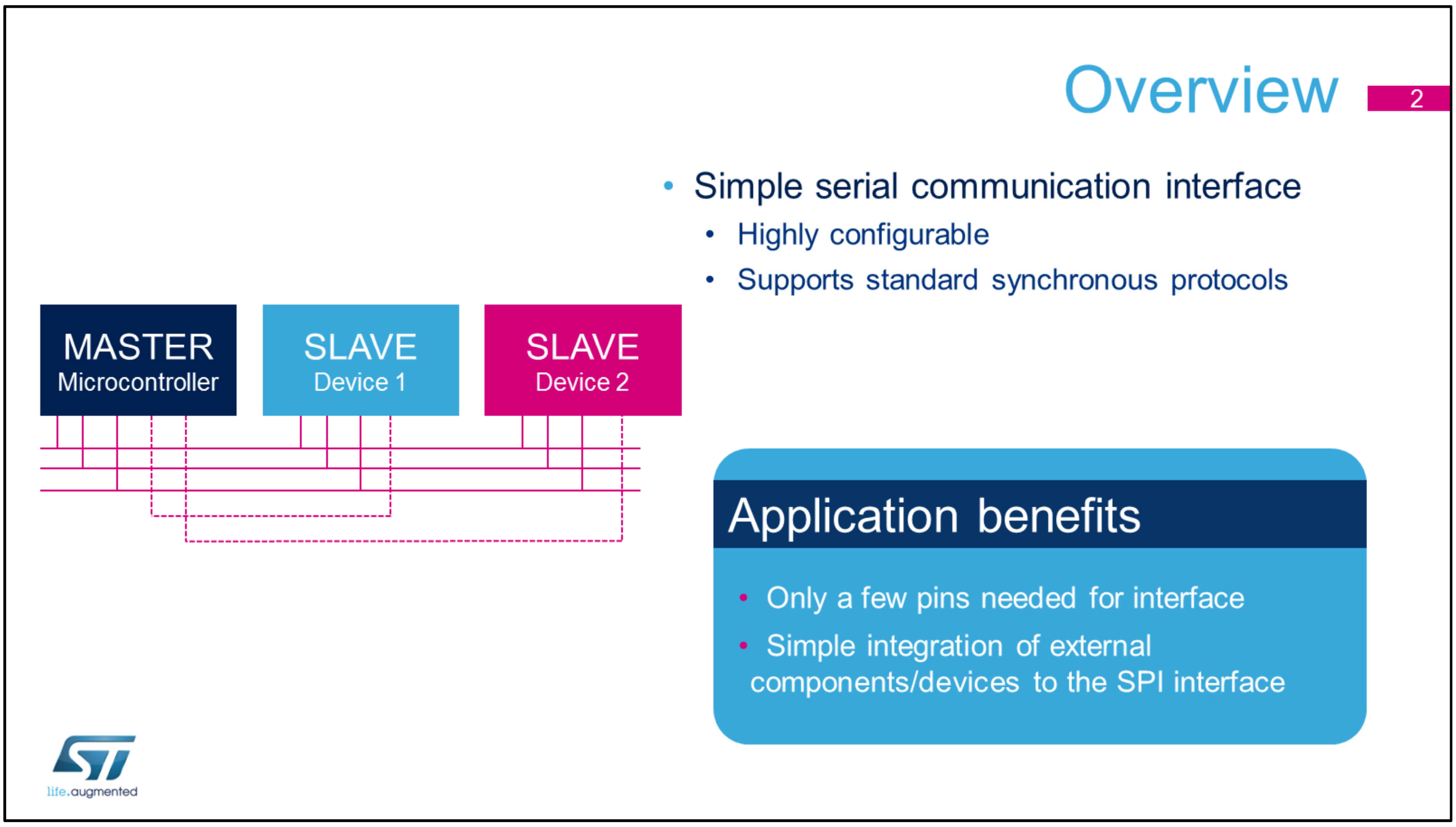
# STM32H7 - SPI

Serial Peripheral Interface

Revision 1.0

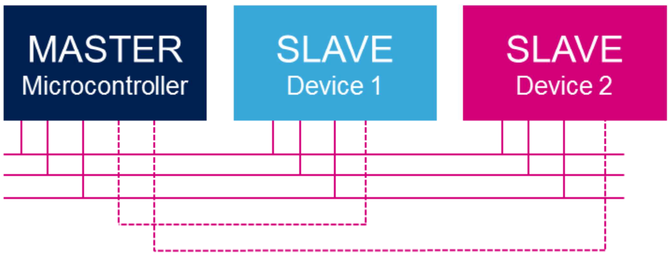
life.augmented

Hello, and welcome to this presentation of the STM32 Serial Peripheral Interface.



## Overview

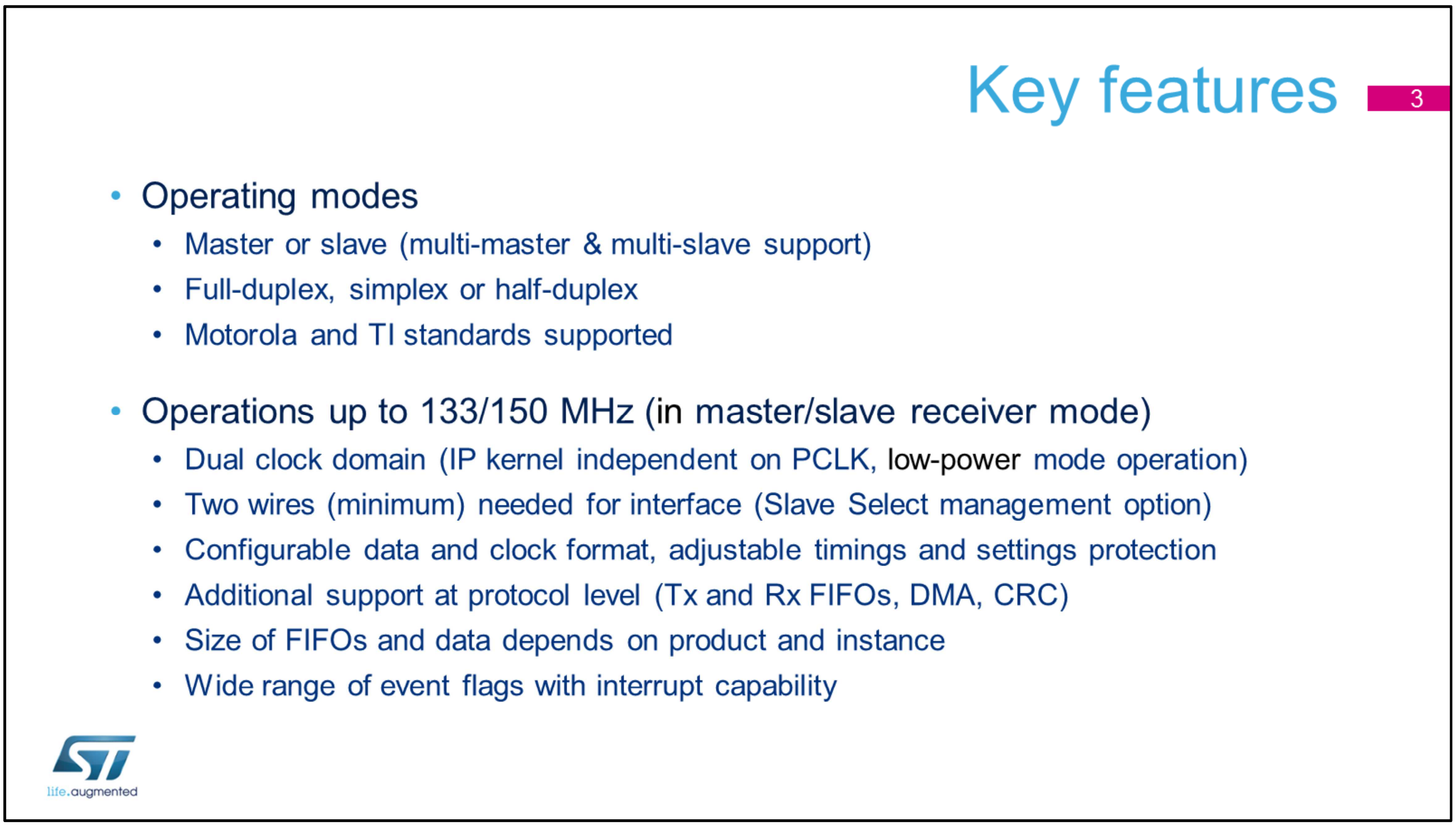
- Simple serial communication interface
  - Highly configurable
  - Supports standard synchronous protocols



### Application benefits

- Only a few pins needed for interface
- Simple integration of external components/devices to the SPI interface

The internal Serial Peripheral Interface or SPI provides a simple communication interface allowing the microcontroller to communicate with external devices. This interface is highly configurable to support many standard protocols. Applications benefit from the simple and direct connection to components which only requires a few pins. Thanks to the highly configurable capabilities of the SPI, many devices can be simply accommodated in the existing project.


## Key features

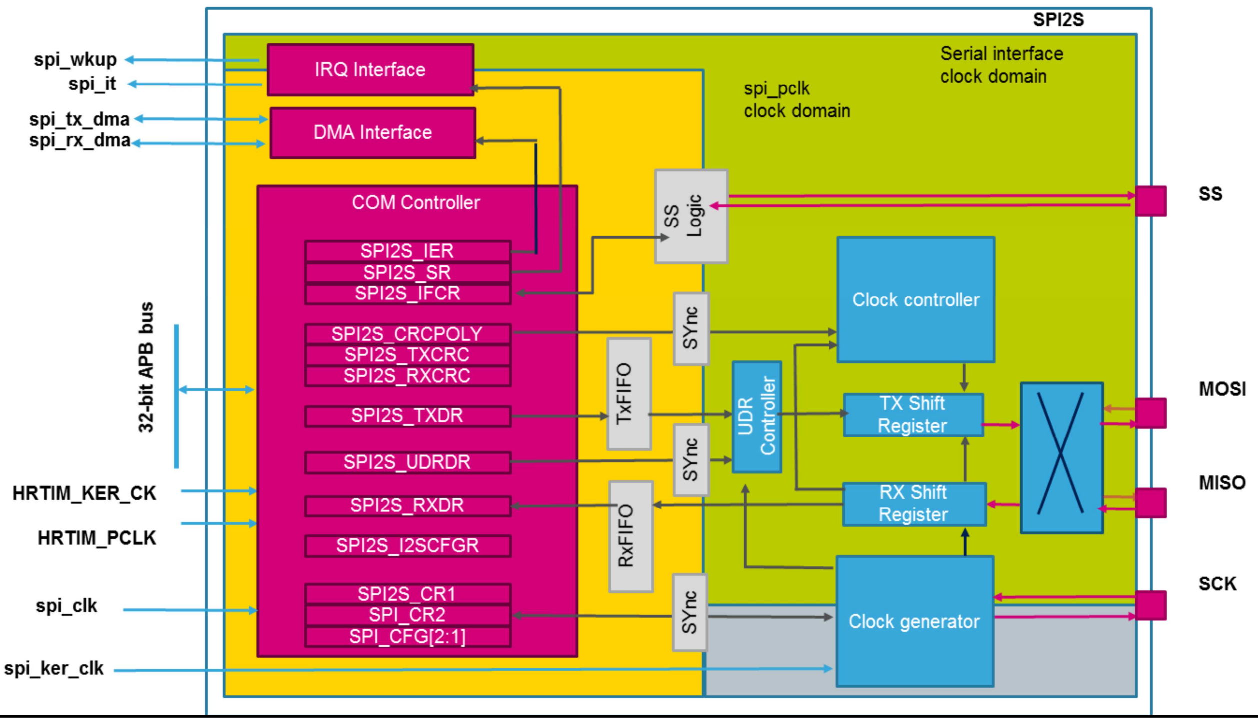
- Operating modes
  - Master or slave (multi-master & multi-slave support)
  - Full-duplex, simplex or half-duplex
  - Motorola and TI standards supported
- Operations up to 133/150 MHz (in master/slave receiver mode)
  - Dual clock domain (IP kernel independent on PCLK, low-power mode operation)
  - Two wires (minimum) needed for interface (Slave Select management option)
  - Configurable data and clock format, adjustable timings and settings protection
  - Additional support at protocol level (Tx and Rx FIFOs, DMA, CRC)
  - Size of FIFOs and data depends on product and instance
  - Wide range of event flags with interrupt capability

life.augmented

The STM32 SPI offers various operating modes that are explained in more detail in this presentation.
The communication speed can't exceed half of the internal bus frequency, and a minimum of two wires is required to provide the serial data flow synchronized by clock signal in a single direction. An optional hardware slave select control signal can be added. The data size and transmit shift order are configurable, as well as the clock signal polarity and phase or polarity and timing adjustment of the slave select signal. The crucial configuration and settings can be protected by locking.
At the protocol level, the user can use specific data buffers with an optional automatic cyclic redundancy check or CRC calculation, and transfers through the DMA controller.
There are a wide range of SPI events that can generate interrupt requests.

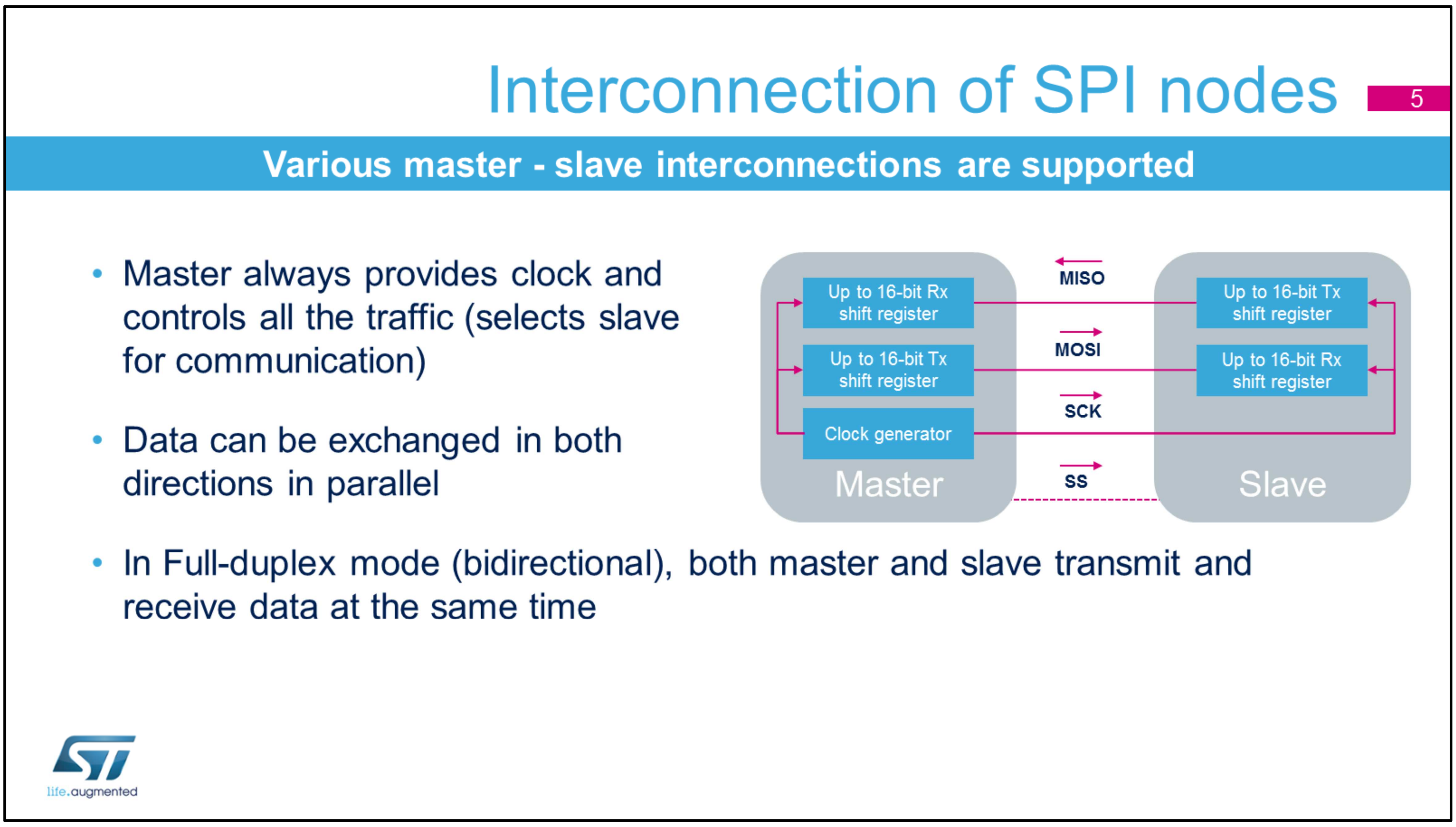
## Block diagram

4



life.augmented

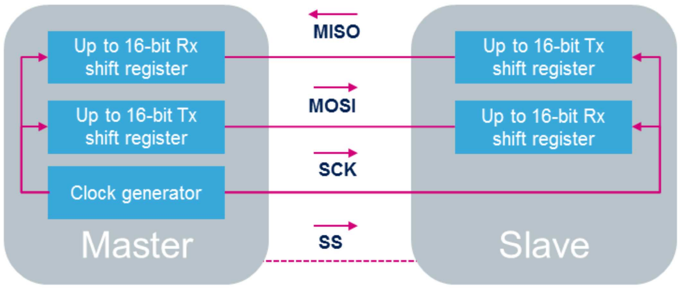
The simplified SPI block diagram shows the peripherals basic control mechanisms and functions. The separated clock domains are highlighted by areas with different colors. All the interconnection signals between the domains are synchronized. The PCLK clock domain has to be clocked when any access of the SPI registers is needed via the peripheral bus interface. The SPI master needs at least an active kernel clock as it has to output the clock signal for the slave. The SPI slave can transfer data without any internal clock signal as the serial interface domain is fully clocked externally via the SCK pin. All of the data passes through receive and transmit buffers via their specific interfaces. The available features can be enabled or disabled depending on the peripheral’s configuration. There are four I/O signals associated with the SPI peripheral. If required, the peripheral can keep control over the associated I/O signals, even when it is disabled, to prevent any unexpected glitches.



## Interconnection of SPI nodes

**Various master - slave interconnections are supported**

- Master always provides clock and controls all the traffic (selects slave for communication)
- Data can be exchanged in both directions in parallel

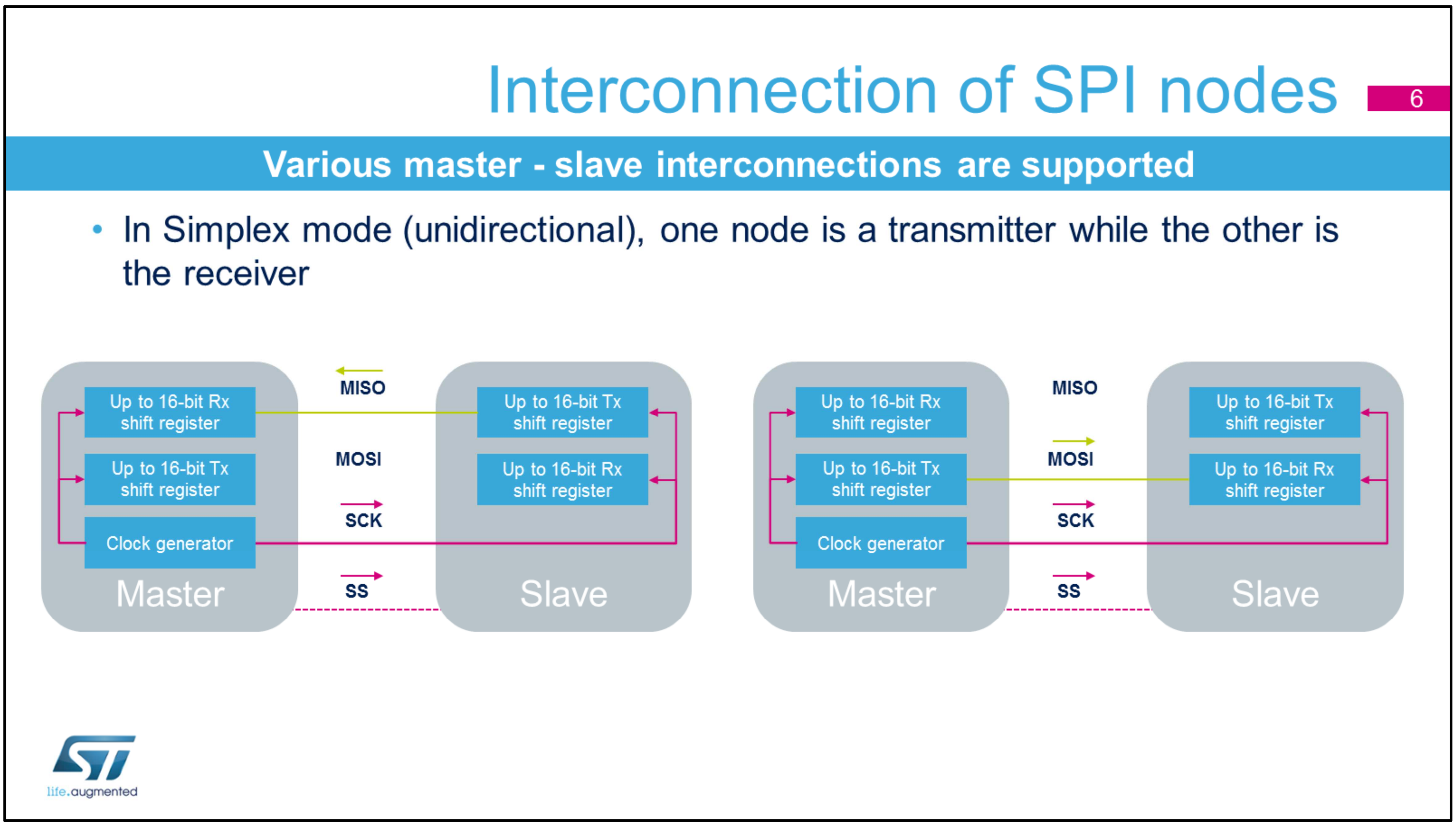


Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MISO
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave

- In Full-duplex mode (bidirectional), both master and slave transmit and receive data at the same time

life.augmented

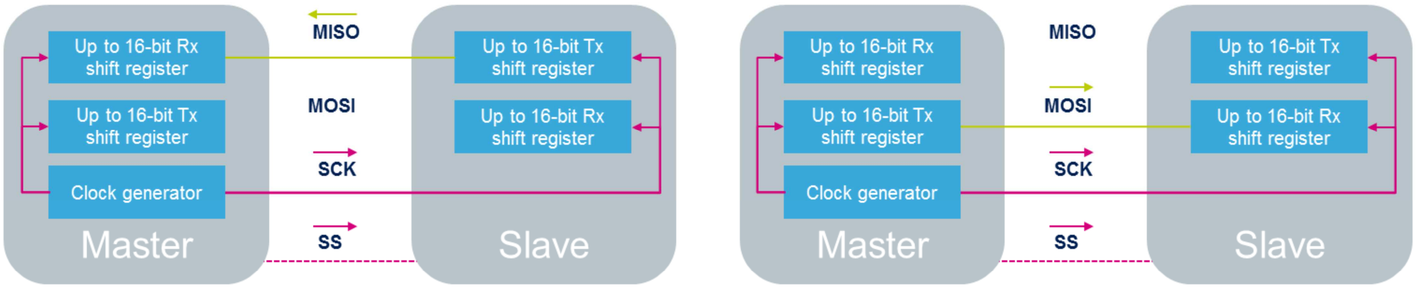
The SPI master always controls the bus traffic and provides the clock signal to the dedicated slave through the SCK line. The master can select the slave it wants to communicate with through the optional Slave Select or SS signal. Data stored in the dedicated shift registers can be exchanged synchronously between the master and slave through the MOSI (Master Output, Slave Input) and the MISO (Master Input, Slave Output) data lines. In Full-duplex mode, both data lines are used and synchronous data flows in both directions.



## Interconnection of SPI nodes

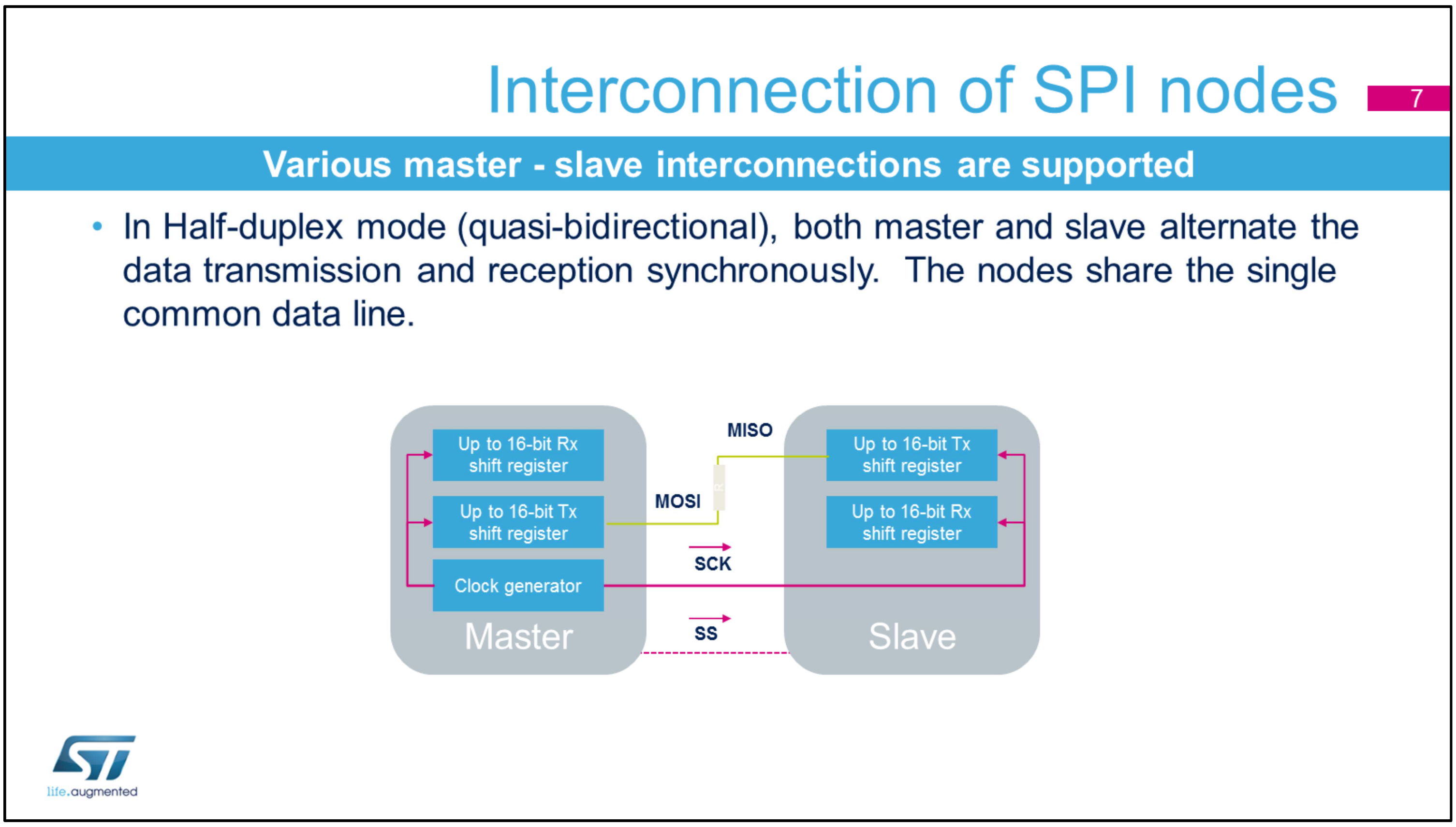
**Various master - slave interconnections are supported**

- In Simplex mode (unidirectional), one node is a transmitter while the other is the receiver


Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MISO
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave
Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MISO
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave

life.augmented

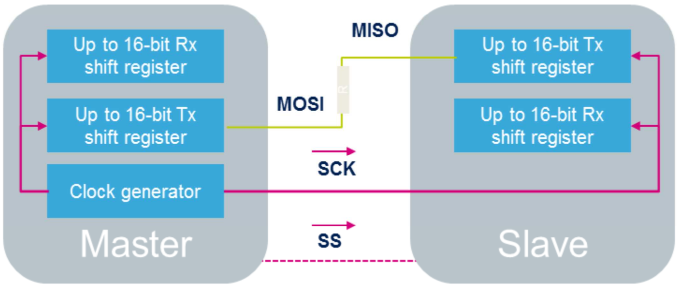
In Simplex mode, one node transmits data while the other receives the data. Data only flows in one direction. Depending on the communication direction, only one data line is used. Unused SPI pins can be used for other purposes.



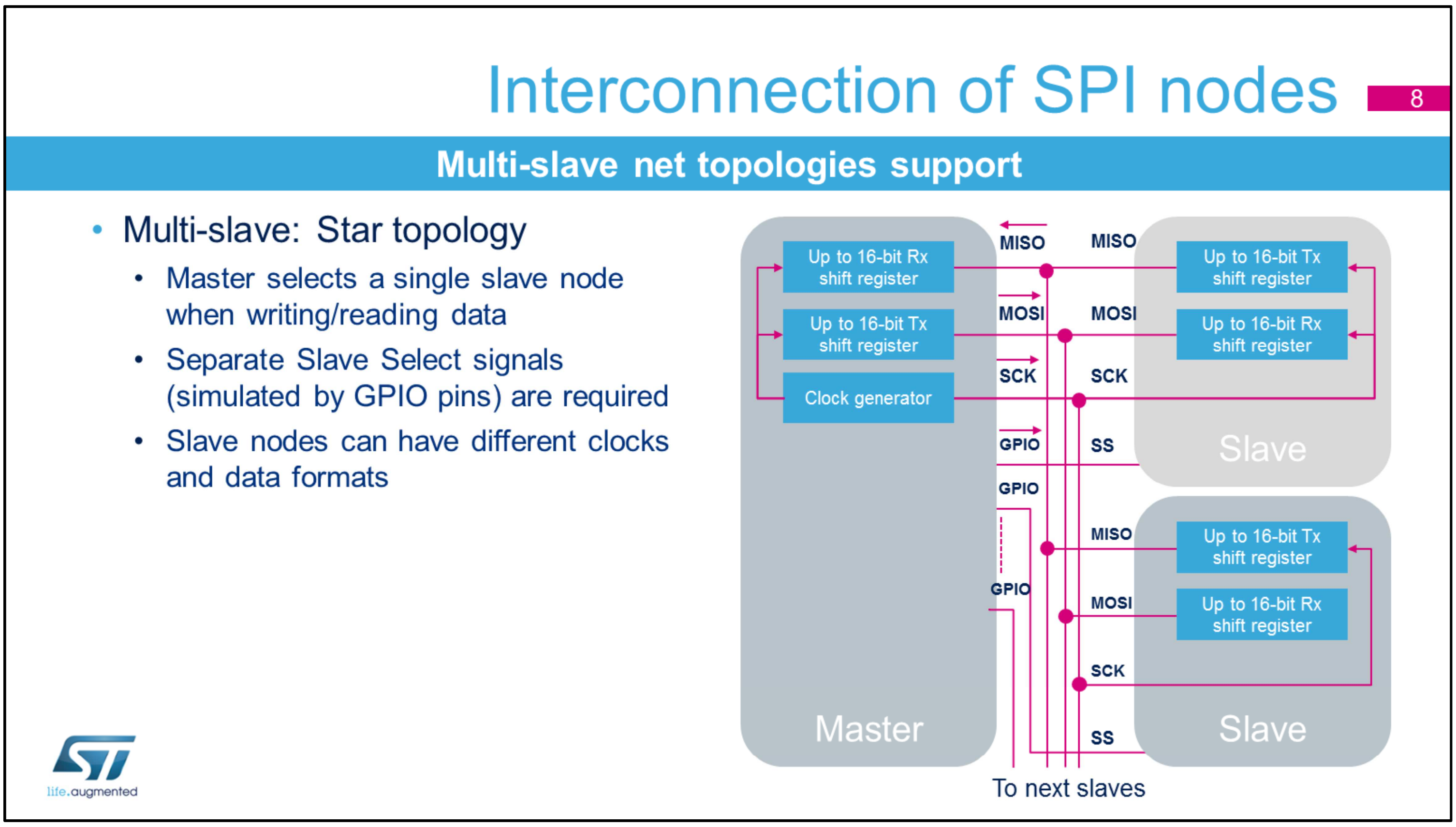
## Interconnection of SPI nodes

**Various master - slave interconnections are supported**

- In Half-duplex mode (quasi-bidirectional), both master and slave alternate the data transmission and reception synchronously. The nodes share the single common data line.


MISO
Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave

Half-duplex mode integrates the previous two modes by sharing a single line for data exchanges and data flows in a single direction at a time. There is a cross connection between the master MOSI and the slave MISO pins in this mode. The master and slave have to alternate their transmitter and receiver roles synchronously when having a common data line. It is common to add a serial resistor on the half duplex data line to prevent possible temporary short-circuit connection, since master and slave nodes are not usually synchronized.

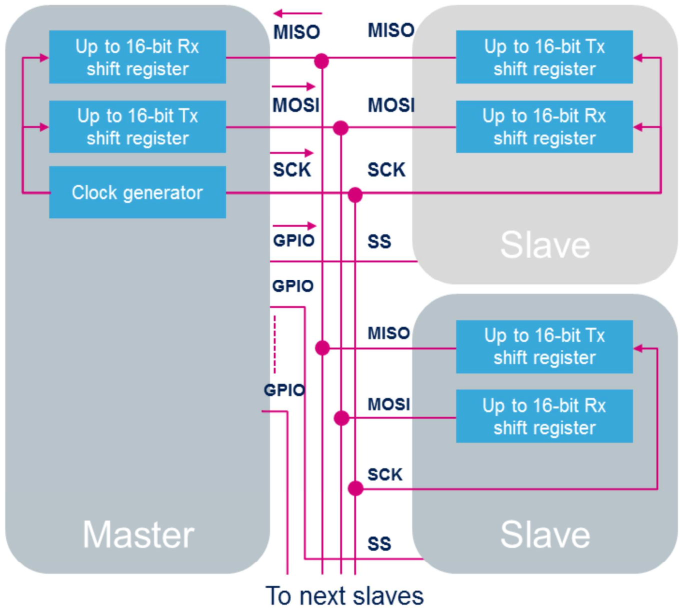


## Interconnection of SPI nodes

8

**Multi-slave net topologies support**

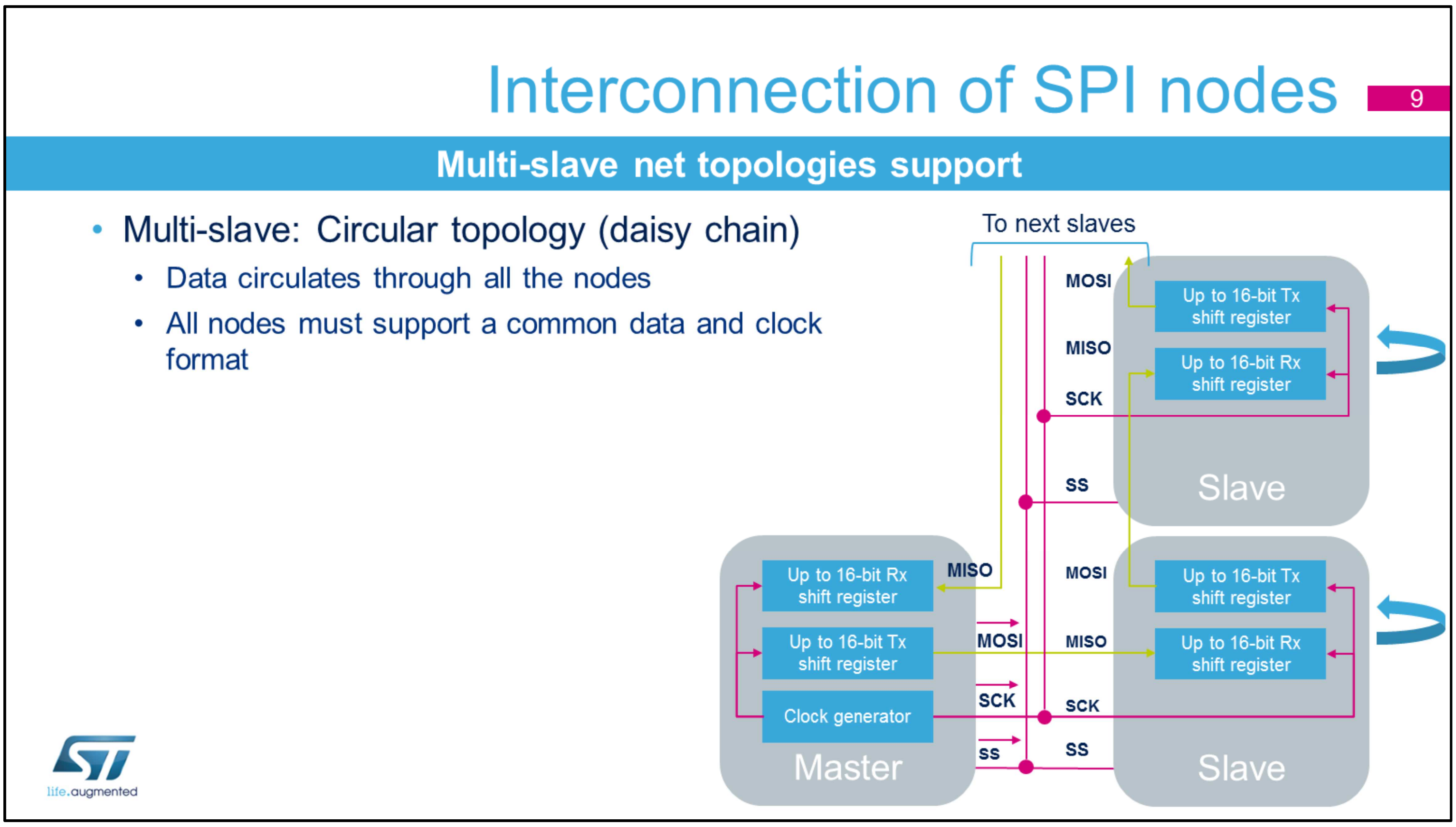
- Multi-slave: Star topology
  - Master selects a single slave node when writing/reading data
  - Separate Slave Select signals (simulated by GPIO pins) are required
  - Slave nodes can have different clocks and data formats



Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MISO
MOSI
SCK
GPIO
GPIO
GPIO
MISO
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave
MISO
MOSI
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave
To next slaves

life.augmented

When the SPI network includes more than one slave, a star topology is commonly used. The master communicates with one slave at a time, since you can only have one slave transmit data back to the master through the common MISO pin. In this topology, a separate Slave Select signal from the master has to be provided to each slave node, so the master can select which slave to communicate with. Thanks to separate Slave Select signals, SPI data and clock format can be adapted for each slave, if the multiple slave nodes do not have a common configuration.

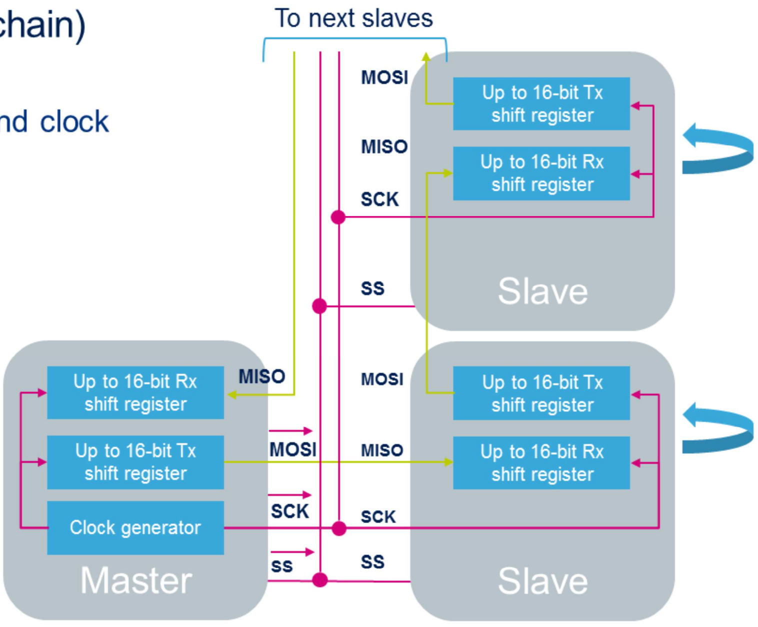


## Interconnection of SPI nodes

9

**Multi-slave net topologies support**

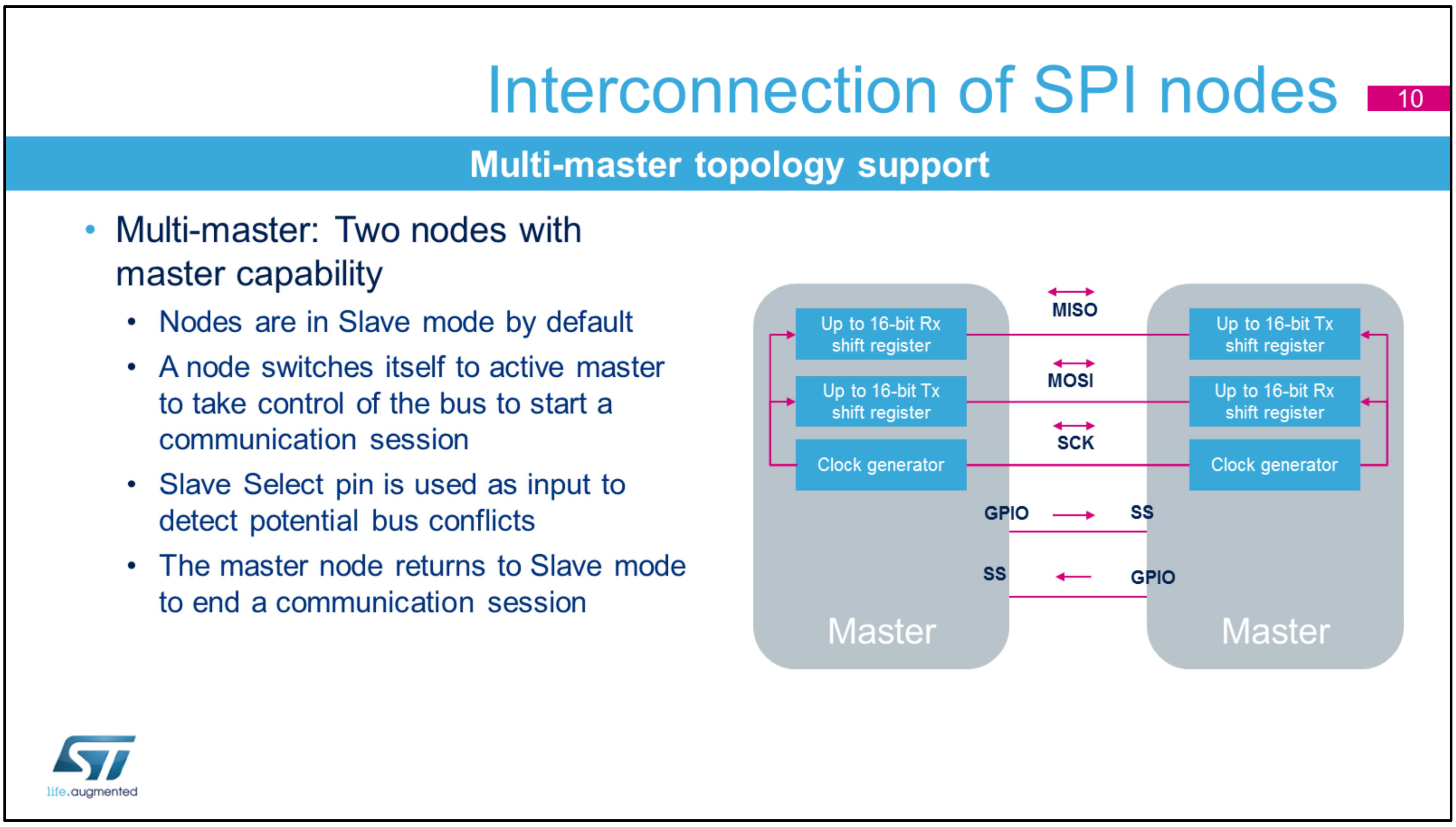
- Multi-slave: Circular topology (daisy chain)
  - Data circulates through all the nodes
  - All nodes must support a common data and clock format



To next slaves
MOSI
MISO
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave
Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
Master
MISO
MOSI
SCK
SS
MOSI
MISO
SCK
SS
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Slave

life.augmented

Another multi-slave configuration is the circular topology where the inputs and outputs of all the nodes are connected together in a closed serial chain. A common Slave Select signal is used for all the nodes as communication occurs at the same time. All nodes must have the same data and clock format configuration. Microcontroller SPI nodes typically use separate internal transmit and receive shift registers, so the data transferred between them has to be handled by software in a circular mode.
A specific underrun feature can be selected during the slave configuration to automatically handle these transfers by hardware.

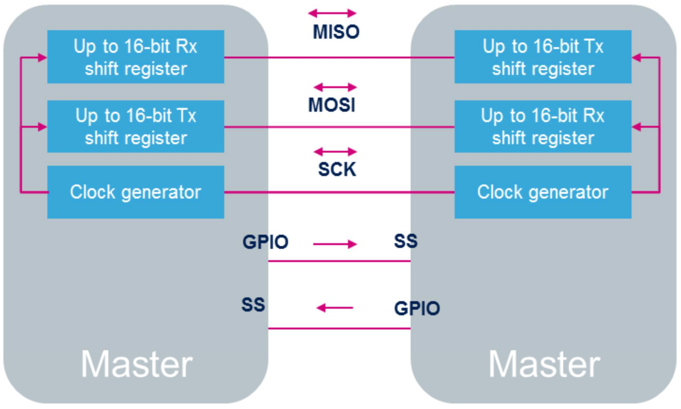


## Interconnection of SPI nodes

10

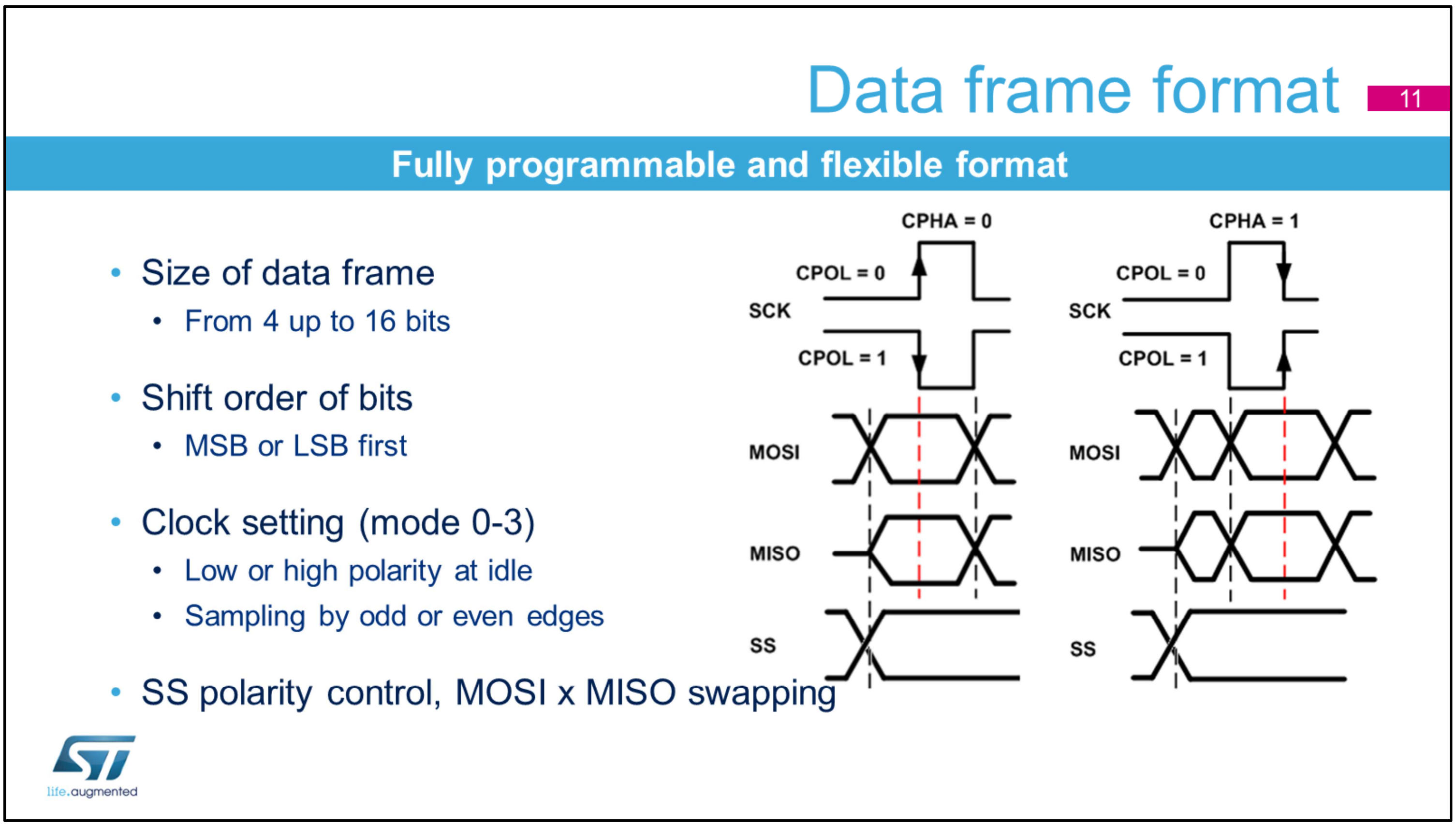
### Multi-master topology support

- Multi-master: Two nodes with master capability
  - Nodes are in Slave mode by default
  - A node switches itself to active master to take control of the bus to start a communication session
  - Slave Select pin is used as input to detect potential bus conflicts
  - The master node returns to Slave mode to end a communication session


Up to 16-bit Rx shift register
Up to 16-bit Tx shift register
Clock generator
MISO
MOSI
SCK
Up to 16-bit Tx shift register
Up to 16-bit Rx shift register
Clock generator
GPIO
SS
SS
GPIO
Master
Master

life.augmented

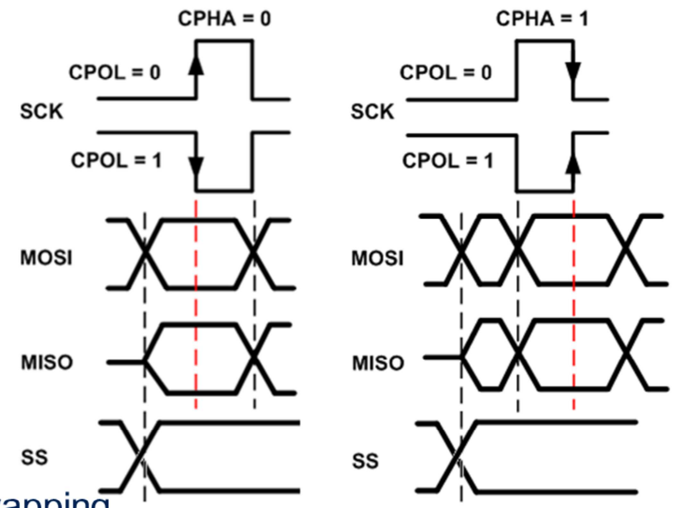
SPI networks can operate in a multi-master environment. This mode is used to connect two master nodes exclusively. When neither node is active, they are by default in a slave mode. When one node wants to take control of the bus, it switches itself into Master mode and asserts the Slave Select signal on the other node through a GPIO pin. Both Slave Select (SS) pins work as a hardware input to detect potential bus collisions between nodes as only one can master the SPI bus at a single time. After the session is completed, the active node master releases the Slave Select signal and returns back to passive slave mode waiting for the next session to start.



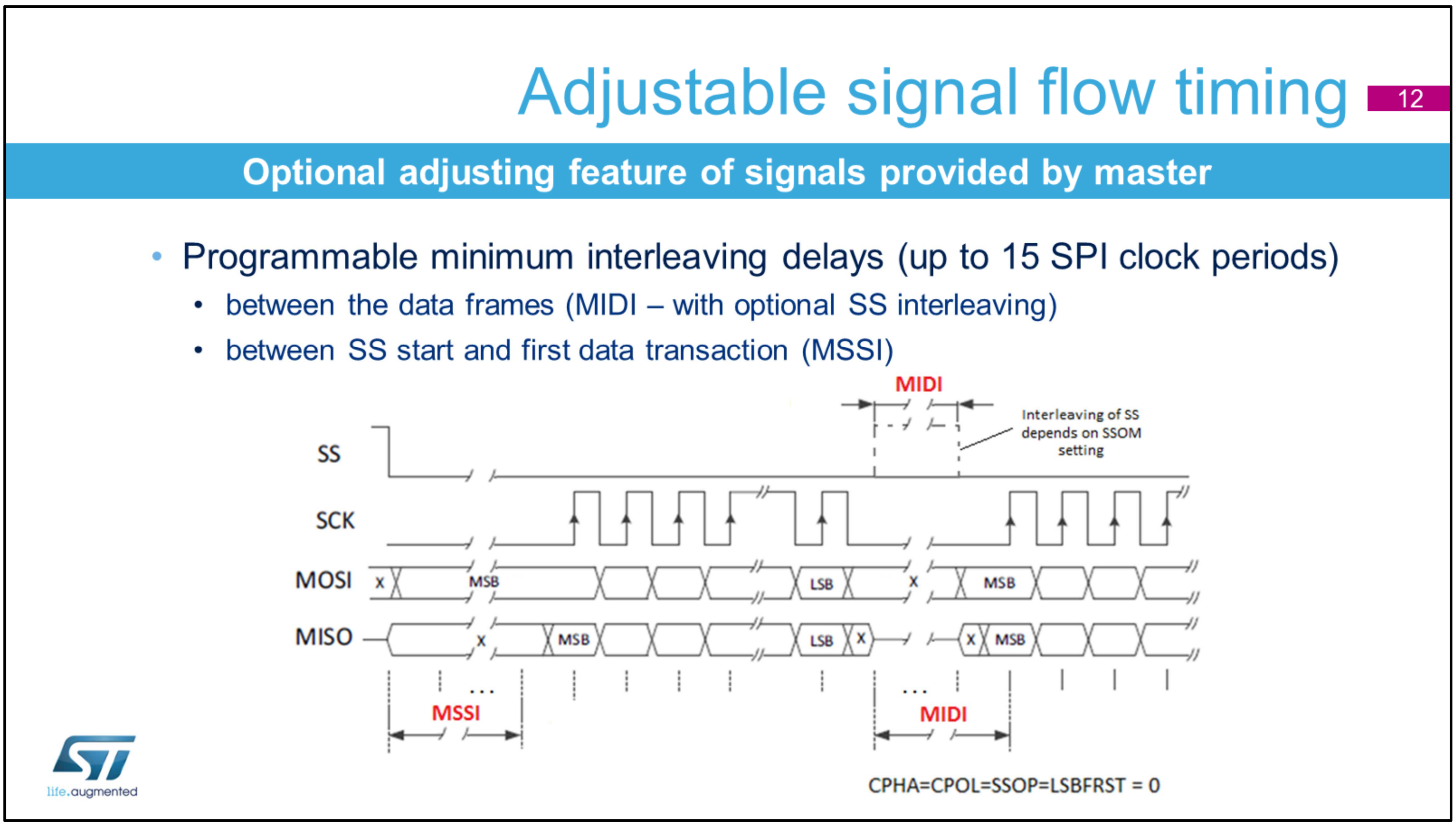
## Data frame format

**Fully programmable and flexible format**

- Size of data frame
  - From 4 up to 16 bits
- Shift order of bits
  - MSB or LSB first
- Clock setting (mode 0-3)
  - Low or high polarity at idle
  - Sampling by odd or even edges
- SS polarity control, MOSI x MISO swapping



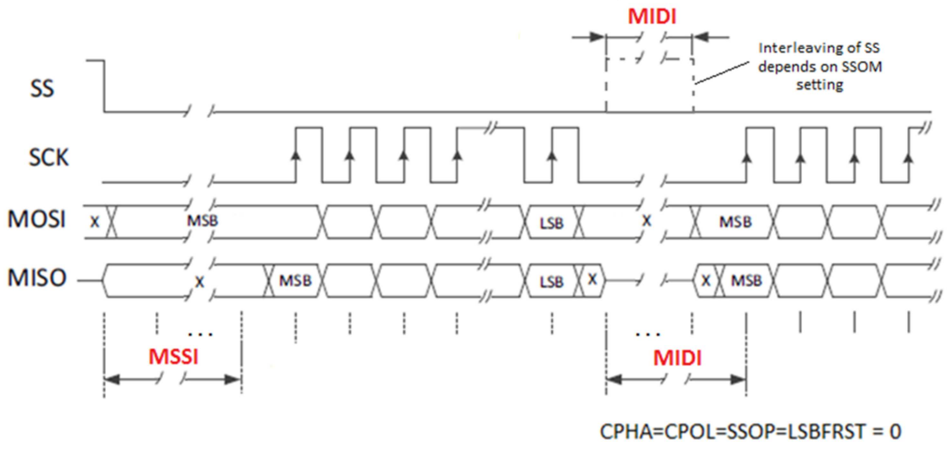
Several parameters are used to setup the data format. Users can define the data frame size and the transmit order of the shift register. The clock can be set to one of four basic configurations defined in the Motorola SPI specifications. The combination of two bits controls the polarity and phase of the clock signal. When the phase control bit is cleared, data bits are sampled on the odd clock edges, and the even clock edges synchronize the shifting of the next bit onto the data line. This is the opposite when the phase control bit is set. The clock polarity bit defines the idle state of the clock signal and which clock edge is used for data sampling or shifting.
The Slave Select signal can be used with both polarities and the functionality of the MOSI and MISO signals can be swapped.



## Adjustable signal flow timing

**Optional adjusting feature of signals provided by master**

- Programmable minimum interleaving delays (up to 15 SPI clock periods)
  - between the data frames (MIDI – with optional SS interleaving)
  - between SS start and first data transaction (MSSI)


MIDI
Interleaving of SS depends on SSOM setting
SS
SCK
MOSI
MISO
x
MSB
LSB
MSSI
MIDI
CPHA=CPOL=SSOP=LSBFRST = 0

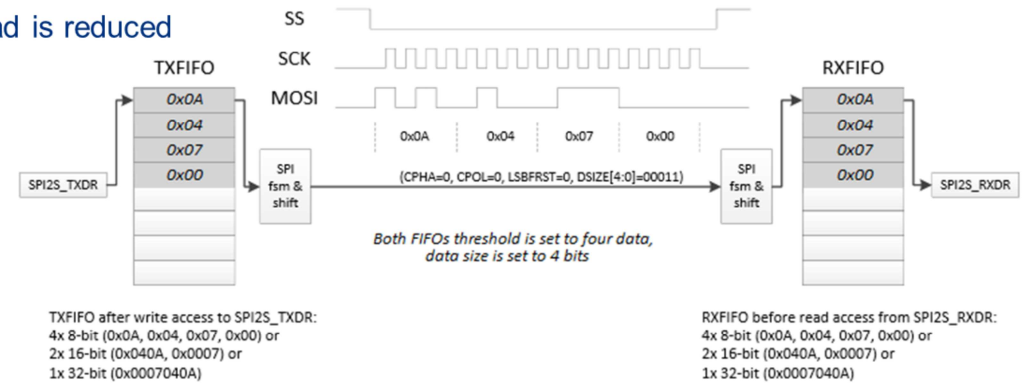
life.augmented

Several parameters can be used to optionally adjust the timing of the master transaction flow.
Signal timings for the master node can be adjusted when needed. This is the case when a slave node needs a longer time to wake up from sleep mode following the setup of the SS active edge or when the slave is not able to handle a data flow that is too fast. Up to 15 additional serial clock signal period~~s~~ delays can be inserted by MIDI or MSSI parameters. Data frames can be optionally interleaved by SS pulses.

## Data packing, FIFO access

**Advanced low demand control**

- Packing mode
  - Access of FIFO registers by multiply data patterns
  - Configurable FIFO threshold levels
  - DMA access
  - Number of events and required services are decreased
  - System load is reduced

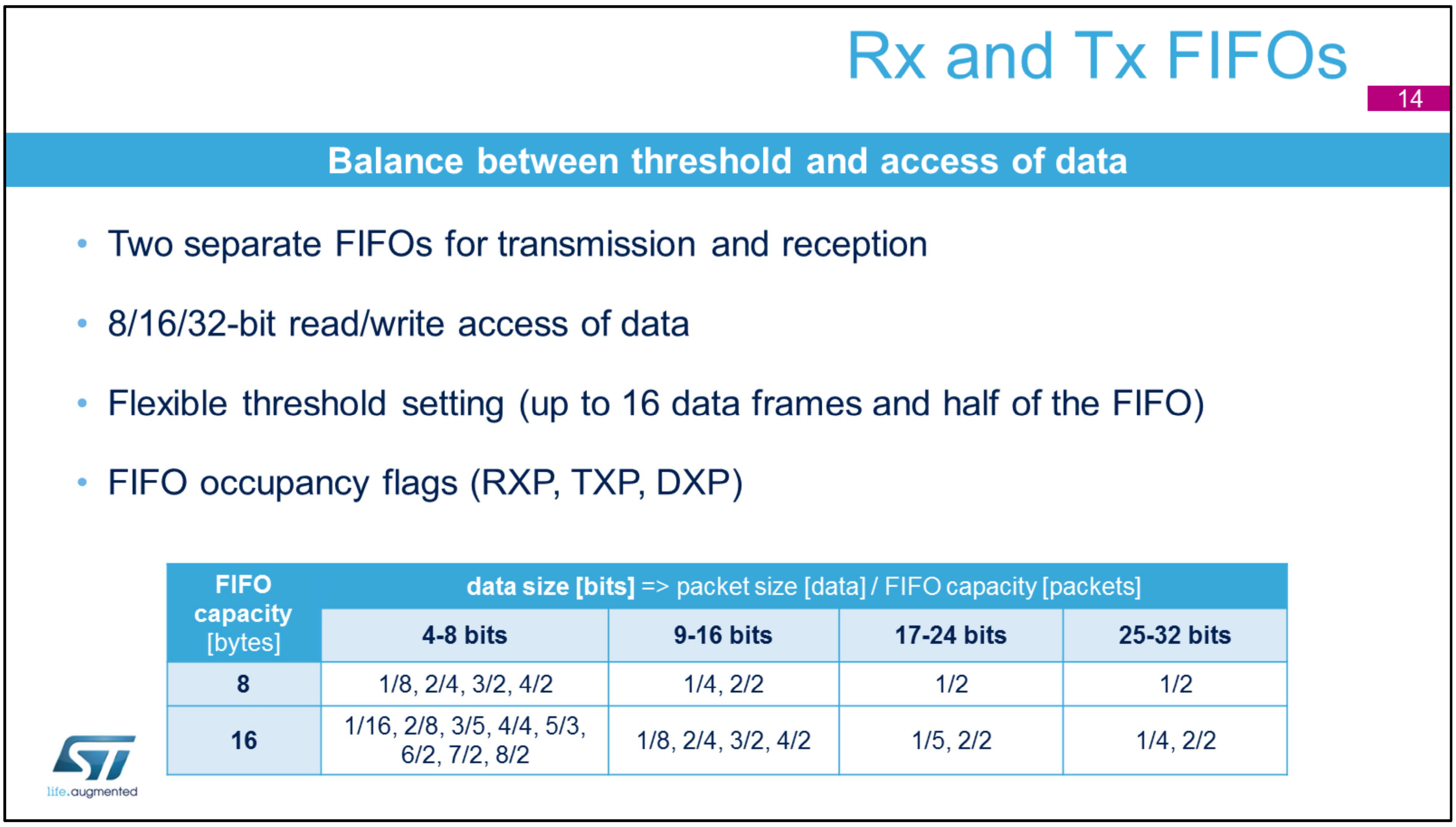

SS
SCK
MOSI
TXFIFO
0x0A
0x04
0x07
0x00
SPI2S_TXDR
SPI fsm & shift
0x0A
0x04
0x07
0x00
(CPHA=0, CPOL=0, LSBFRST=0, DSIZE[4:0]=00011)
Both FIFOs threshold is set to four data, data size is set to 4 bits
RXFIFO
0x0A
0x04
0x07
0x00
SPI fsm & shift
SPI2S_RXDR
TXFIFO after write access to SPI2S_TXDR:
4x 8-bit (0x0A, 0x04, 0x07, 0x00) or
2x 16-bit (0x040A, 0x0007) or
1x 32-bit (0x0007040A)
RXFIFO before read access from SPI2S_RXDR:
4x 8-bit (0x0A, 0x04, 0x07, 0x00) or
2x 16-bit (0x040A, 0x0007) or
1x 32-bit (0x0007040A)

life.augmented

All SPI data transactions pass through the embedded FIFOs organized by bytes. A write access to the SPI write data register stores the written data in the transmit FIFO at the end of a send queue. A read access to the SPI read data register returns the oldest value stored in receive FIFO that has not been read yet.

When the communication speed is fast and data frames too short, it can be a demanding task to ensure correct data flows especially when the clock signal becomes continuous and Full-duplex mode is used. Slave nodes are more critical as they have to follow properly all the transactions timing provided by the master to prevent any data overrun or underrun conditions. The user can organize data into wider packets and handle multiple data read or write events by minimizing access to the data registers. The services are based on FIFO threshold events signalizing Packet Ready operations. If FIFO thresholds are properly set, multiple data

packet services can be performed upon minimum FIFO occupancy events. Read and write events can even be serviced together based on common dual events. These features efficiently decrease the number of services and so minimize the load of the microcontroller when handling data flows. This can be especially useful in low-power modes when data is sent while the microcontroller is sleeping. When the DMA is applied additionally, it helps to significantly reduce overall loading on the system in run mode.
In the figure shown, you can see the principle of how four short data frames can be written and read by a single 32-bit or 16-bit access of the dedicated data registers associated with the FIFOs in according with the FIFO capacity and maximum configurable data size of the SPI instance.



## Rx and Tx FIFOs

14

**Balance between threshold and access of data**

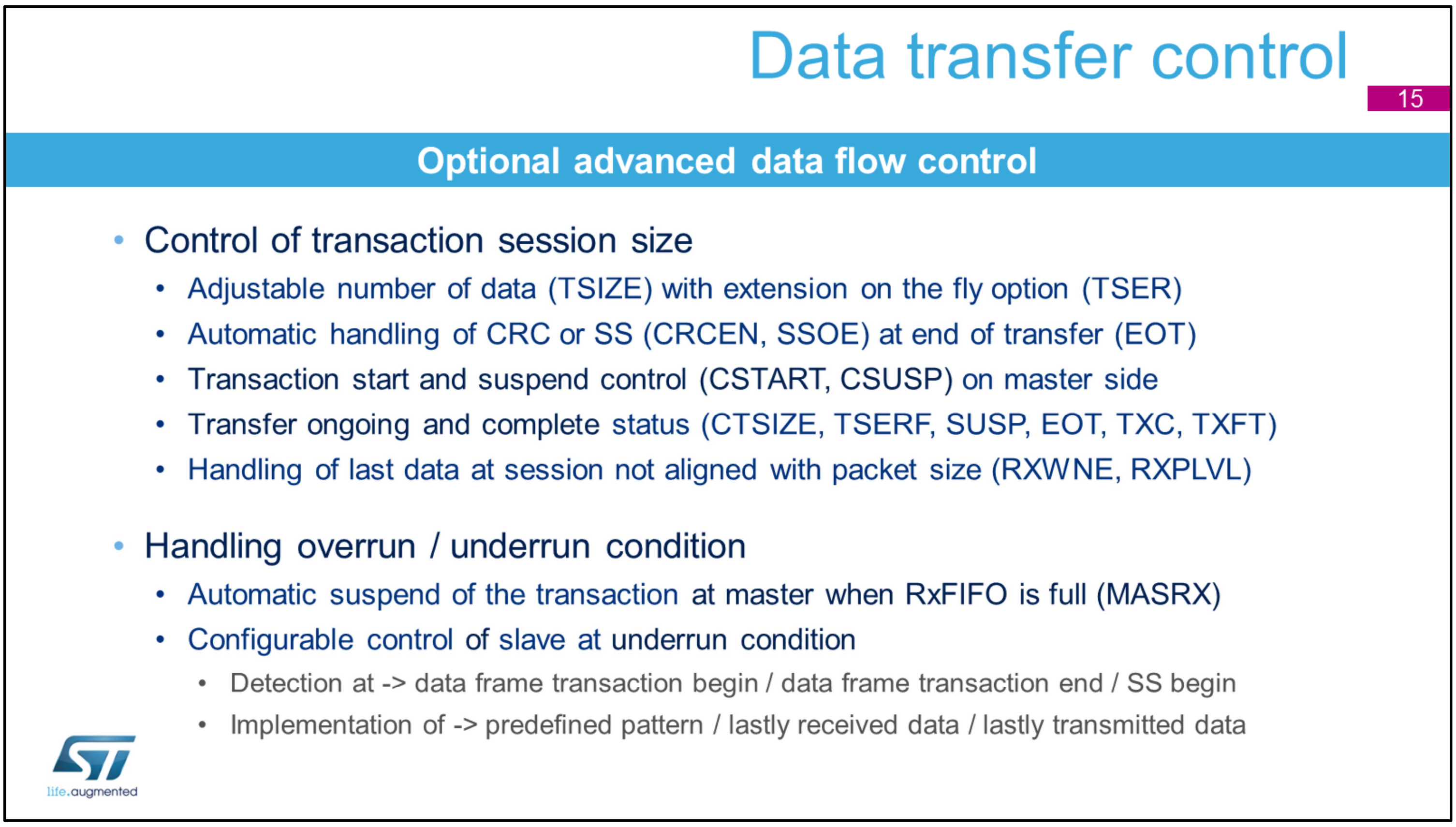
- Two separate FIFOs for transmission and reception
- 8/16/32-bit read/write access of data
- Flexible threshold setting (up to 16 data frames and half of the FIFO)
- FIFO occupancy flags (RXP, TXP, DXP)

| **FIFO capacity** [bytes] | **data size [bits]** => packet size [data] / FIFO capacity [packets] | | | |
|---|---|---|---|---|
| | **4-8 bits** | **9-16 bits** | **17-24 bits** | **25-32 bits** |
| **8** | 1/8, 2/4, 3/2, 4/2 | 1/4, 2/2 | 1/2 | 1/2 |
| **16** | 1/16, 2/8, 3/5, 4/4, 5/3, 6/2, 7/2, 8/2 | 1/8, 2/4, 3/2, 4/2 | 1/5, 2/2 | 1/4, 2/2 |

life.augmented

The SPI peripheral features two FIFOs to handle the data flow.
The capacity of the FIFOs and maximum data frame size depends on the product and the peripheral instance. The frequency of the FIFO occupancy events depends on the FIFO threshold setting which organizes data into packets. A single packet can include up to 16 data frames but its size may not exceed half of the FIFO size. Possible combinations of data and packet size to fit into available FIFO space are listed in the table. Once room for a single packet is available in the transmit FIFO to store a new complete packet of data or a single complete data packet is ready to read in the receive FIFO, the corresponding TXP or RXP occupancy flag is set. The user can then perform proper write or read service of data registers corresponding to a single packet size. Both packets to be transmitted and received can be service by a common handling procedure when the dual occupancy DXP flag is set. The occupancy flags are

evaluated dynamically in relation to the bus traffic and the current FIFO content so they have to be checked exclusively once the associated packet service is fully completed.



## Data transfer control

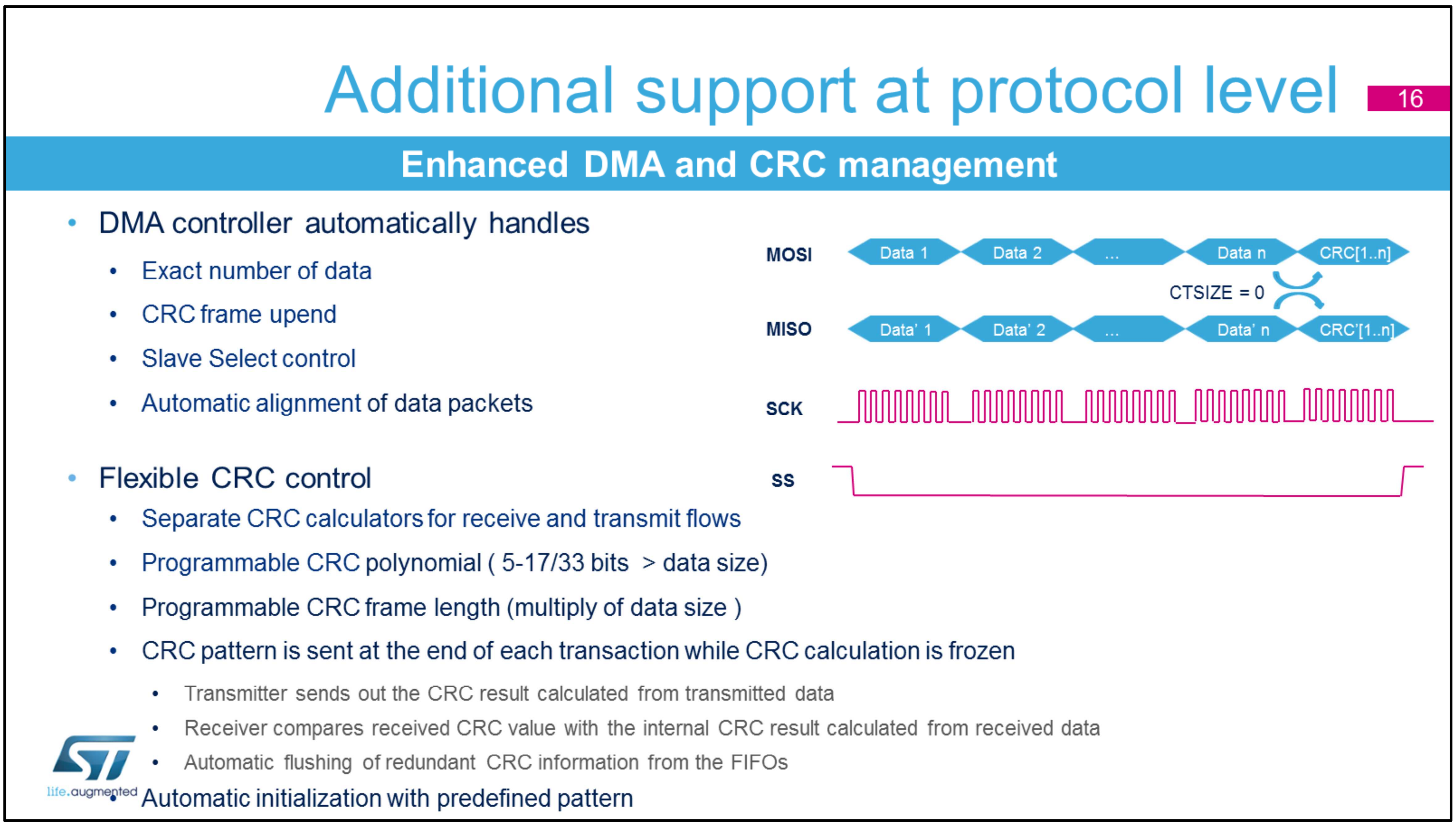
15

**Optional advanced data flow control**

- Control of transaction session size
  - Adjustable number of data (TSIZE) with extension on the fly option (TSER)
  - Automatic handling of CRC or SS (CRCEN, SSOE) at end of transfer (EOT)
  - Transaction start and suspend control (CSTART, CSUSP) on master side
  - Transfer ongoing and complete status (CTSIZE, TSERF, SUSP, EOT, TXC, TXFT)
  - Handling of last data at session not aligned with packet size (RXWNE, RXPLVL)
- Handling overrun / underrun condition
  - Automatic suspend of the transaction at master when RxFIFO is full (MASRX)
  - Configurable control of slave at underrun condition
    - Detection at -> data frame transaction begin / data frame transaction end / SS begin
    - Implementation of -> predefined pattern / lastly received data / lastly transmitted data

Both the master and slave can apply either endless data transfers or handle a defined number of data to be sent within a single session. The number of data is practically unlimited as it can be extended on the fly as long as the transaction is ongoing. A cyclic redundancy check can be implemented automatically or the Slave Select signal can be handled by hardware on the master side during the session. The master can start or suspend ongoing communication at any time. The current data frame is finished when the transaction is suspended. A specific CTSIZE counter counts the number of data frames remaining in the current session, end of transaction (EOT) and the transmission complete TXC flags indicate the end of session and bus idle, while an additional specific TXTF flag indicates all data submitted for transmission and its event disables interrupts from transmission occupancy events. Transmissions automatically stop when the session is completed at the master side, even if there is additional data in the transmit FIFO. All the FIFOs

content is flushed when the SPI is disabled. The user has to read out all the received data before the SPI is disabled. Except for occasional errors, the software has to take care of FIFO occupancy events only while the session is ongoing. Once the session is finished and the remaining received data is not aligned with the packet size, the last RXP occupancy flag is not set but the RXWNE flag is active and level of the current FIFO occupancy is signaled by the specific RXPLVL level counter that depends on the data size. The counter value signals how many data frames remain to handle at the receive FIFO. Nevertheless, application software can still perform full packet reads without any drawbacks as only the consistent data will be popped from the FIFO. Similarly, full packet writes to the transmit FIFO can be applied in this case as only the consistent data will be pushed into the FIFO while redundant writes will be discarded. When the DMA is applied, such non-aligned data is handled automatically.
To prevent overrun conditions and the loss of any data, the master can temporarily suspend ongoing transactions when its receive FIFO is full. The slave can be configured to detect and react to underrun conditions when no data is ready to be sent in its transmit FIFO and when the master continues or starts a new session.



## Additional support at protocol level

16

### Enhanced DMA and CRC management

- DMA controller automatically handles
  - Exact number of data
  - CRC frame upend
  - Slave Select control
  - Automatic alignment of data packets
- Flexible CRC control
  - Separate CRC calculators for receive and transmit flows
  - Programmable CRC polynomial ( 5-17/33 bits > data size)
  - Programmable CRC frame length (multiply of data size )
  - CRC pattern is sent at the end of each transaction while CRC calculation is frozen
    - Transmitter sends out the CRC result calculated from transmitted data
    - Receiver compares received CRC value with the internal CRC result calculated from received data
    - Automatic flushing of redundant CRC information from the FIFOs
  - Automatic initialization with predefined pattern

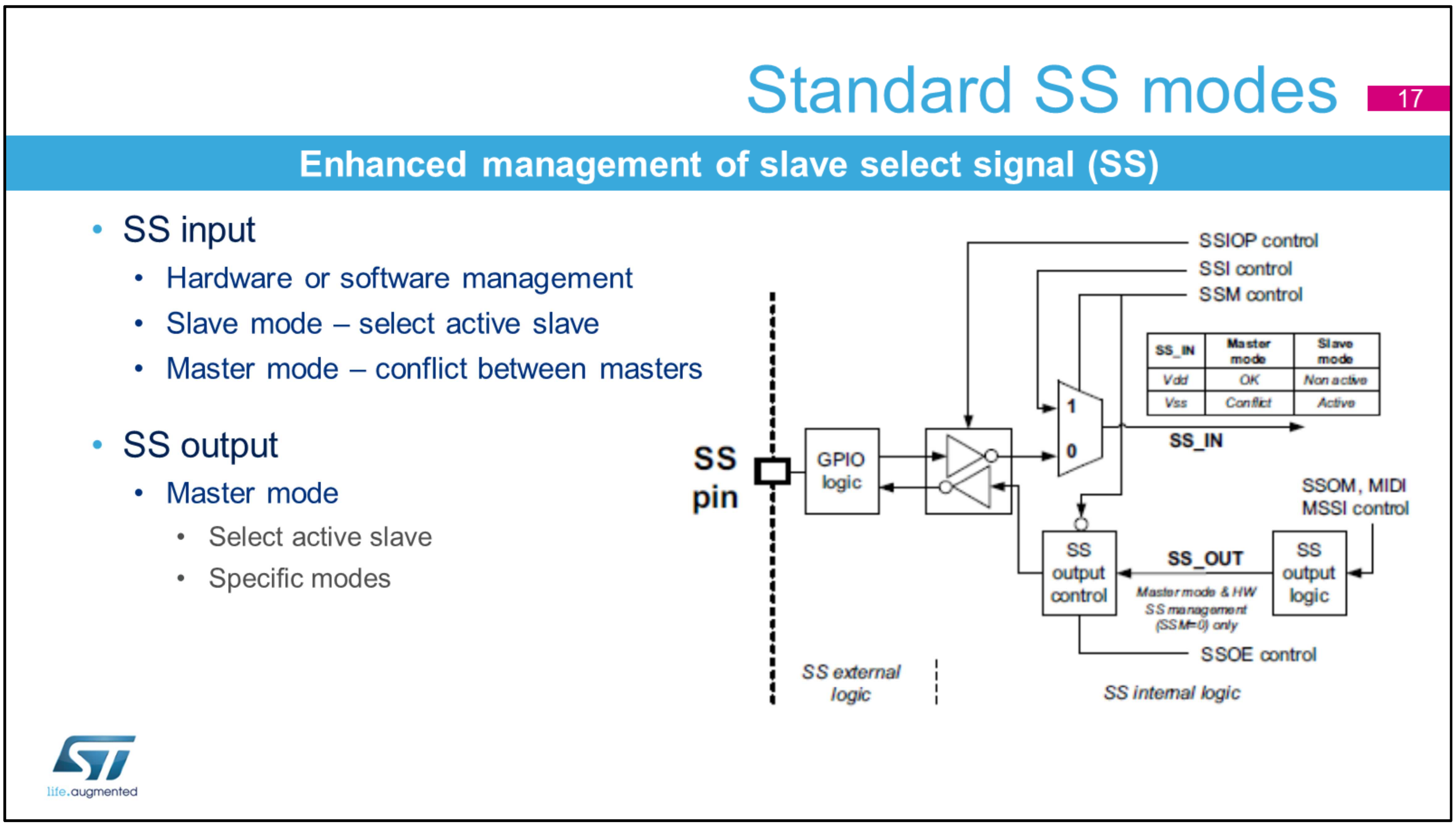
During protocol level communications, the DMA controller can be used to automatically handle the data flow events, the CRC calculations, and the updating of the FIFO threshold. In case of threshold control, the last odd data frame is correctly applied in packed mode when the number of frames is not aligned with the packet size.

If the CRC is enabled, separate CRC calculators are used for the transmitter and receiver. The CRC calculation result is automatically appended at the end of each transfer by the DMA controller or by software control.
Both CRC polynomial pattern and CRC frame length are programmable. The size of the polynomial is defined by its most significant non-zero bit and it has to be always longer than the data size.
The CRC frame length has to be equal or a multiple of

the data size. The CRC calculation is frozen during the CRC pattern transaction.

Results from the transmitter CRC calculator register are loaded directly into the shift register, and the received CRC value is stored in the FIFO and compared with the receiver CRC result. Redundant CRC information are automatically flushed from the FIFOs.

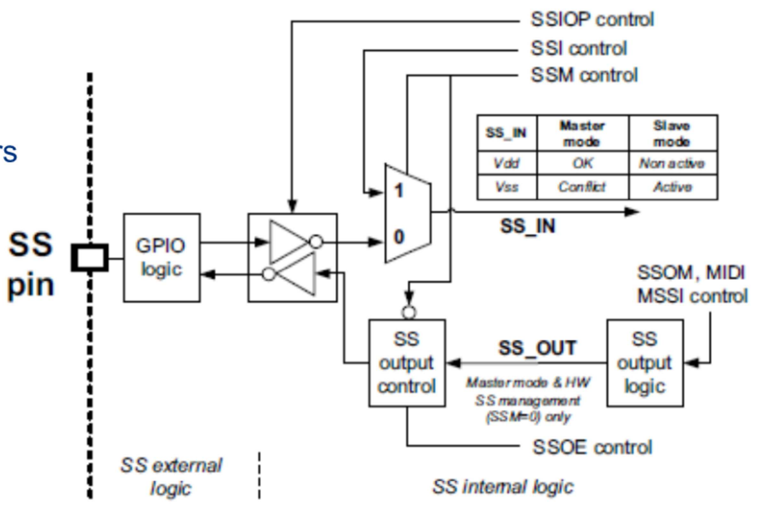
As the CRC registers are automatically initialized, the CRC can be used in DMA circular mode. Initialization patterns for the receiver and transmitter can be configured either to zero or to all ones. By this flexibility, a wide range of protocols is covered.



## Standard SS modes

**Enhanced management of slave select signal (SS)**

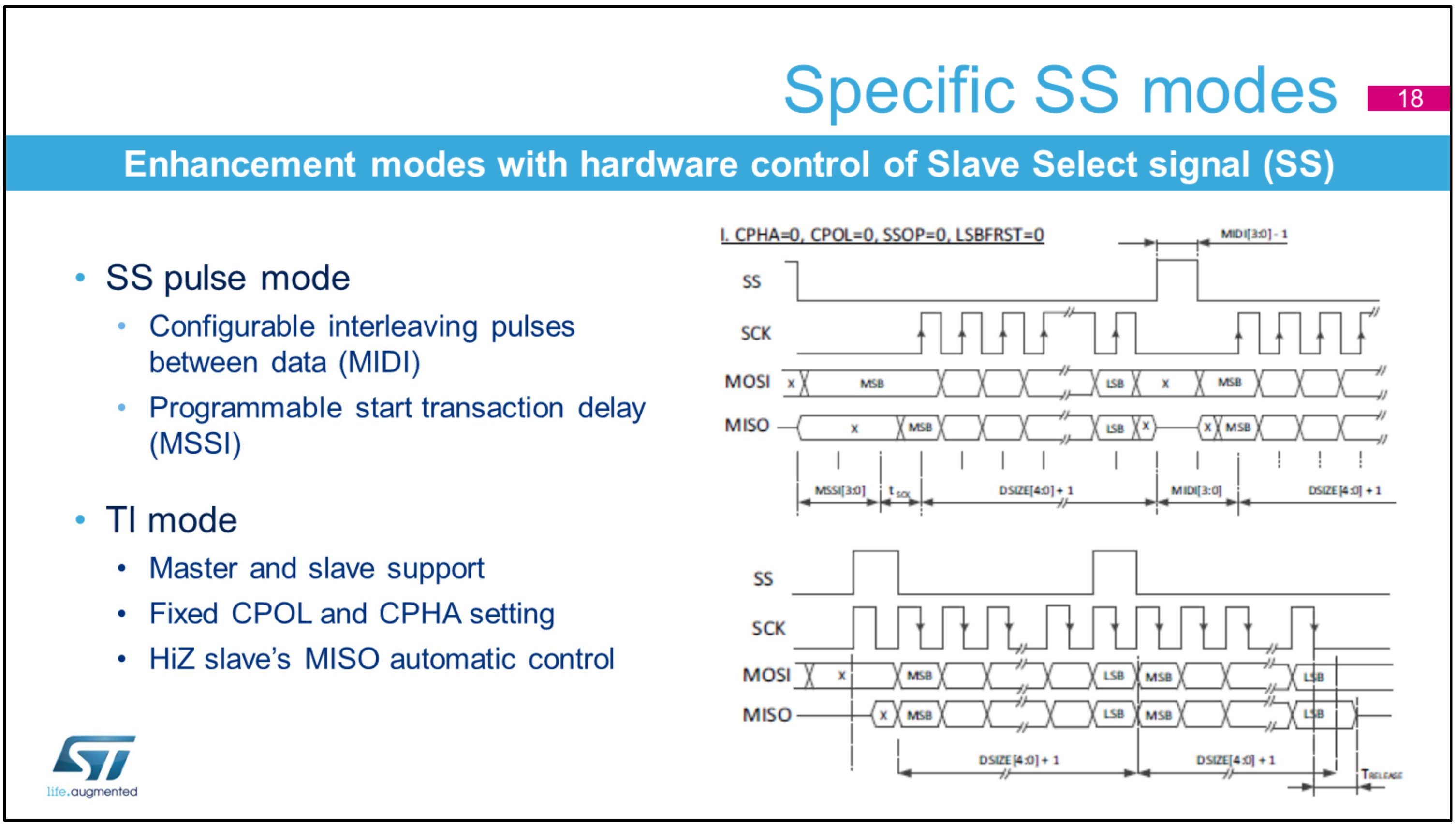
- SS input
  - Hardware or software management
  - Slave mode – select active slave
  - Master mode – conflict between masters
- SS output
  - Master mode
    - Select active slave
    - Specific modes



| SS_IN | Master mode | Slave mode |
|---|---|---|
| Vdd | OK | Non active |
| Vss | Conflict | Active |

life.augmented

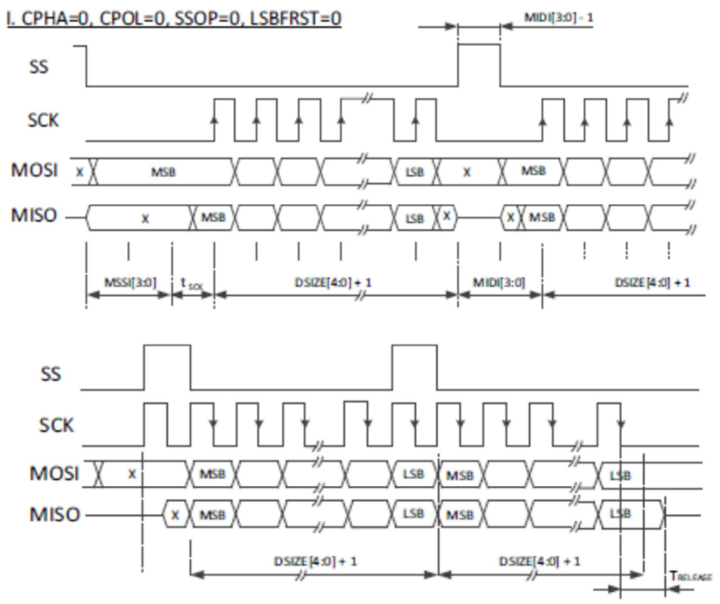
The Slave Select signal is commonly used by the master node to select the slave node for communication.
The signal implementation is mandatory in multi-master and multi-slave topologies. Though it is not mandatory at a single master-slave pair, it could be helpful for data flow synchronization, regardless of the topology case.
The Slave Select signal can operate as an input or as an output depending on the SSIOP control bit.
The SS input can be managed by hardware or software depending on the SSM and SSI control bits, in either Master or Slave modes.
As a slave input, it is used to identify itself as the active slave for communication. As a master input, it signalizes a potential conflict between masters in a multi-master system.
The SS only works as an output in Master mode and is managed by hardware in a standard or specific control mode. Additional slave select outputs can be provided by the GPIOs under software control.



## Specific SS modes

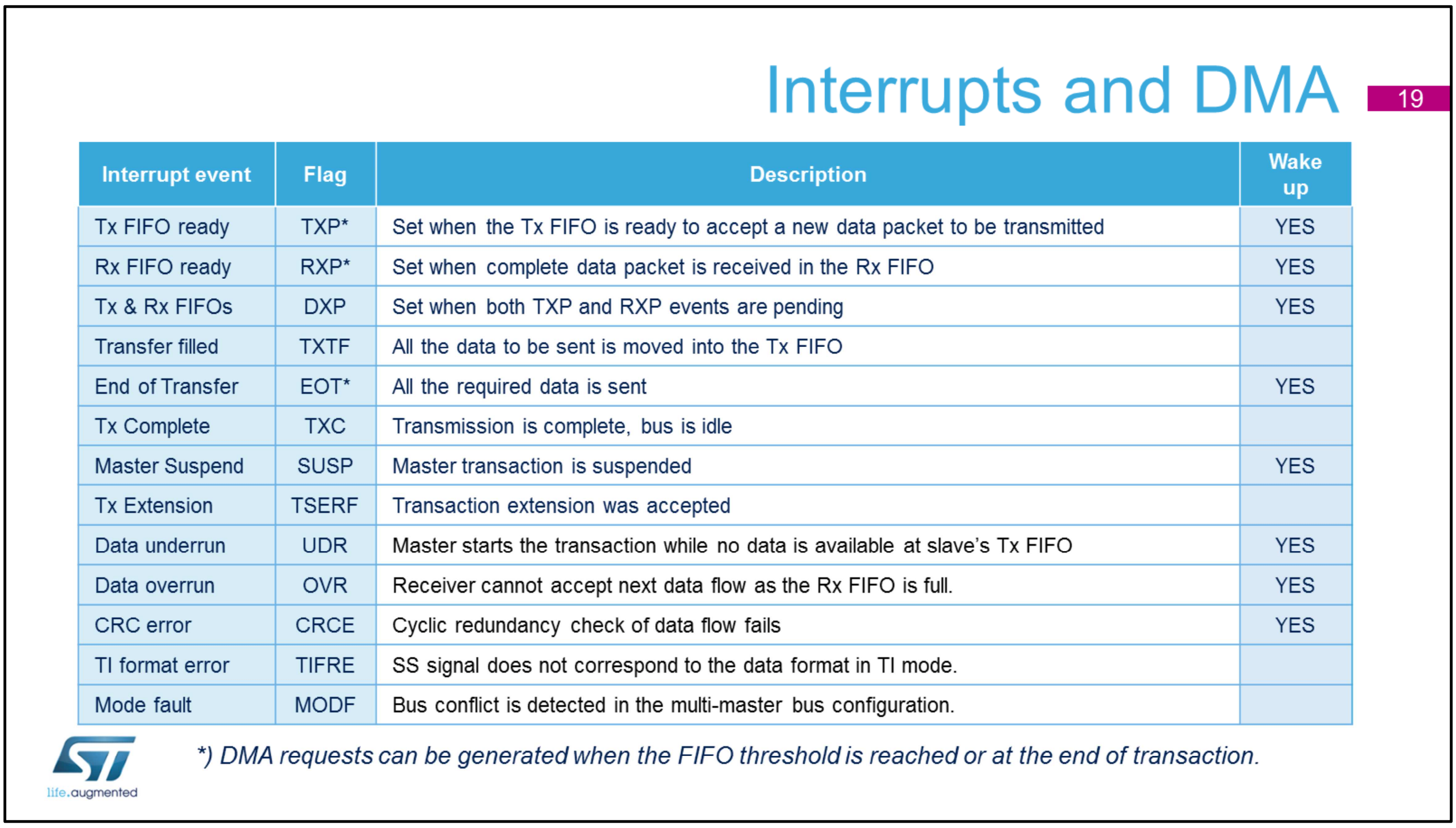
**Enhancement modes with hardware control of Slave Select signal (SS)**

- SS pulse mode
  - Configurable interleaving pulses between data (MIDI)
  - Programmable start transaction delay (MSSI)
- TI mode
  - Master and slave support
  - Fixed CPOL and CPHA setting
  - HiZ slave's MISO automatic control



life.augmented

There are a few enhanced modes when the Slave Select signal is under specific hardware control. The Slave Select signal can operate in a pulse mode where the master generates pulses on the signal between data frames. The duration of the pulses is programmable in steps of SPI clock periods. The clock phase and polarity are configurable in this mode.
Another enhanced mode is the TI mode where the data flow is synchronized by the SS pulses, provided by the master, on the last bit of data. The clock polarity and phase configuration is fixed and the slave data output is automatically switched into high impedance when the bus traffic stops and on a specific configurable timeout.



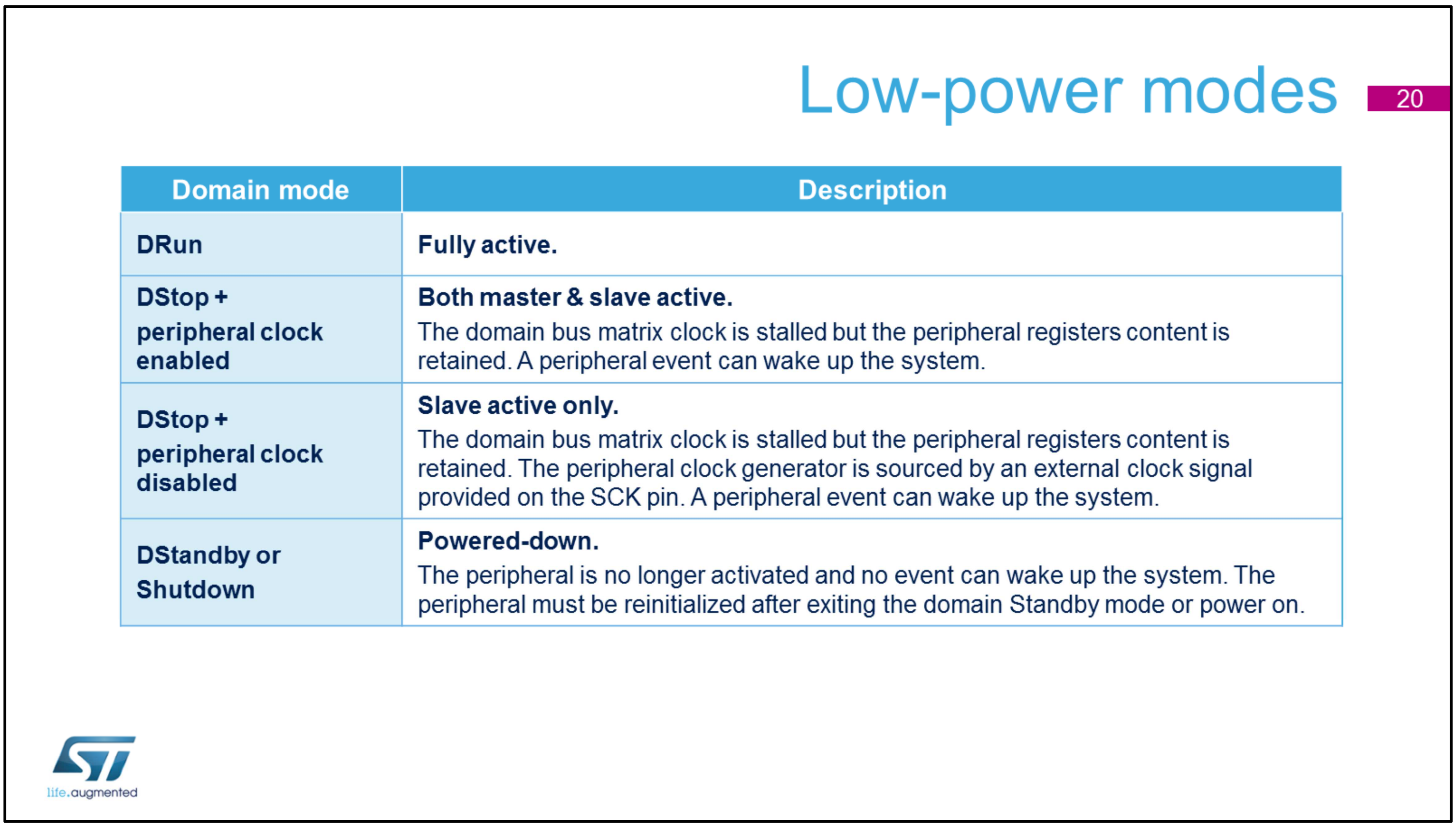
## Interrupts and DMA

| Interrupt event | Flag | Description | Wake up |
|---|---|---|---|
| Tx FIFO ready | TXP* | Set when the Tx FIFO is ready to accept a new data packet to be transmitted | YES |
| Rx FIFO ready | RXP* | Set when complete data packet is received in the Rx FIFO | YES |
| Tx & Rx FIFOs | DXP | Set when both TXP and RXP events are pending | YES |
| Transfer filled | TXTF | All the data to be sent is moved into the Tx FIFO | |
| End of Transfer | EOT* | All the required data is sent | YES |
| Tx Complete | TXC | Transmission is complete, bus is idle | |
| Master Suspend | SUSP | Master transaction is suspended | YES |
| Tx Extension | TSERF | Transaction extension was accepted | |
| Data underrun | UDR | Master starts the transaction while no data is available at slave's Tx FIFO | YES |
| Data overrun | OVR | Receiver cannot accept next data flow as the Rx FIFO is full. | YES |
| CRC error | CRCE | Cyclic redundancy check of data flow fails | YES |
| TI format error | TIFRE | SS signal does not correspond to the data format in TI mode. | |
| Mode fault | MODF | Bus conflict is detected in the multi-master bus configuration. | |

**) DMA requests can be generated when the FIFO threshold is reached or at the end of transaction.*

life.augmented

Here is an overview of the SPI interrupt events. There are FIFO and error detection events to handle data flows. DMA requests are triggered internally by FIFO threshold events. The EOT event raises automatic control of the last incomplete packet when the overall number of data sent is not aligned with data packet size.

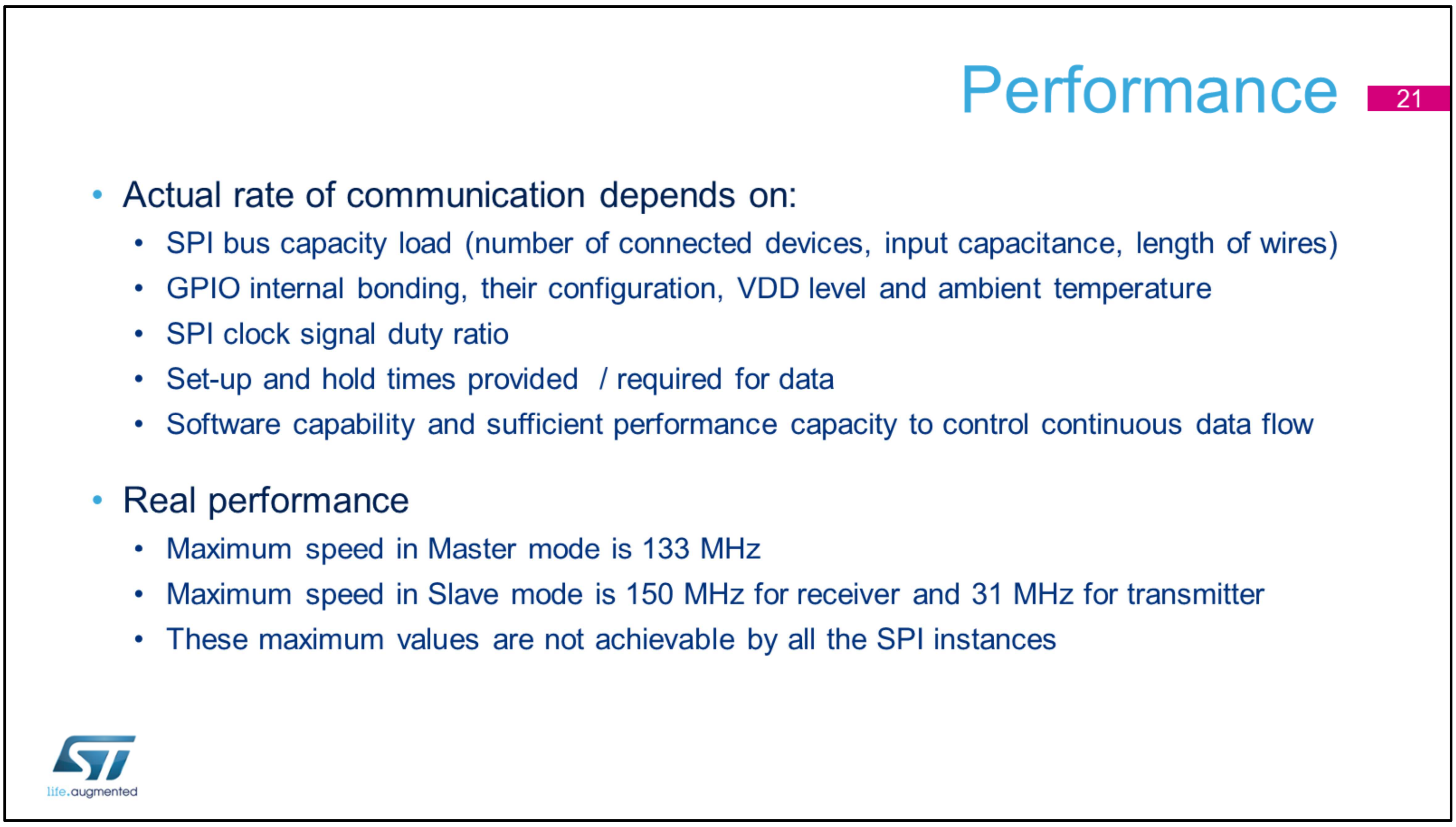


## Low-power modes

| Domain mode | Description |
|---|---|
| **DRun** | **Fully active.** |
| **DStop + peripheral clock enabled** | **Both master & slave active.** The domain bus matrix clock is stalled but the peripheral registers content is retained. A peripheral event can wake up the system. |
| **DStop + peripheral clock disabled** | **Slave active only.** The domain bus matrix clock is stalled but the peripheral registers content is retained. The peripheral clock generator is sourced by an external clock signal provided on the SCK pin. A peripheral event can wake up the system. |
| **DStandby or Shutdown** | **Powered-down.** The peripheral is no longer activated and no event can wake up the system. The peripheral must be reinitialized after exiting the domain Standby mode or power on. |

life.augmented

Here is an overview of the SPI status in specific low-power modes. The relevant system power domain depends on the peripheral instance. Every instance is divided and handled by three separated clock domains. The PCLK clock domain has to be clocked when any access of the SPI registers is needed via the peripheral bus interface. If the bus matrix clock is stalled, the peripheral registers content is retained. If the peripheral kernel clock is enabled, the clock generator can control the SPI master operation too; otherwise the peripheral is able to operate in slave mode only because the peripheral serial interface is then clocked by the external clock signal via SCK pin exclusively. A peripheral event can be configured to wake up the system. The device cannot operate when its domain is in standby or shut down mode. That is why it is important to ensure that all SPI traffic is completed before the peripheral domain enters standby or shut down mode. The application acknowledges all pending interrupts before switching the SPI to low-power mode.



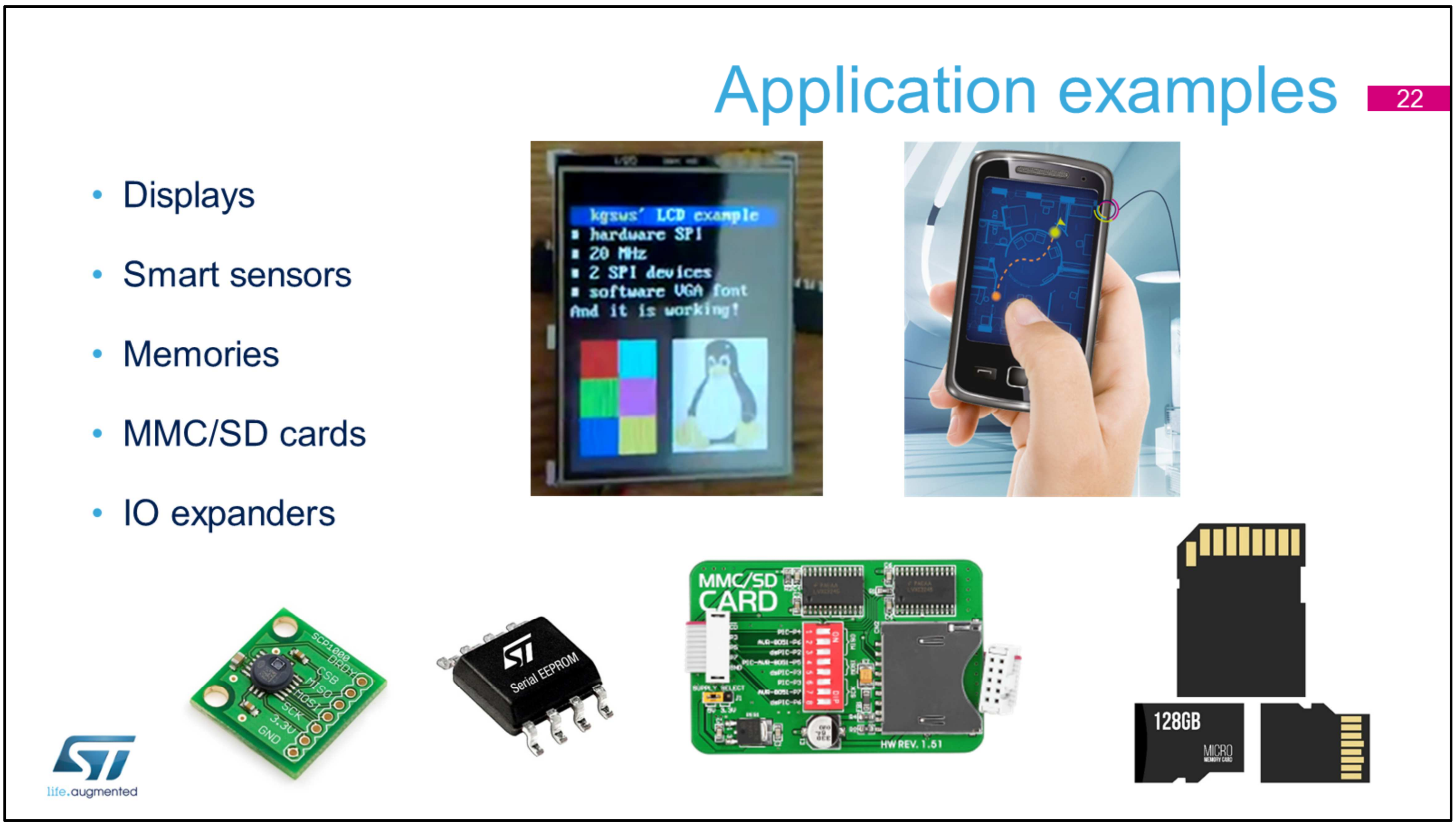
## Performance

- Actual rate of communication depends on:
  - SPI bus capacity load (number of connected devices, input capacitance, length of wires)
  - GPIO internal bonding, their configuration, VDD level and ambient temperature
  - SPI clock signal duty ratio
  - Set-up and hold times provided / required for data
  - Software capability and sufficient performance capacity to control continuous data flow
- Real performance
  - Maximum speed in Master mode is 133 MHz
  - Maximum speed in Slave mode is 150 MHz for receiver and 31 MHz for transmitter
  - These maximum values are not achievable by all the SPI instances

life.augmented

The SPI performance depends mainly on the applied clock. At a minimum, the clock frequency should be twice the required communication frequency. The actual rate of communication can be decreased by application factors. The user has to consider SPI bus loads such as the number of nodes, the connection distance, the input capacitance, as well as the GPIO settings. Fast GPIO mode should be applied on the data and clock signals. Lower power supply voltage and extreme ambient temperatures slow down edges. Sometimes slower data hold or setup time requirements have to be respected between nodes. Applications can’t always manage the fast data flow due to frequent servicing of exceptions.

The DMA capacity has to be considered as well as the number of DMA channels used by the system, frequent interrupt services or execution of non-interruptible instructions.



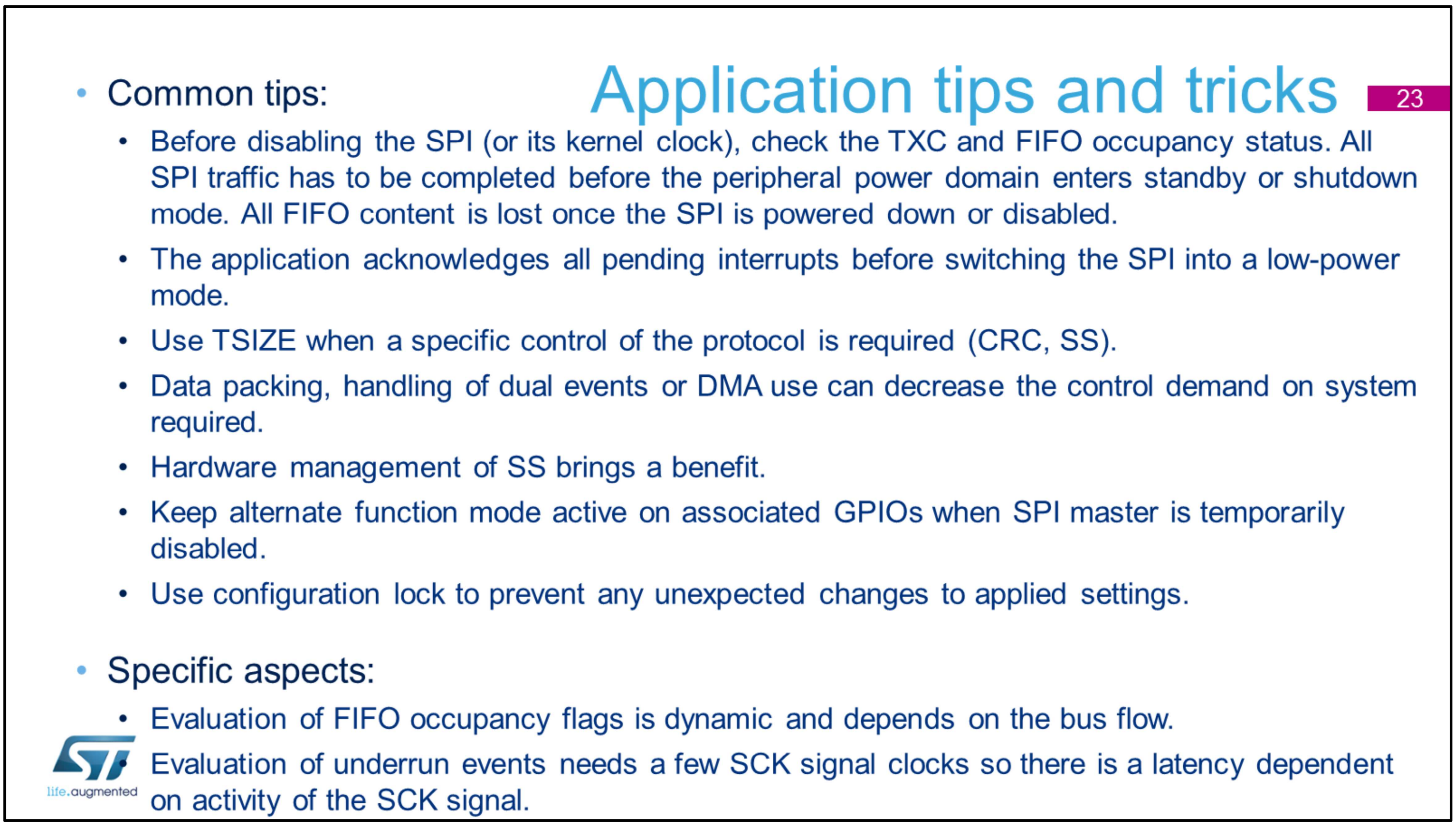
## Application examples

22

- Displays
- Smart sensors
- Memories
- MMC/SD cards
- IO expanders

kgsws' LCD example
hardware SPI
20 MHz
2 SPI devices
software VGA font
And it is working!
SCP1000
Serial EEPROM
MMC/SD CARD
HW REV. 1.61
128GB
MICRO
life.augmented

The SPI can be used in a wide range of applications where a simple data transfer is required without the need for a complex communication protocol. Secured transfers are also supported when used with smart cards.



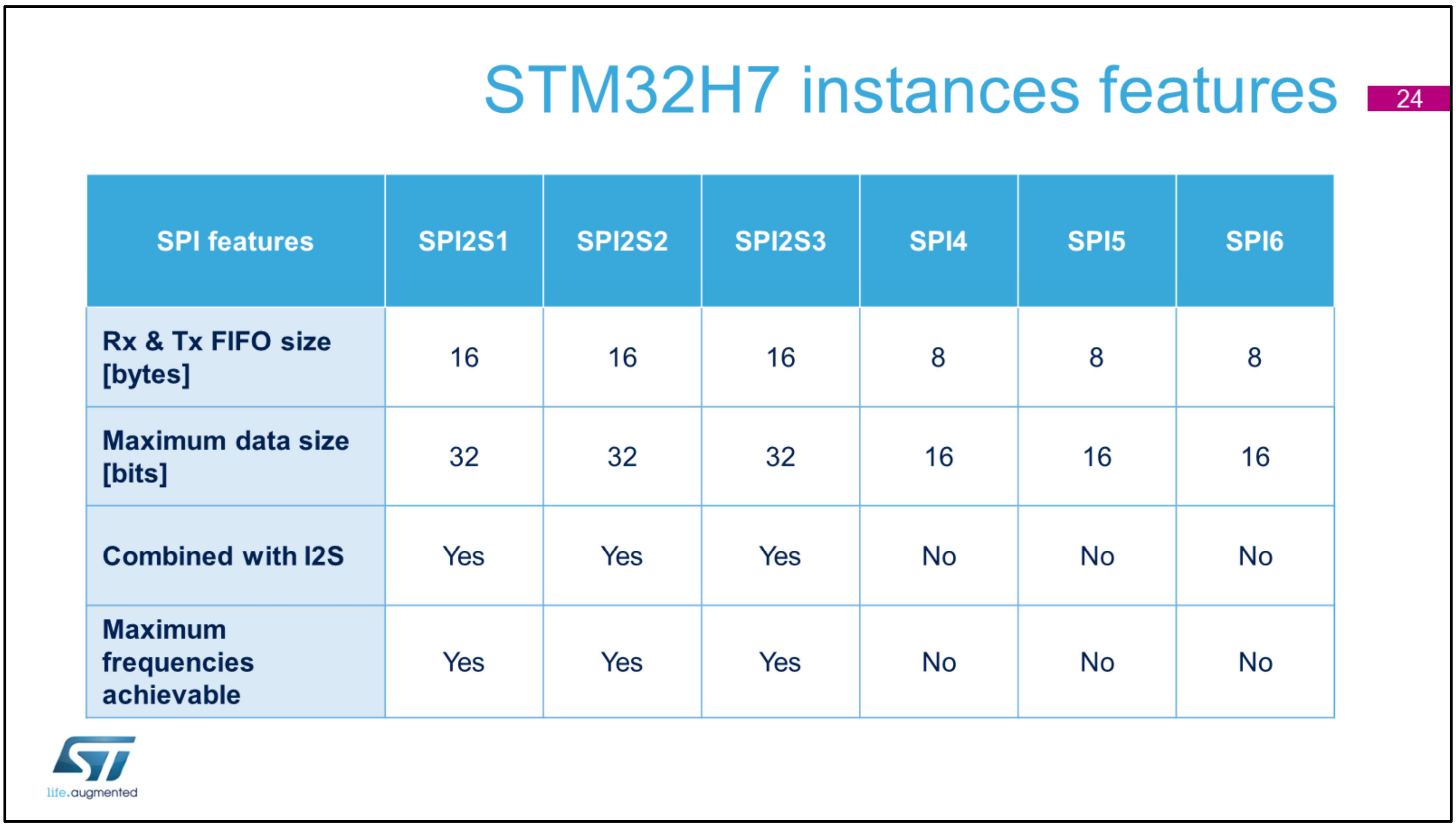
## Application tips and tricks

- Common tips:
  - Before disabling the SPI (or its kernel clock), check the TXC and FIFO occupancy status. All SPI traffic has to be completed before the peripheral power domain enters standby or shutdown mode. All FIFO content is lost once the SPI is powered down or disabled.
  - The application acknowledges all pending interrupts before switching the SPI into a low-power mode.
  - Use TSIZE when a specific control of the protocol is required (CRC, SS).
  - Data packing, handling of dual events or DMA use can decrease the control demand on system required.
  - Hardware management of SS brings a benefit.
  - Keep alternate function mode active on associated GPIOs when SPI master is temporarily disabled.
  - Use configuration lock to prevent any unexpected changes to applied settings.
- Specific aspects:
  - Evaluation of FIFO occupancy flags is dynamic and depends on the bus flow.
  - Evaluation of underrun events needs a few SCK signal clocks so there is a latency dependent on activity of the SCK signal.

life.augmented

Here are some helpful tips:
The user should be aware that traffic on the bus may still be ongoing even if the DMA transaction is completed or the transmit FIFO becomes empty.
That is why the user has to carefully check the peripheral status and follow the suggested procedures before disabling the SPI or placing it in standby or shut down mode. Use the data size control if you need specific control like CRC or slave select signal handling by hardware.
The use of the DMA, data packing or handling dual event at full duplex mode can increase the system's overall performance. These features can help especially when data frames are short and a fast, continuous communication flow is required.
Hardware management of the Slave Select signal is not quite necessary in a single-master/single-slave pair, but it can help synchronize the data flow and prevent conflicts in a Multi-master system. When the SPI master has to be

disabled temporary for any reason, user can prevent any glitches on the associated outputs working at alternate function mode by keeping them forced under control. Then the GPIOs are kept at the state corresponding to the SPI idle configuration. The user should keep all the configuration and settings locked to prevent any accidental changes.
There are some additional specific aspects which should be taken into account when designing an SPI network:
Evaluation of FIFO occupancy flags is dynamic and depends on the bus flow. That is why the event service has to be applied when the complete data packet, which corresponds to the FIFO threshold, is finished. Once the complete packet service is finished, either read from or written in the FIFO, the occupancy flag can be tested again.
Slave internal logic is clocked from an external SCK pin. Certain flags need a few periods of SCK signal cycling to be evaluated and can’t be evaluated as long as the SCK signal stays at idle.

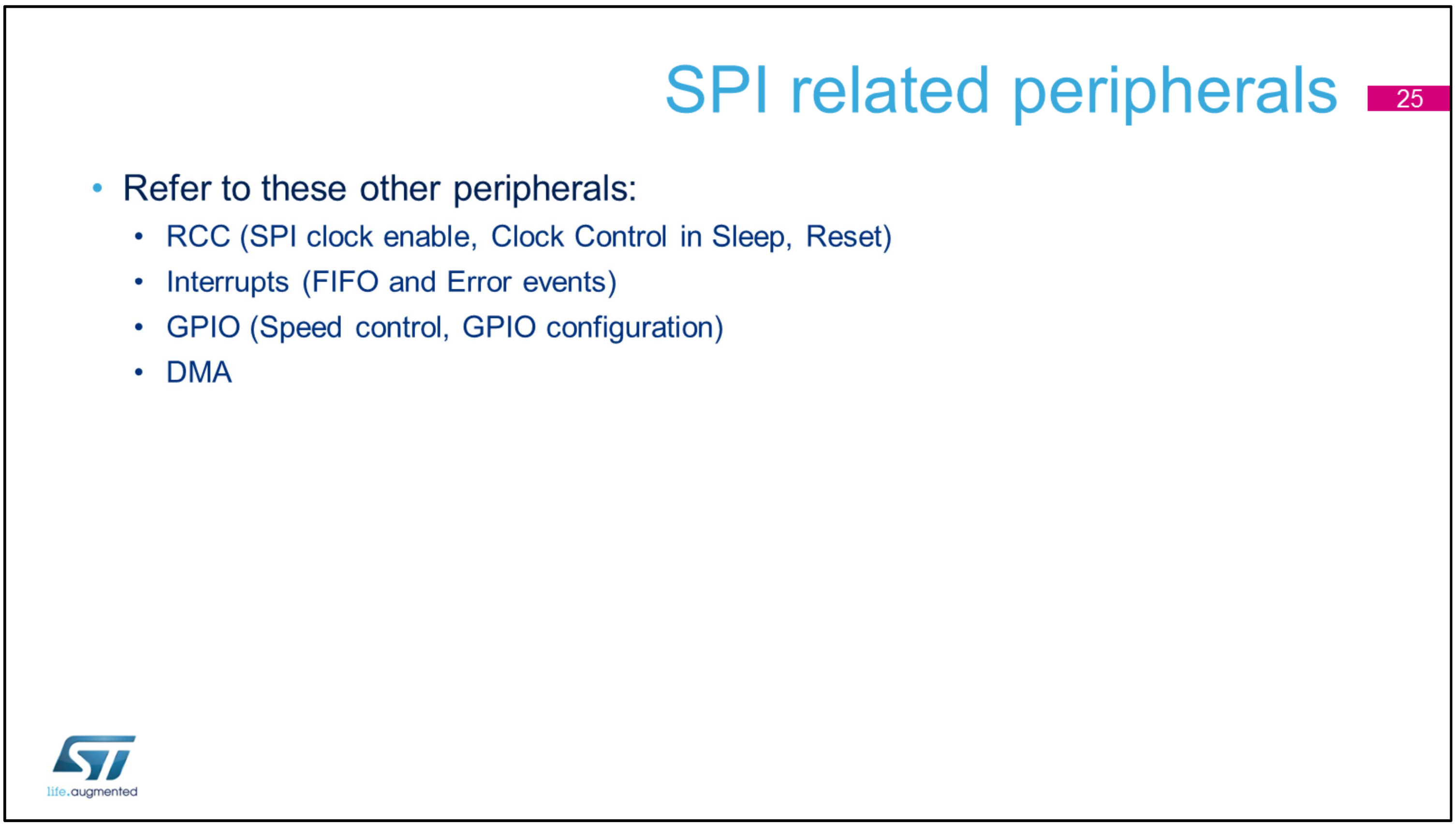


## STM32H7 instances features

| SPI features | SPI2S1 | SPI2S2 | SPI2S3 | SPI4 | SPI5 | SPI6 |
|---|---|---|---|---|---|---|
| Rx & Tx FIFO size [bytes] | 16 | 16 | 16 | 8 | 8 | 8 |
| Maximum data size [bits] | 32 | 32 | 32 | 16 | 16 | 16 |
| Combined with I2S | Yes | Yes | Yes | No | No | No |
| Maximum frequencies achievable | Yes | Yes | Yes | No | No | No |

life.augmented

There are six SPI instances within the STM32H7, and each support all the features presented so far. SPI2S1, SPI2S2 and SPI2S3 are multiplexed with I2S interface and have double extended FIFOs and data size registers.



## SPI related peripherals

- Refer to these other peripherals:
  - RCC (SPI clock enable, Clock Control in Sleep, Reset)
  - Interrupts (FIFO and Error events)
  - GPIO (Speed control, GPIO configuration)
  - DMA

life.augmented

Refer to these other trainings which are linked directly to the SPI. Users should be familiar with all the peripherals that can affect the behavior of the SPI.

## References

- For more details, please refer to following resources
  - STM32H7 reference manual and datasheet
  - AN4286 - SPI protocol used in the STM32 bootloader
  - AN3364 - Migration and compatibility guidelines for STM32 microcontroller applications
  - Web (connection examples, available monitoring tools)

life.augmented

There are some dedicated SPI application notes. To learn more about general SPI connections and interface issues, there are many web pages, as well as SPI bus monitoring tools available.  Many digital oscilloscopes support direct reading and analysis of data and clock signals on the SPI bus.