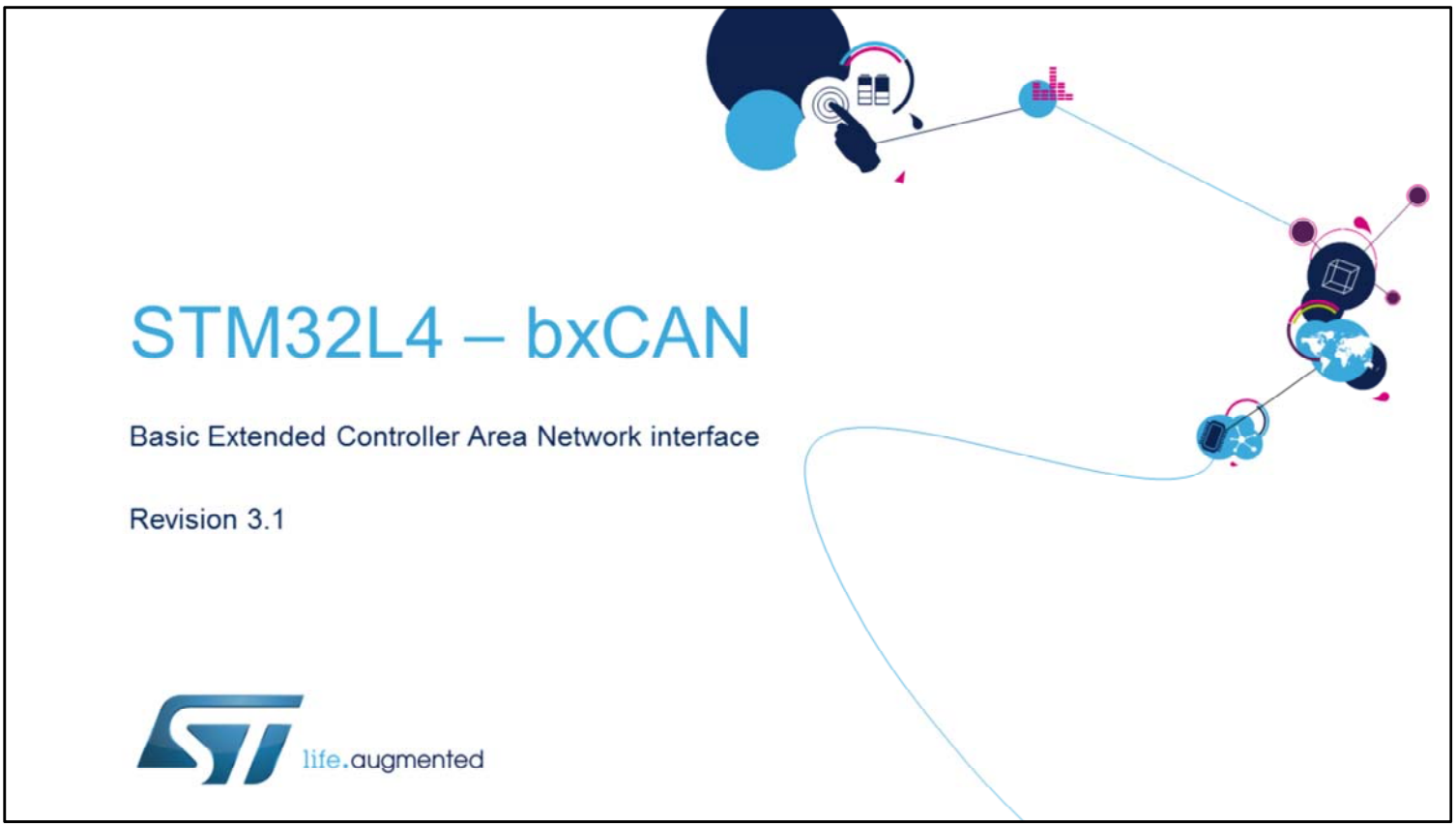
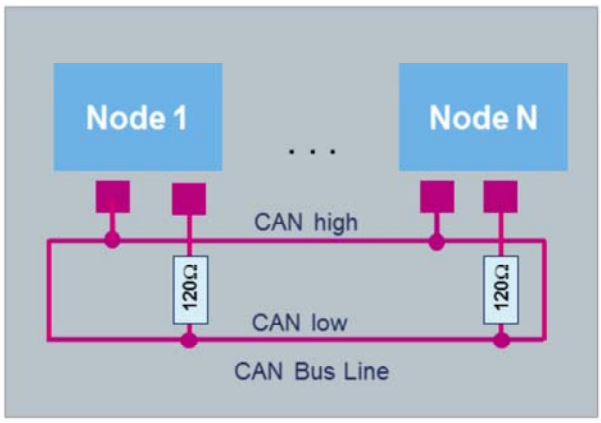


Hello and welcome to this presentation of the STM32L4 Basic Extended Controller Area Network interface. It will cover the main features of this interface, which is widely used to connect multiple devices to the microcontroller.

Overview



- Provides simple communication interface with external devices via two pins
  - Highly configurable
  - Supports standard asynchronous serial protocols

Application benefits

- Low cost: Only two pins needed
- High-speed, real-time communication.
- Robust in electromagnetically noisy environments

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 the low-cost, robust, and direct asynchronous serial asynchronous interface.

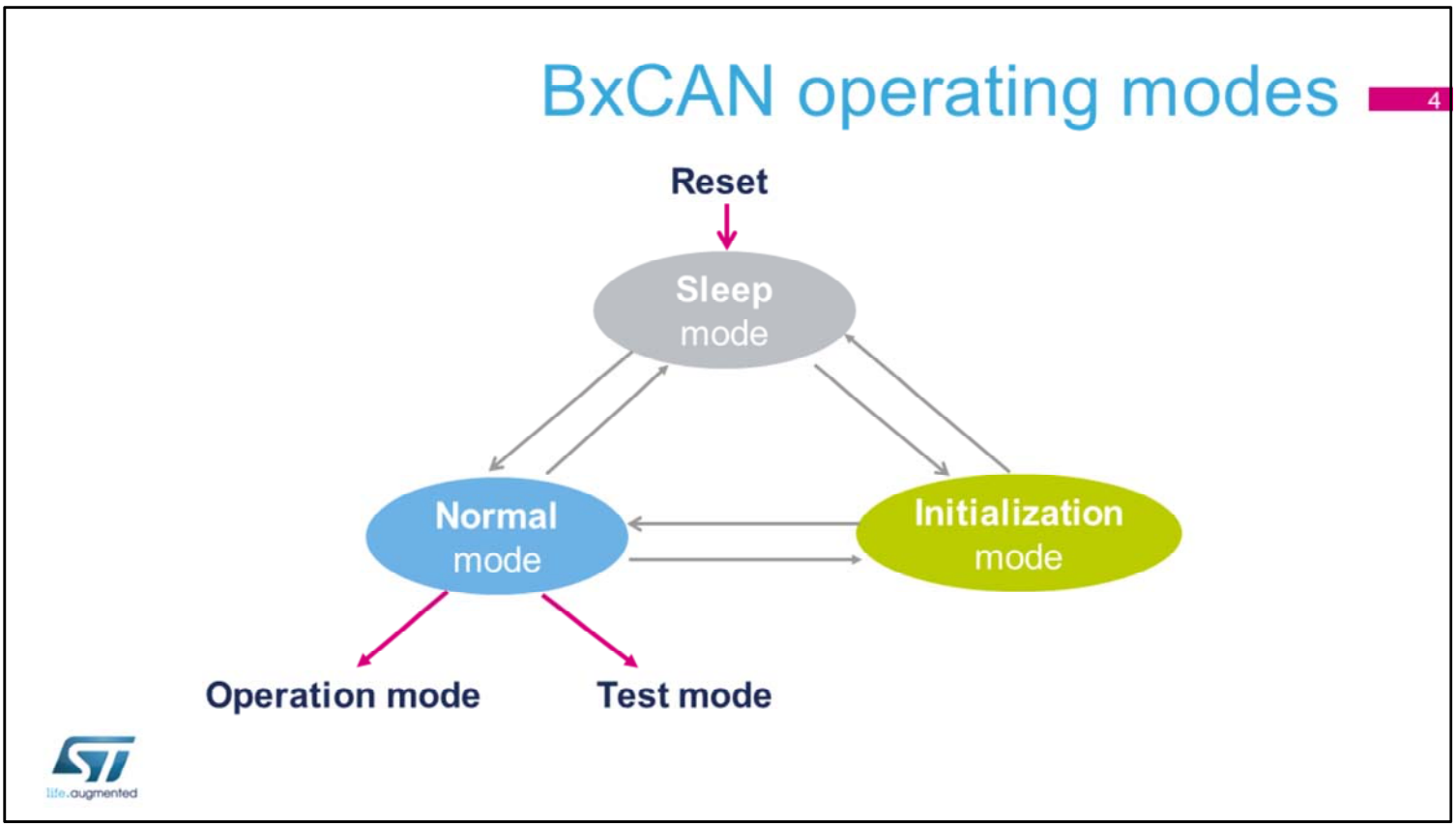
BxCAN features

3

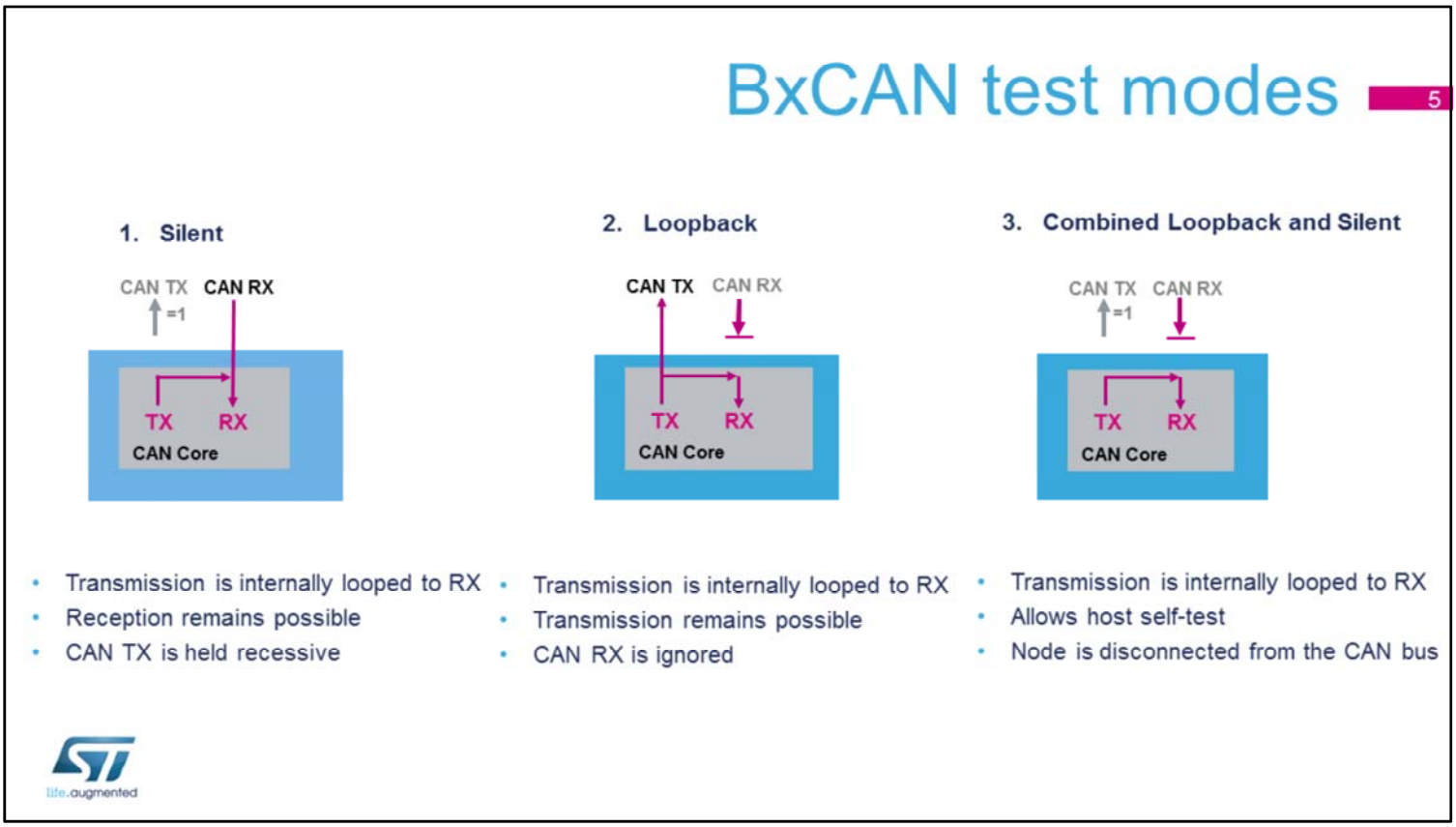
- Supports CAN protocol versions 2.0 A and B Active
- Bit rates up to 1Mbit/s
  - Three transmit mailboxes with configurable transmit priority option
- Two receive FIFOs with three stages with 14 scalable filter banks
- 4 dedicated interrupt vectors: transmit interrupt, FIFO0 interrupt, FIFO1 interrupt 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 14 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.



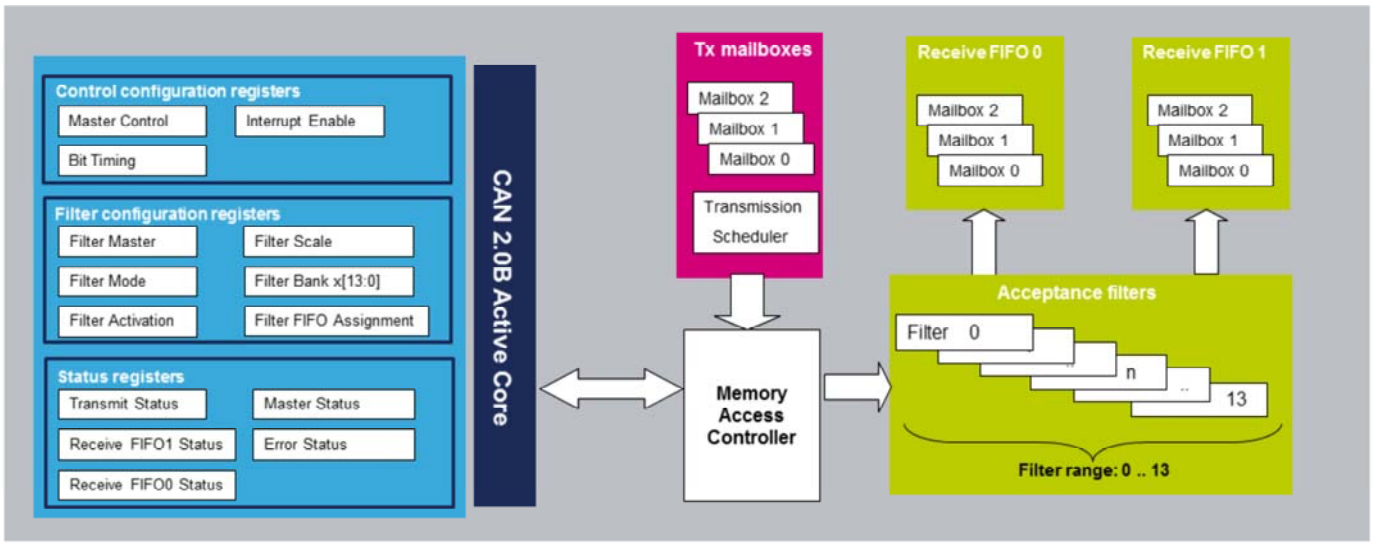
The BxCAN has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, the BxCAN is in Sleep mode which operates at a lower power (Note: In Sleep mode, the internal pull-up is active on pin CANTX). The BxCAN enters Initialization mode via software to allow the configuration 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:

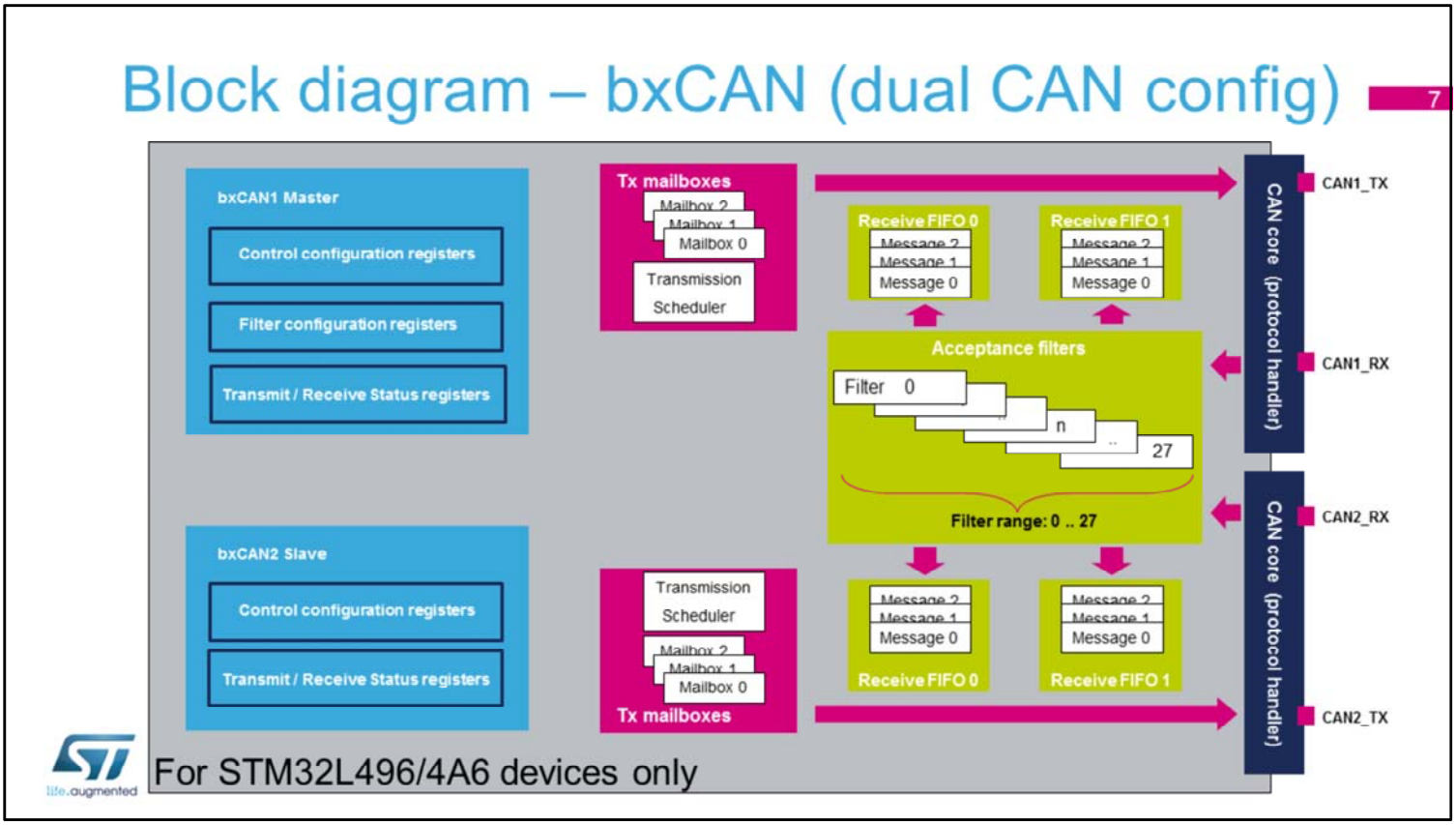
- In Silent mode, the BxCAN is able to receive valid frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. Silent mode can be used to analyze traffic on a CAN bus without affecting it by the transmission of dominant bits.
- In Loop Back mode, the BxCAN treats its 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.
- Combined Loop Back and Silent mode. In this mode, the BxCAN can be tested in Loop Back mode but without affecting the running CAN system connected to the CANTX and CANRX pins.

Block diagram – bxCAN

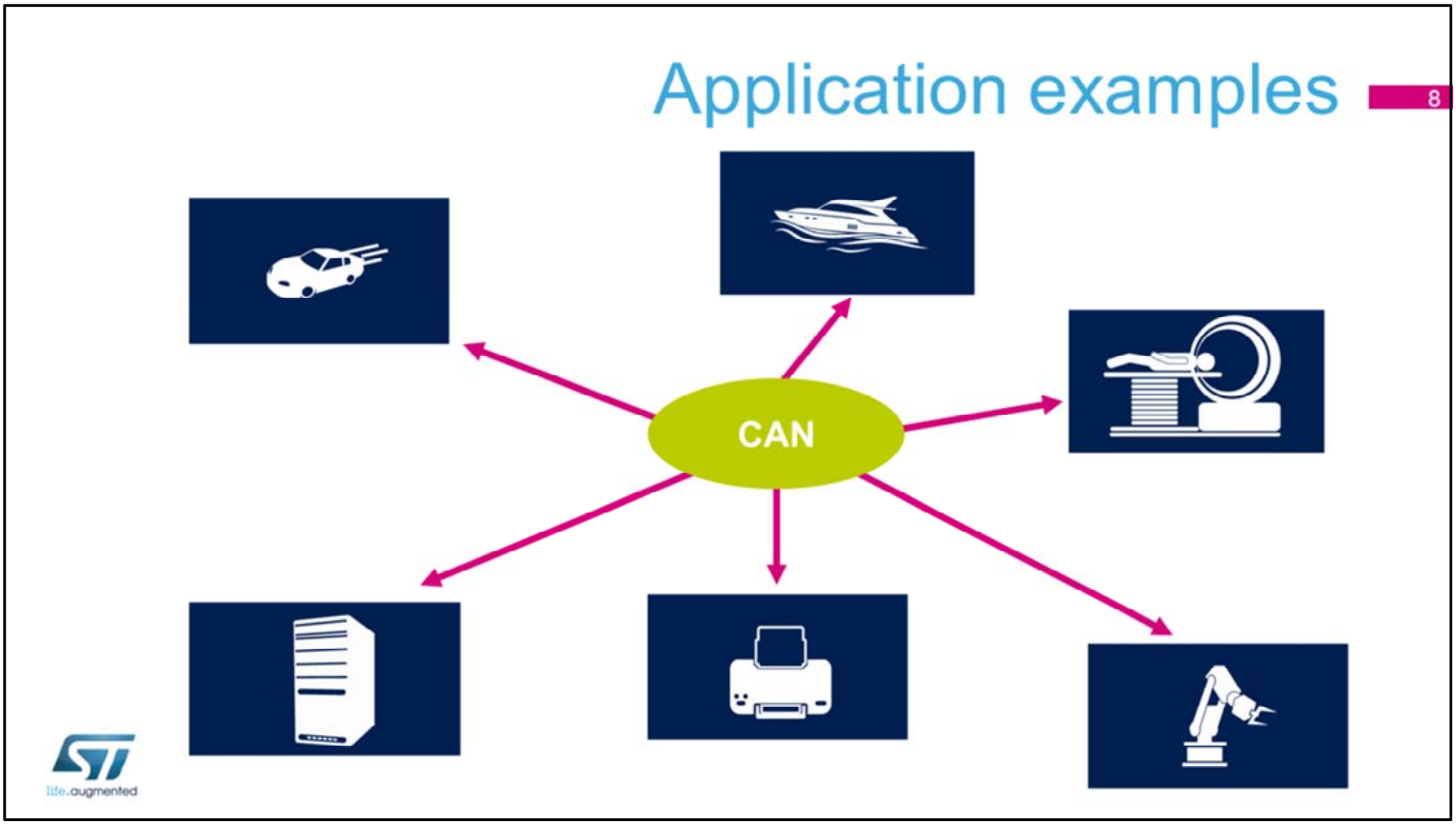


This simplified block diagram of the CAN shows its basic functional and control features.

- 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: FIFO0 and FIFO1 are used by hardware to store incoming messages. Each FIFO can store three complete messages. The FIFOs are managed completely by hardware.



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.
Note that this dual CAN configuration is only available for STM32L496/4A6 devices.



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

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 FIFO0 (Full or Overrun)
FIFO 1 interrupt	Set when message is received at FIFO1 (Full or Overrun)
Error and Status change interrupts	Set on Error, Wakeup, or Entry into Sleep mode

Here is a summary of CAN interrupt events: Transmit, receive buffers for FIFO0 and FIFO1, and error and status change interrupts.

Low-power modes

Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Low-power run	Active.
Low-power sleep	Active. Peripheral interrupts cause the device to exit Low-power sleep mode.
Stop 0/Stop 1	Frozen. Peripheral registers content is retained.
Stop 2	Frozen. Peripheral registers content is retained.
Standby	Powered-down. Peripherals must be reinitialized after exiting Standby mode.
Shutdown	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, Standby or Shutdown modes. It is important to ensure that all CAN traffic is completed before the peripheral enters Stop or Powered-down modes.

CAN related peripherals

- Refer to these other peripherals:
  - 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.

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.

References

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

There are few dedicated application notes with the CAN topic. To learn more about the CAN interface, you can visit a wide range of web pages discussing CAN interface issues and bus monitoring tools. Many digital oscilloscopes support direct reading and analysis of data performed on the CAN bus.