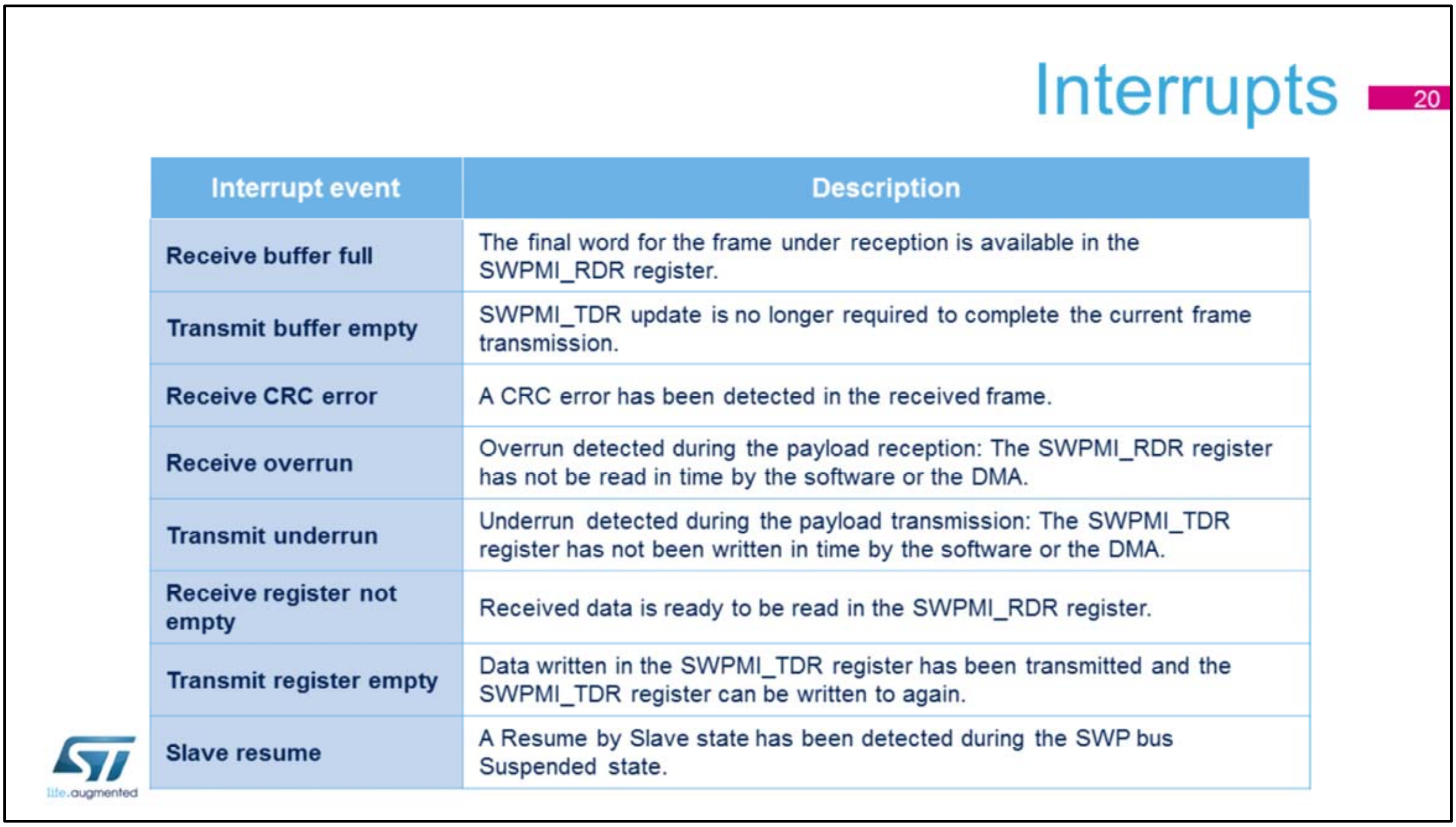
The last mode is Software Multi-Buffer mode (SMB). This mode also uses the DMA, and several SWP frames can be handled without software intervention. Let's look at this example of a transmission, with 4 frame buffers in RAM. In this mode, 32 bytes are always reserved for each frame, regardless of the payload size. The DMA must be configured in Circular mode, and the number of words to be transferred must be set to 32. As in SSB mode, the first byte of each buffer is used to code the frame length (this is the TFL field). Software can read the DMA counter and update each frame buffer accordingly. In this example, three frames can be transmitted without software intervention. The transmission is stopped by disabling DMA Circular mode.
In case you need to stop transmission before the DMA end of count, you must set the TFL field to 0. This way, the SWPMI will no longer issue any DMA requests.



Software Multi Buffer mode (SMB) / receive

- Receives several frames without software intervention
- Example with four software frame buffers in RAM
  - DMA configured in Circular mode, with the number of words to be transferred set to 32.
  - The frame length is added in the RAM buffer at the end of the data (31st byte).
  - The following flags are saved in the next byte (32nd byte) after the frame length:
    - RXBERF (error)
    - RXOVRF (overrun error)
    - RXBFF (buffer ready)

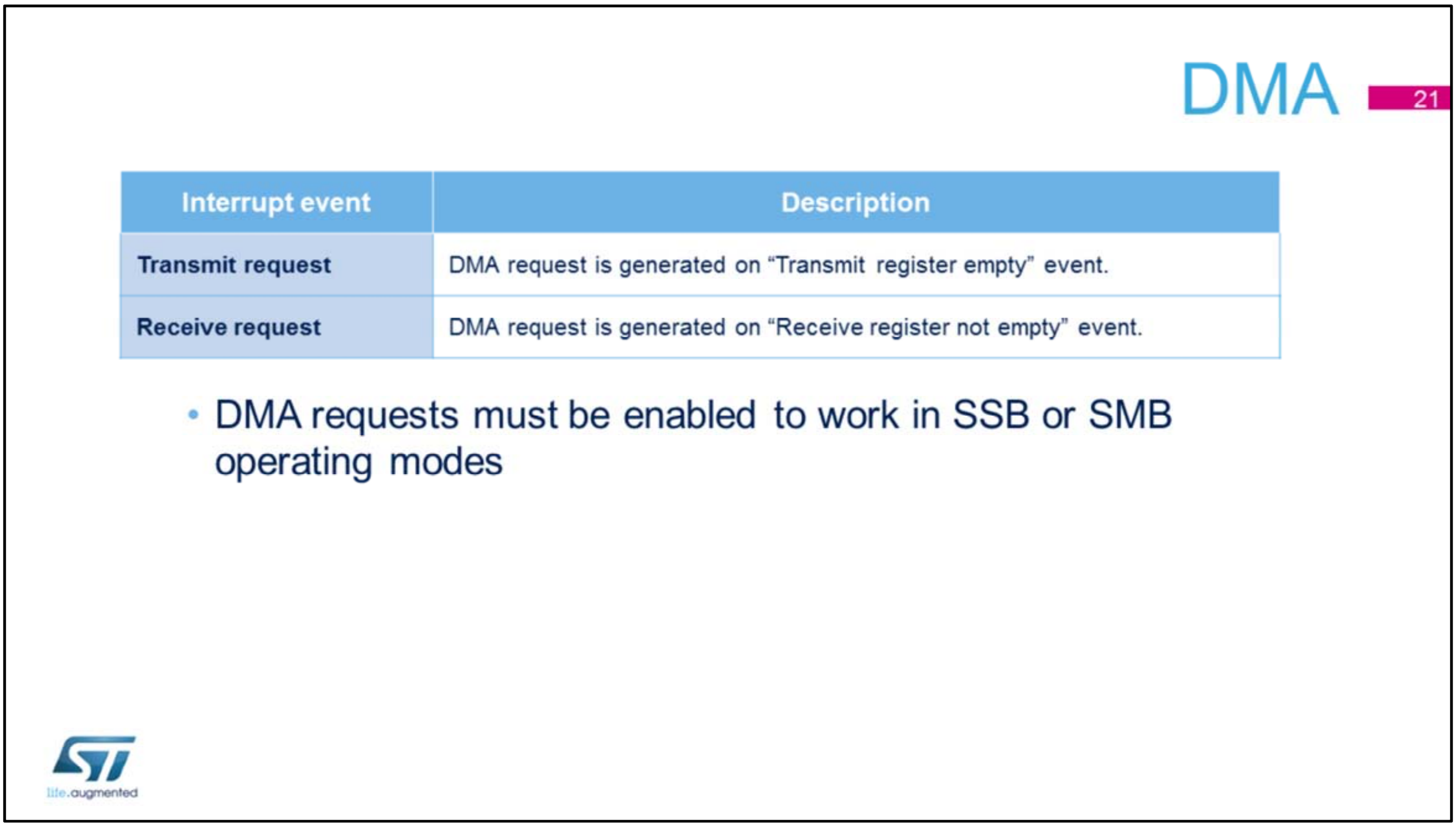
In SMB mode, several frames can be received without software intervention. Let’s look at this example for reception, with four frame buffers in RAM. In this mode, the DMA must be configured in Circular mode, and the number of words to be transferred must be set to 32. The frame length is available at the end of each software buffer, in the 31th byte. The status of the frame stored in each software buffer is available in the 32nd byte which contains the error, overrun and buffer ready flags. This way, software can check the buffer ready flag, read the buffer and clear the 32nd byte.



Interrupts

Interrupt event	Description
Receive buffer full	The final word for the frame under reception is available in the SWPMI_RDR register.
Transmit buffer empty	SWPMI_TDR update is no longer required to complete the current frame transmission.
Receive CRC error	A CRC error has been detected in the received frame.
Receive overrun	Overrun detected during the payload reception: The SWPMI_RDR register has not be read in time by the software or the DMA.
Transmit underrun	Underrun detected during the payload transmission: The SWPMI_TDR register has not been written in time by the software or the DMA.
Receive register not empty	Received data is ready to be read in the SWPMI_RDR register.
Transmit register empty	Data written in the SWPMI_TDR register has been transmitted and the SWPMI_TDR register can be written to again.
Slave resume	A Resume by Slave state has been detected during the SWP bus Suspended state.

Here is a summary of the events able to trigger an interrupt in the NVIC controller: Transmit and Receive buffers, Transmit and Receive registers, errors (CRC, overrun and underrun), and Resume by Slave.

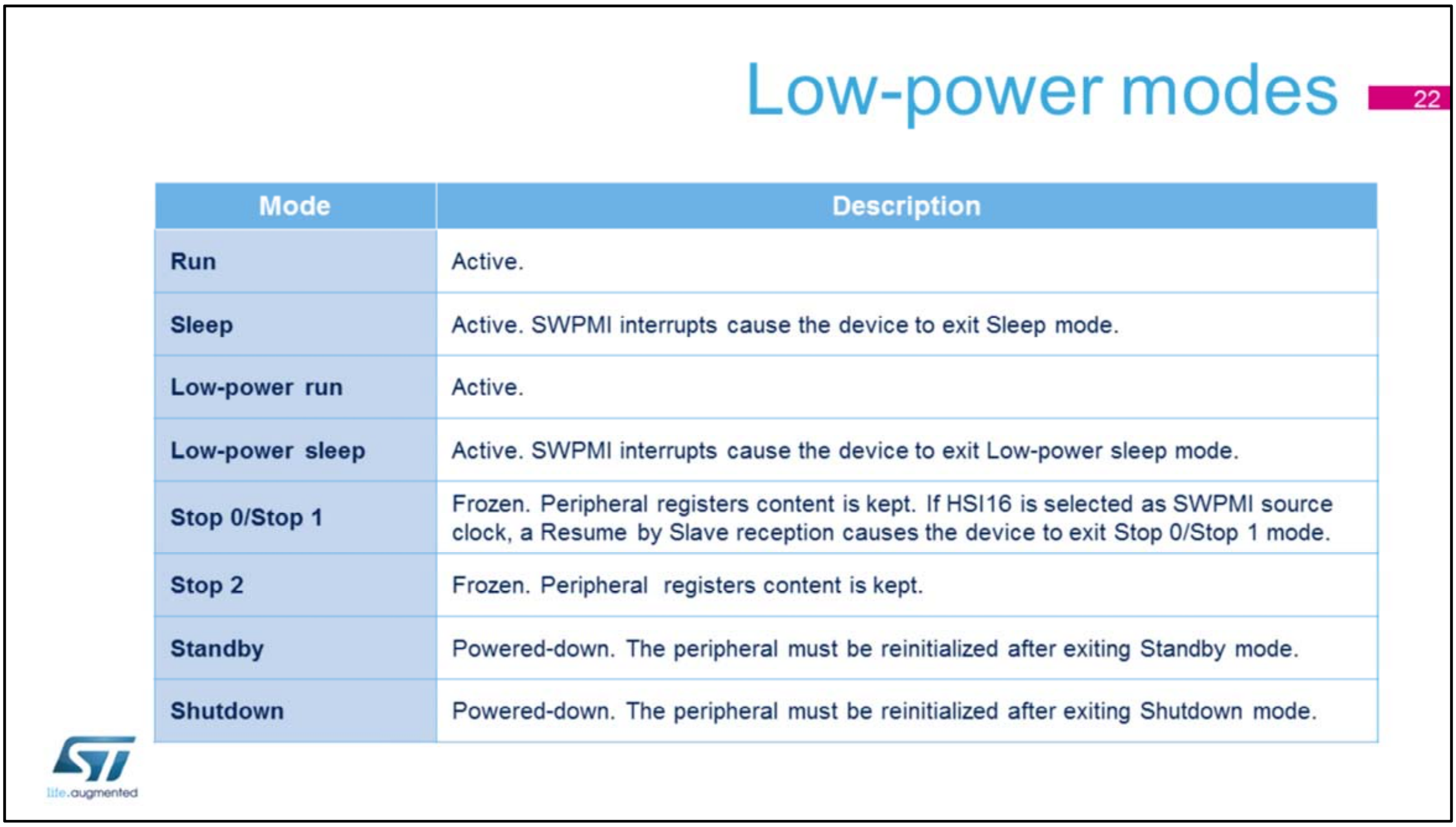


DMA

Interrupt event	Description
Transmit request	DMA request is generated on “Transmit register empty” event.
Receive request	DMA request is generated on “Receive register not empty” event.

- DMA requests must be enabled to work in SSB or SMB operating modes

DMA requests are generated by the SWPMI for transmission and reception. They must be enabled when working in SSB and SMB modes.

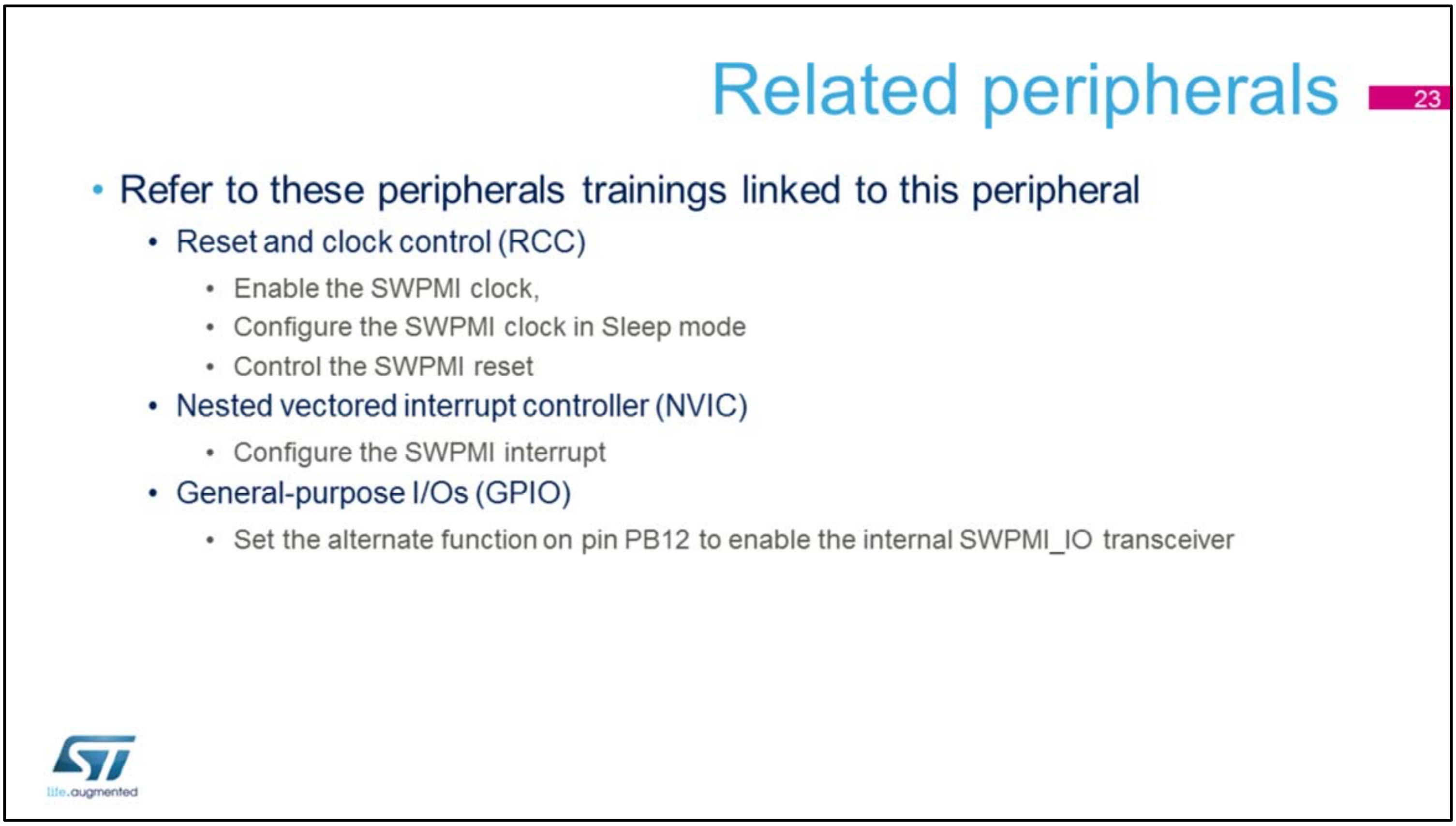


Low-power modes

Mode	Description
Run	Active.
Sleep	Active. SWPMI interrupts cause the device to exit Sleep mode.
Low-power run	Active.
Low-power sleep	Active. SWPMI interrupts cause the device to exit Low-power sleep mode.
Stop 0/Stop 1	Frozen. Peripheral registers content is kept. If HSI16 is selected as SWPMI source clock, a Resume by Slave reception causes the device to exit Stop 0/Stop 1 mode.
Stop 2	Frozen. Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.
Shutdown	Powered-down. The peripheral must be reinitialized after exiting Shutdown mode.

life.augmented

All SWPMI interrupts can wake up the device from Sleep mode. If the device is put in Stop mode, only a Resume by Slave event can wake up the device.



Related peripherals

23

- Refer to these peripherals trainings linked to this peripheral
  - Reset and clock control (RCC)
    - Enable the SWPMI clock,
    - Configure the SWPMI clock in Sleep mode
    - Control the SWPMI reset
  - Nested vectored interrupt controller (NVIC)
    - Configure the SWPMI interrupt
  - General-purpose I/Os (GPIO)
    - Set the alternate function on pin PB12 to enable the internal SWPMI_IO transceiver

This is a list of peripherals related to the single-wire protocol master interface. Please refer to these peripheral trainings for more information if needed.