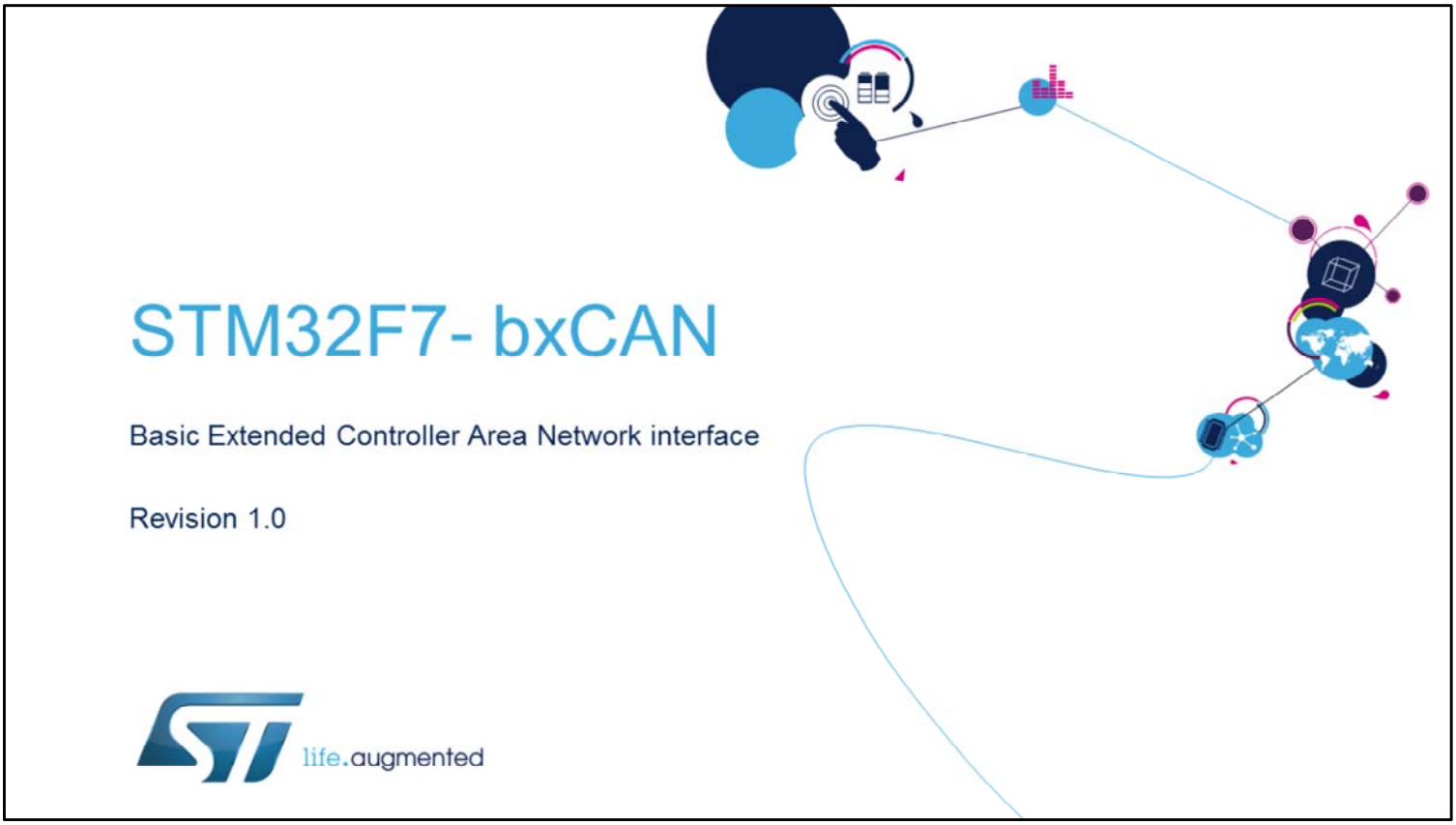
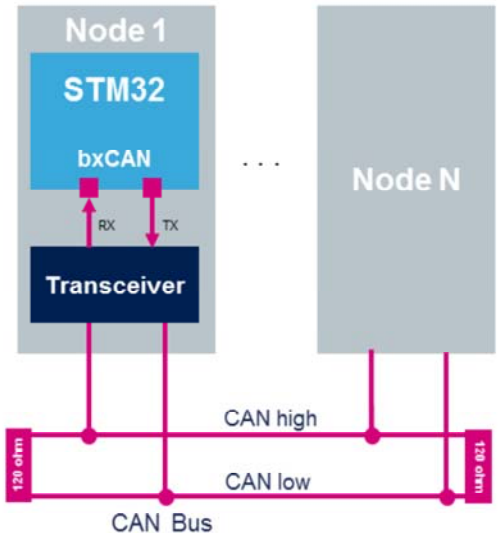


Hello and welcome to this presentation of the STM32F7 Basic Extended Controller Area Network interface. It will cover the main features of this interface, which is widely used to connect the microcontroller to a CAN network,

Overview



- Provides communication interface with external CAN transceiver via two pins

Application benefits

- Multi-master concept
- Object-oriented communication
- Real-time capability
- Low message transfer latency
- System wide message consistency

The controller area network (CAN) is a standard serial differential bus broadcast interface, allowing the microcontroller to communicate with external devices connected to the same network bus.
The CAN interface is highly configurable, allowing nodes to easily connect using two wires.
Applications benefit from a Multi-master concept with message priority, object-oriented communication (no node addressing, but content identification), real-time capability with low message transfer latency and system wide message consistency (error detection & management mechanism).

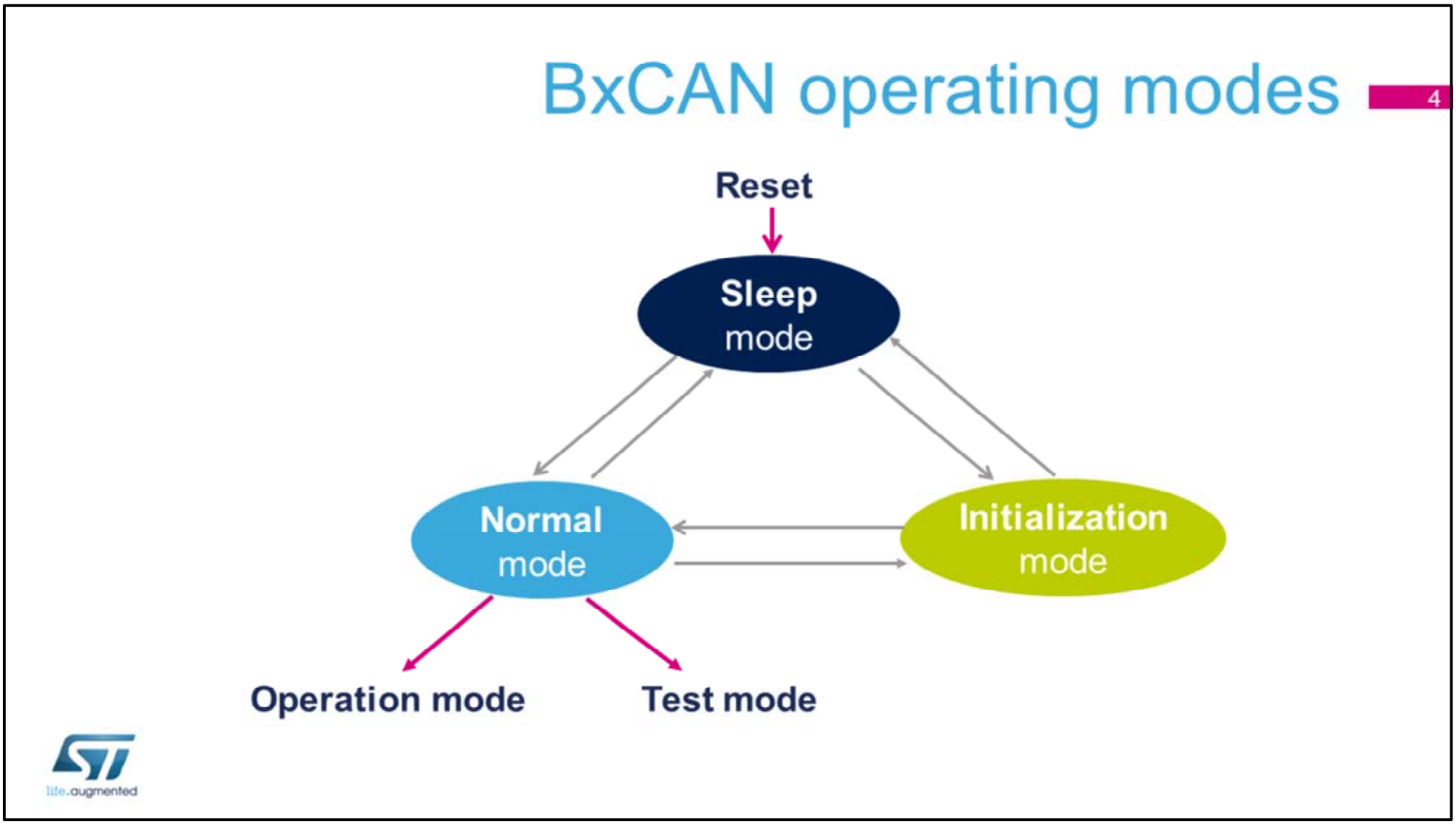
BxCAN features

3

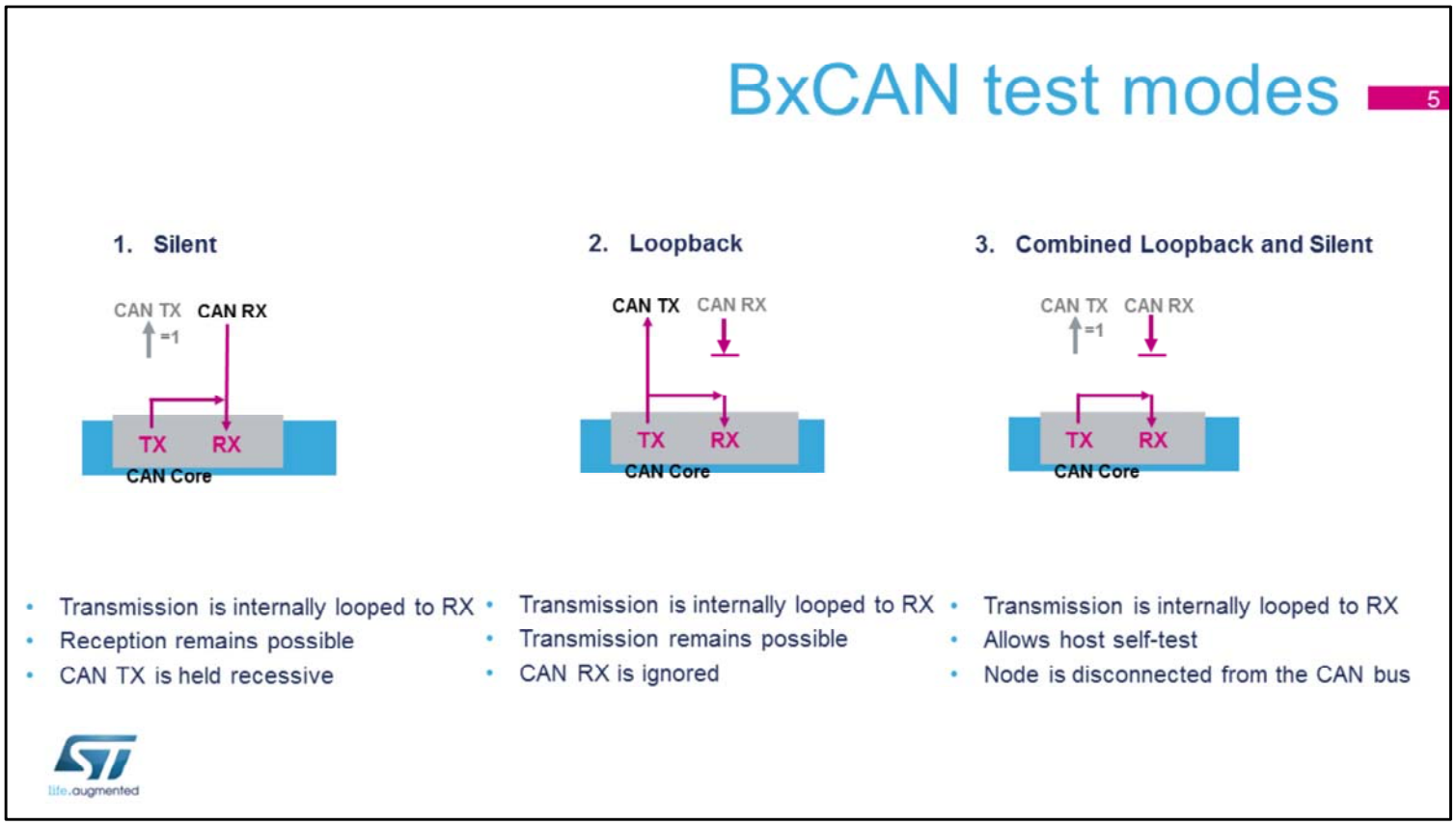
- Supports CAN protocol versions 2.0 A and B Active
- Bit rates up to 1 Mbit/s
- Three transmit mailboxes with configurable transmit priority option
- Two receive FIFOs with three stages with 28 scalable filter banks
- 4 dedicated interrupt vectors: transmit interrupt, FIFO 0 interrupt, FIFO 1 interrupt and status change error interrupt

life.augmented

The STM32 CAN peripheral supports the Basic Extended CAN protocol versions 2.0 A and B Active with a maximum bit rate of 1 Mbit/s. The BxCAN includes 3 transmit mailboxes with a configurable transmit priority option and 2 receive FIFOs with three stages with 28 scalable filter banks. This allows the CAN to efficiently manage a high number of incoming and outgoing messages with a minimum CPU load.
The BxCAN peripheral also manages four dedicated interrupt vectors.

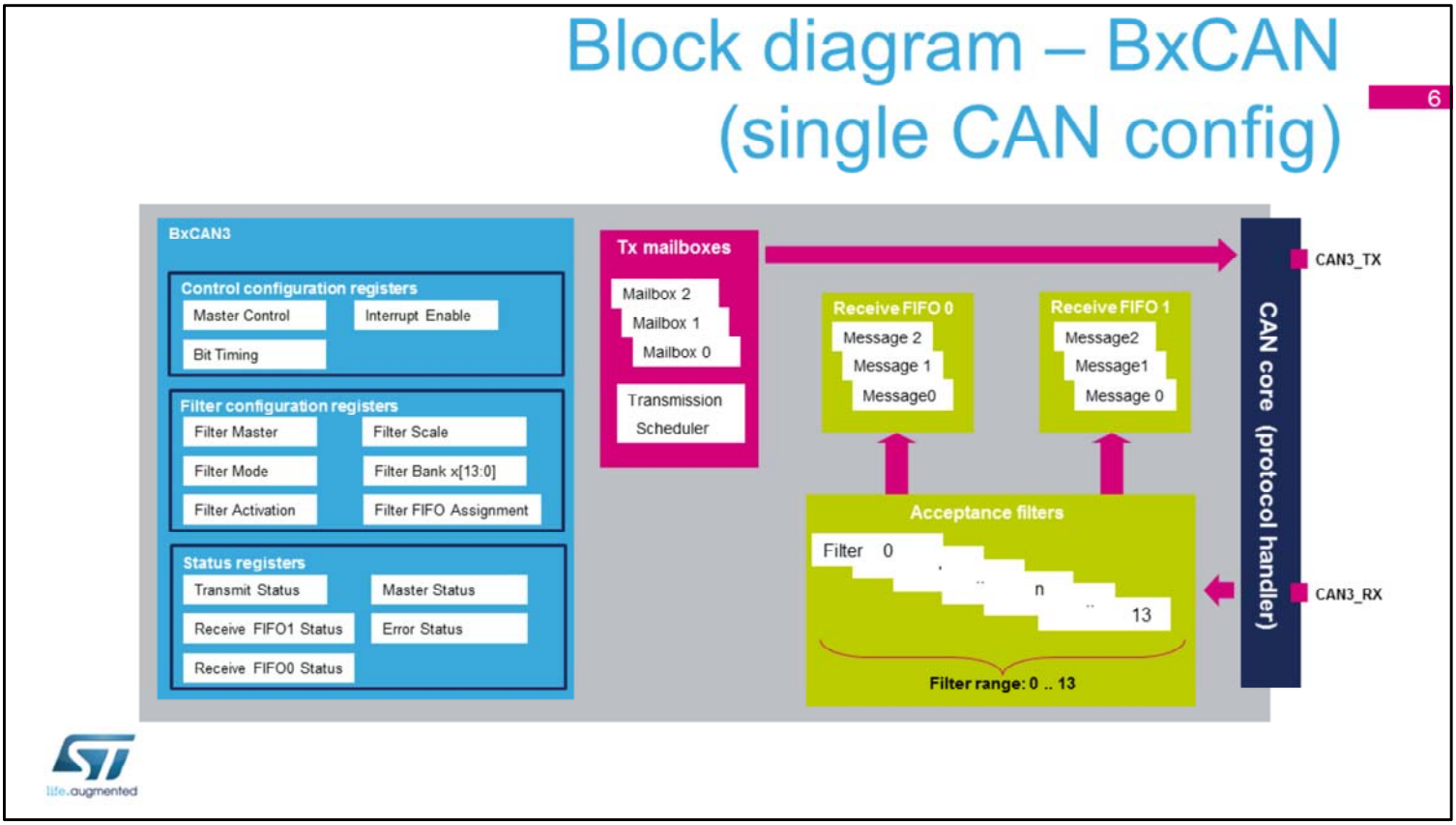


The BxCAN has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, the BxCAN is in Sleep mode which operates at a lower power (Note: In Sleep mode, the internal pull-up is active on pin CANTX). The BxCAN enters Initialization mode via software to allow the configuration of the peripheral. Before entering Normal mode, the BxCAN must synchronize with the CAN bus, so it waits until the bus is idle (this means 11 consecutive recessive bits have been monitored on pin CANRX).
When the CAN is in Normal mode, the user can select whether to run in Operation or Test mode.



The BxCAN supports three test modes:

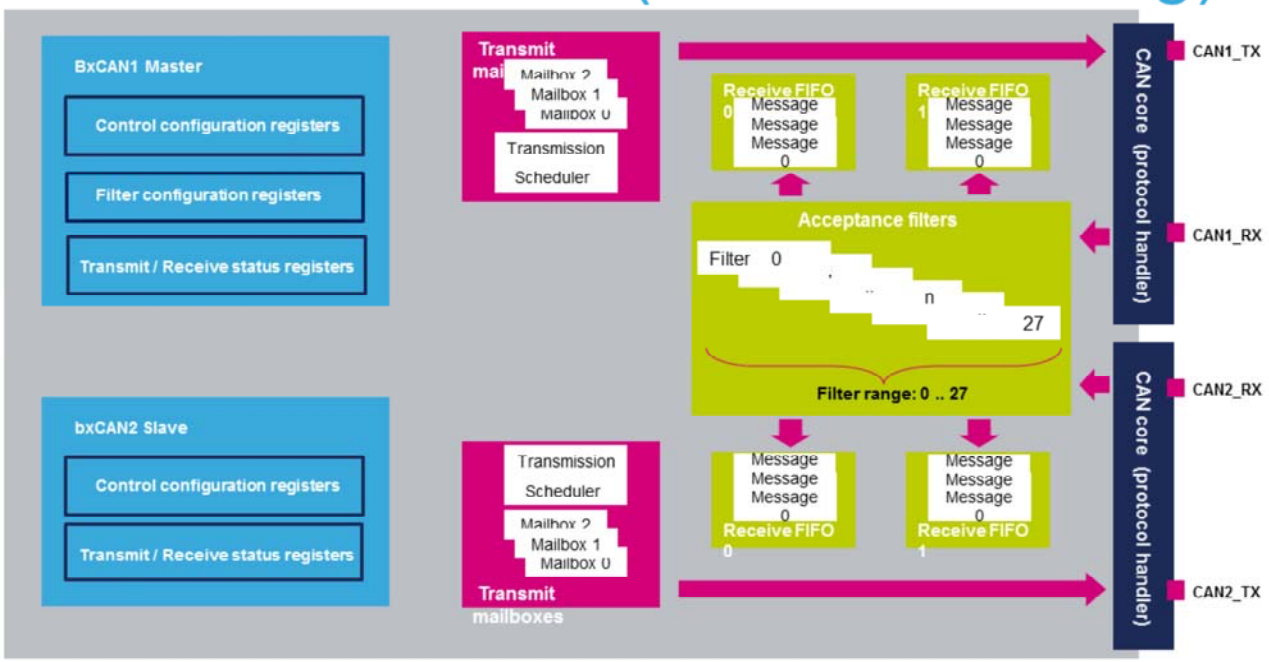
- In Silent mode, the BxCAN is able to receive valid frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. Silent mode can be used to analyze traffic on a CAN bus without affecting it by the transmission of dominant bits.
- In Loop Back mode, the BxCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in a receive mailbox. Loop Back mode is provided for self-test functions.
- In Combined Loop Back and Silent mode, the BxCAN can be tested in Loop Back mode but without affecting the running CAN system connected to the CANTX and CANRX pins.



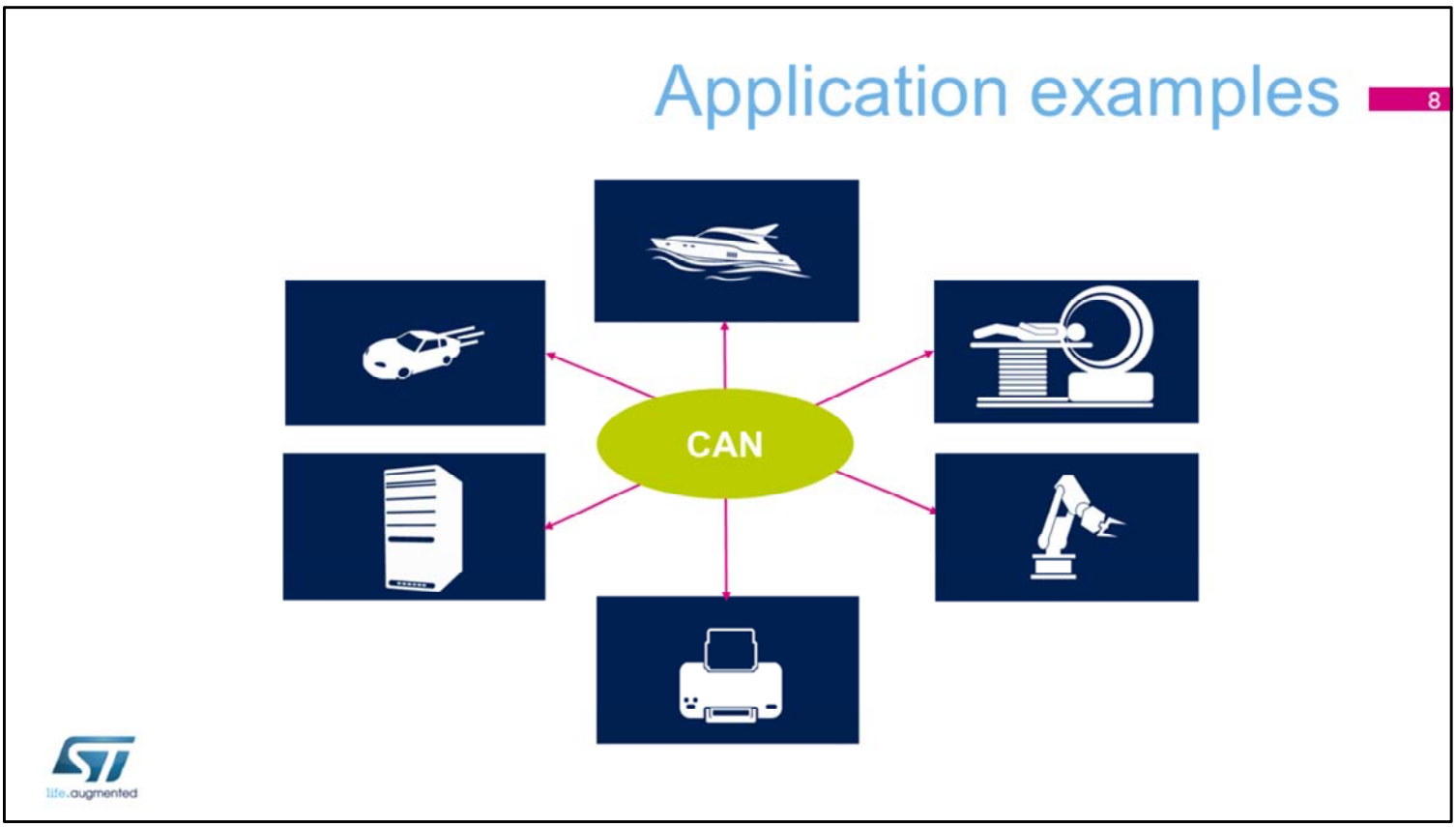
This simplified block diagram of the BxCAN in a single CAN configuration shows its basic functional and control features.

- Three types of registers: Control configuration registers, filter configuration registers and status registers.
- Three transmit mailboxes are provided to the software for setting up messages. The Transmission Scheduler decides which mailbox has priority to be transmitted first.
- The BxCAN provides 14 scalable and configurable identifier filters for selecting the incoming messages the application needs and discarding the others.
- Two receive FIFOs: FIFO 0 and FIFO 1 are used by hardware to store incoming messages. Each FIFO can store three complete messages. The FIFOs are completely managed by hardware.

Block diagram – BxCAN (dual CAN config)



This simplified block diagram of the BxCAN in dual CAN configuration shows the shared 28 Acceptance filters between the two BxCAN modules.
The user can assign each filter to either FIFO 0 or FIFO 1, and configure each filter for Identifier Mask or List mode.



The controller area network (CAN bus) was originally designed for automotive applications, but is now also used in many other contexts.

BxCAN Interrupts

Interrupt event	Description
Transmit interrupt	Set when mailbox is ready to accept a new message.
FIFO 0 interrupt	Set when message is received at FIFO 0 (Full or Overrun).
FIFO 1 interrupt	Set when message is received at FIFO 1 (Full or Overrun).
Error and Status change interrupts	Set on Error, Wakeup, or Entry into Sleep mode.

life.augmented

Here is a summary of CAN interrupt events: Transmit, receive buffers for FIFO 0 and FIFO 1, and error and status change interrupts.

Low-power modes

Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Frozen. Peripheral registers content is retained.
Standby	Powered-down. Peripherals must be reinitialized after exiting Standby mode.

life.augmented

Here is an overview of the CAN low-power configuration modes. The device is not able to perform any communications in Stop or Standby modes. It is important to ensure that all CAN traffic is completed before the peripheral enters Stop or Standby modes.

Debug support

11

- The Debug module allows the selection of the BxCAN behavior when the CPU is halted
  - BxCAN reception continues in Normal mode. This may cause reception overrun errors.
  - BxCAN receive registers and FIFOs are frozen.

The Debug support module allows the user to select the BxCAN behavior when the Core is halted (i.e. stopped at a breakpoint). The default configuration allows the BxCAN reception to continue as normal and this may lead to reception overrun errors. The other debug option is to block updates of the BxCAN receive registers and FIFOs until the core is running.

CAN related peripherals

12

- Refer to these other peripherals:
  - **Reset and clock controller** (RCC) for more information about the CAN clock control and enable/reset.
  - **Nested vectored interrupt controller (NVIC)** for more information about the mapping of the BxCAN's interrupts.
  - **General-purpose I/Os** (GPIO) for more information about the BxCAN's input and output pins.
  - **Debug Support** (DBG) for more information about the BxCAN's behavior when the CPU is halted.

For additional information, refer to the training modules for these peripherals which may affect BxCAN behavior:

•**Reset and clock controller** (RCC) for more information about the CAN clock control and enable/reset.

•**Interrupts** for more information about the mapping of the BxCAN's interrupts.

•**General-purpose I/Os** (GPIO) for more information about the BxCAN's input and output pins.

•**Debug Support** (DBG) for more information about the BxCAN's behavior in debug mode.

References

13

- For more details, please refer to following resources
  - Application note AN3154: Description of the CAN protocol used in the STM32 boot loader
  - Application note AN3364: Migration and compatibility guidelines for STM32 microcontroller applications
  - Web (connection examples, available monitoring tools, etc.)

life.augmented

Application notes covering the CAN topic are available on www.st.com. To learn more about the CAN interface, you can also visit a wide range of web pages discussing the CAN communication protocol and bus monitoring tools. Many digital oscilloscopes support direct reading and analysis of data transmitted over the CAN bus.