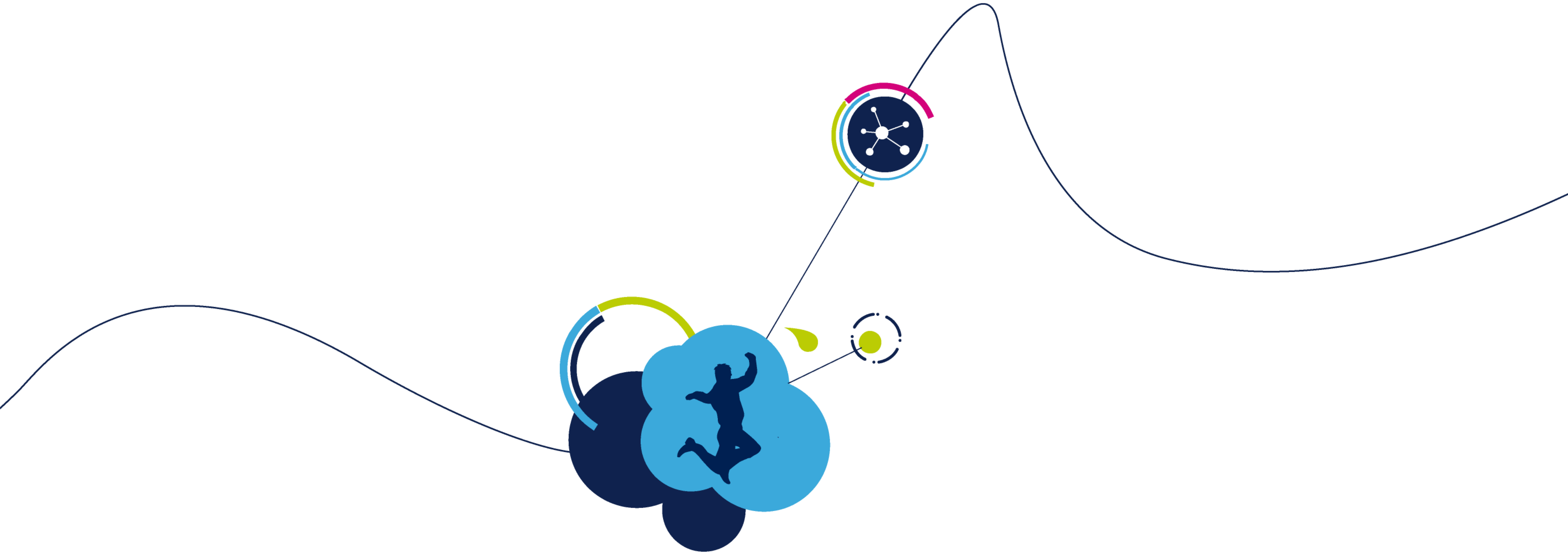
STM32L4 technical training

Controller Area Network (CAN)

Hands-on session



CAN Lab

CAN connectivity – sending and receiving of CAN messages

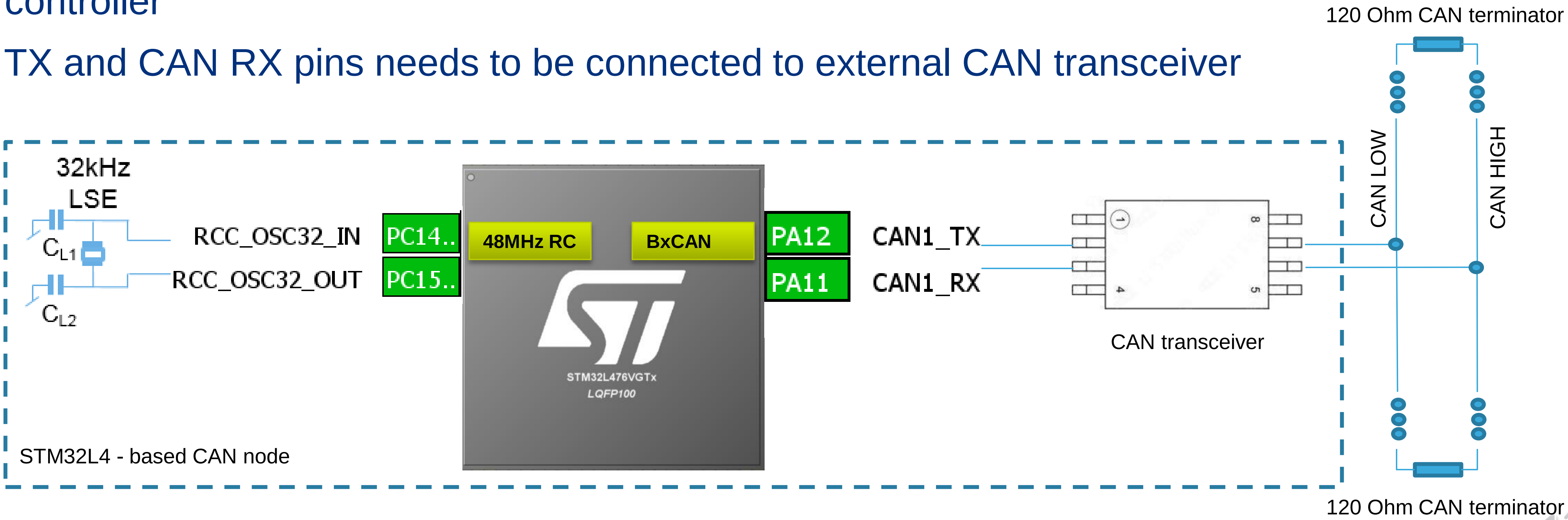
life.augmented

CAN connectivity

- Objective
  - Learn how to configure CAN in CubeMX
  - Learn how to generate code in CubeMX and use HAL functions
- Method
  - Develop an application, which sends CAN messages and receives CAN messages

CAN hardware connection

- STM32L4 as a CAN node
  - BxCAN - embedded CAN controller 2.0A/B
  - Internal RC 48 MHz (MSI – Multi Speed Internal), which can be trimming by LSE (Low Speed External) and can be used as an accurate source of clock signal for CAN controller
  - CAN TX and CAN RX pins needs to be connected to external CAN transceiver



CAN connection

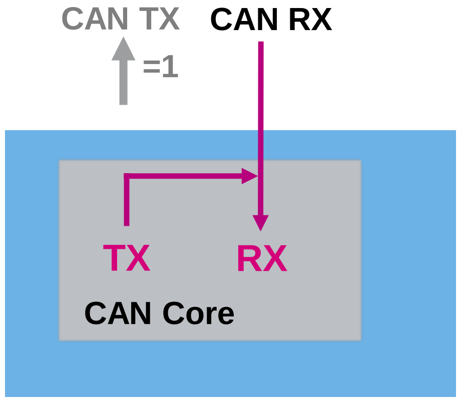
STM32L476RG-Discovery

- STM32L476RG-Discovery is not equipped with a CAN transceiver, which is needed for CAN connectivity in network.

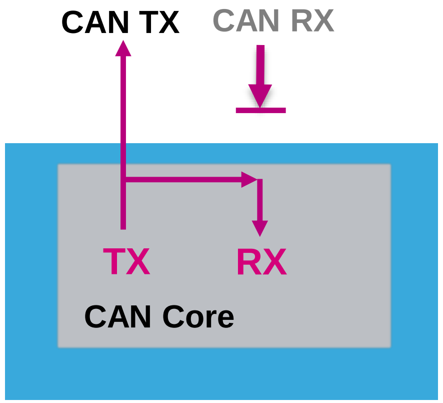
Not ready for CAN network connectivity

- For purpose of CAN evaluation with STM32L476RG-Discovery, one of CAN test modes can be used (silent, loopback or silent+loopback).

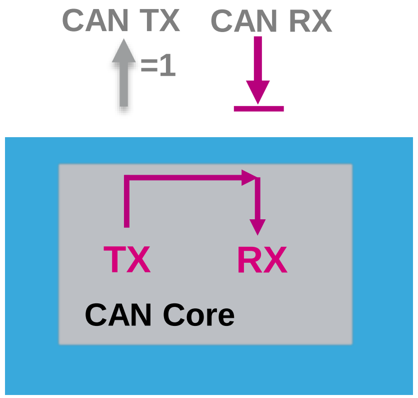
1. Silent



2. Loopback



3. Combined Loopback and Silent

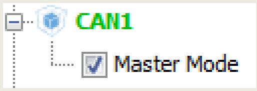


STM32CubeMX

Selecting CAN interface and clock

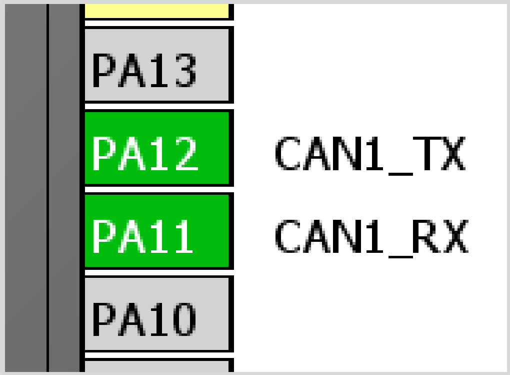
- Create project in STM32CubeMX
  - Menu > File > New Project
  - Select STM32L4 -> STM32L4x6 -> LQFP100 package -> STM32L476VGTx

- Select CAN:
  - Select “**Master Mode**” for **CAN1**

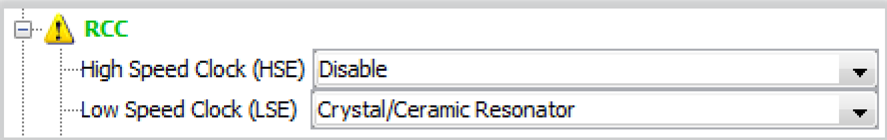


CAN1

☑ Master Mode



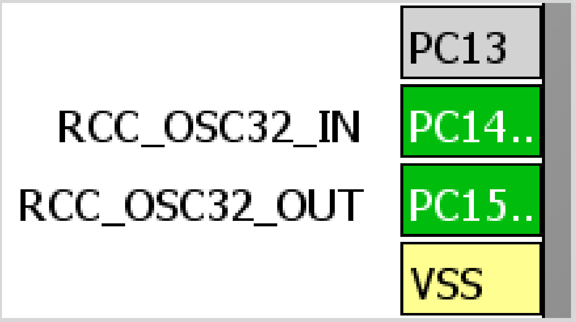
- Select LSE:
  - Select “**Crystal/Ceramic Resonator**” for **Low Speed Clock (LSE)** of **RCC**



RCC

High Speed Clock (HSE) Disable

Low Speed Clock (LSE) Crystal/Ceramic Resonator

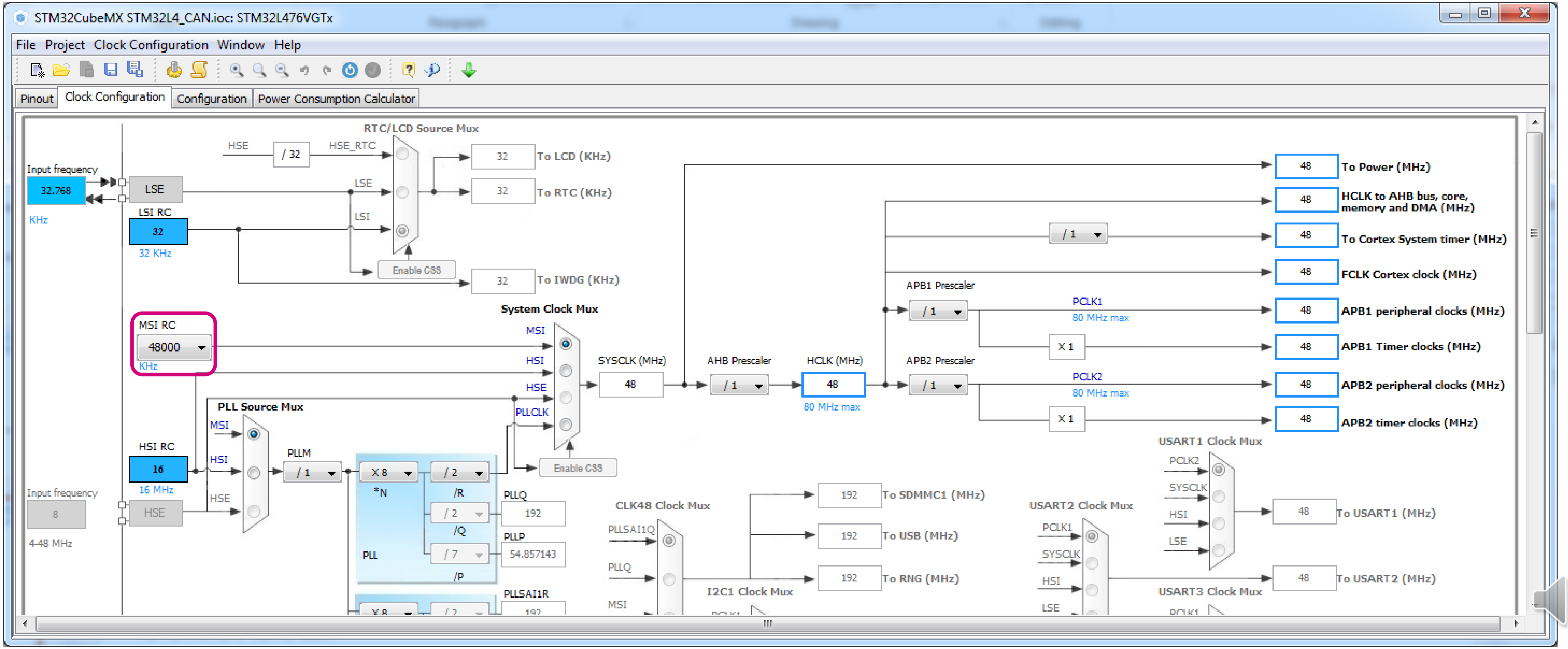


LSE is
needed to
trimm the MSI

STM32CubeMX

clock configuration

- Go to Clock Configuration tab and configure MCU clock system:
  - Change MSI default value (4 MHz) to 48 MHz



STM32CubeMX STM32L4_CAN.ioc: STM32L476VGTx

MSI RC: 48000 KHz

Input frequency (LSE): 32.768 KHz

LSI RC: 32 (32 KHz)

HSI RC: 16 (16 MHz)

Input frequency (HSE): 8 (4-48 MHz)

SYSCLK (MHz): 48

AHB Prescaler: / 1

HCLK (MHz): 48 (80 MHz max)

APB1 Prescaler: / 1

PCLK1: 80 MHz max

APB2 Prescaler: / 1

PCLK2: 80 MHz max

PLLM: / 1

*N: X 8

/R: / 2

/Q: / 2

/P: / 7

PLLQ: 192

PLLP: 54.857143

To LCD (KHz): 32

To RTC (KHz): 32

To IWDG (KHz): 32

To Power (MHz): 48

HCLK to AHB bus, core, memory and DMA (MHz): 48

To Cortex System timer (MHz): 48 (/ 1)

FCLK Cortex clock (MHz): 48

APB1 peripheral clocks (MHz): 48

APB1 Timer clocks (MHz): 48 (X 1)

APB2 peripheral clocks (MHz): 48

APB2 timer clocks (MHz): 48 (X 1)

To SDMMC1 (MHz): 192

To USB (MHz): 192

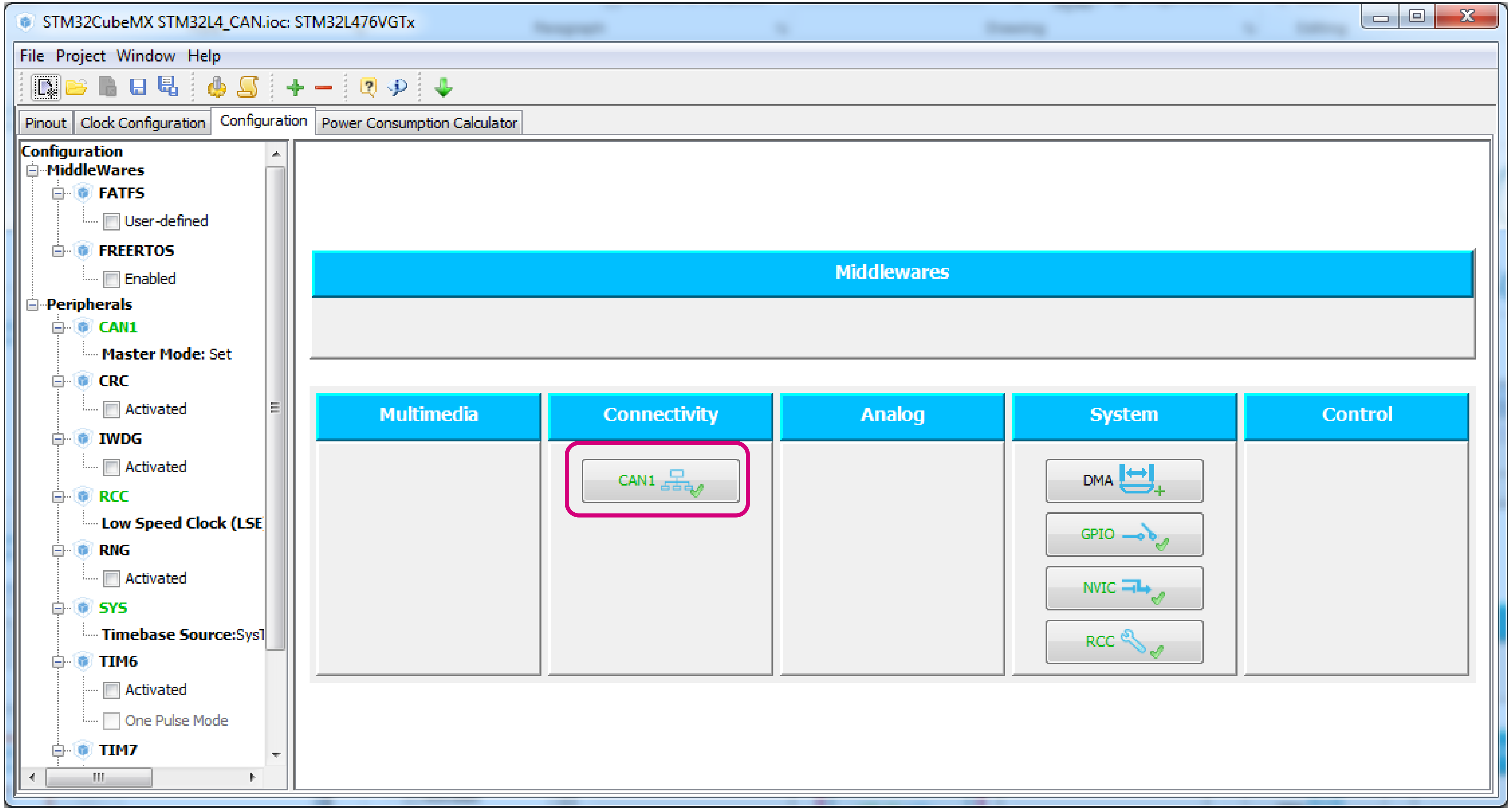
To RNG (MHz): 192

To USART1 (MHz): 48

To USART2 (MHz): 48

STM32CubeMX
Configure CAN

- Go to Configuration tab and select CAN peripheral

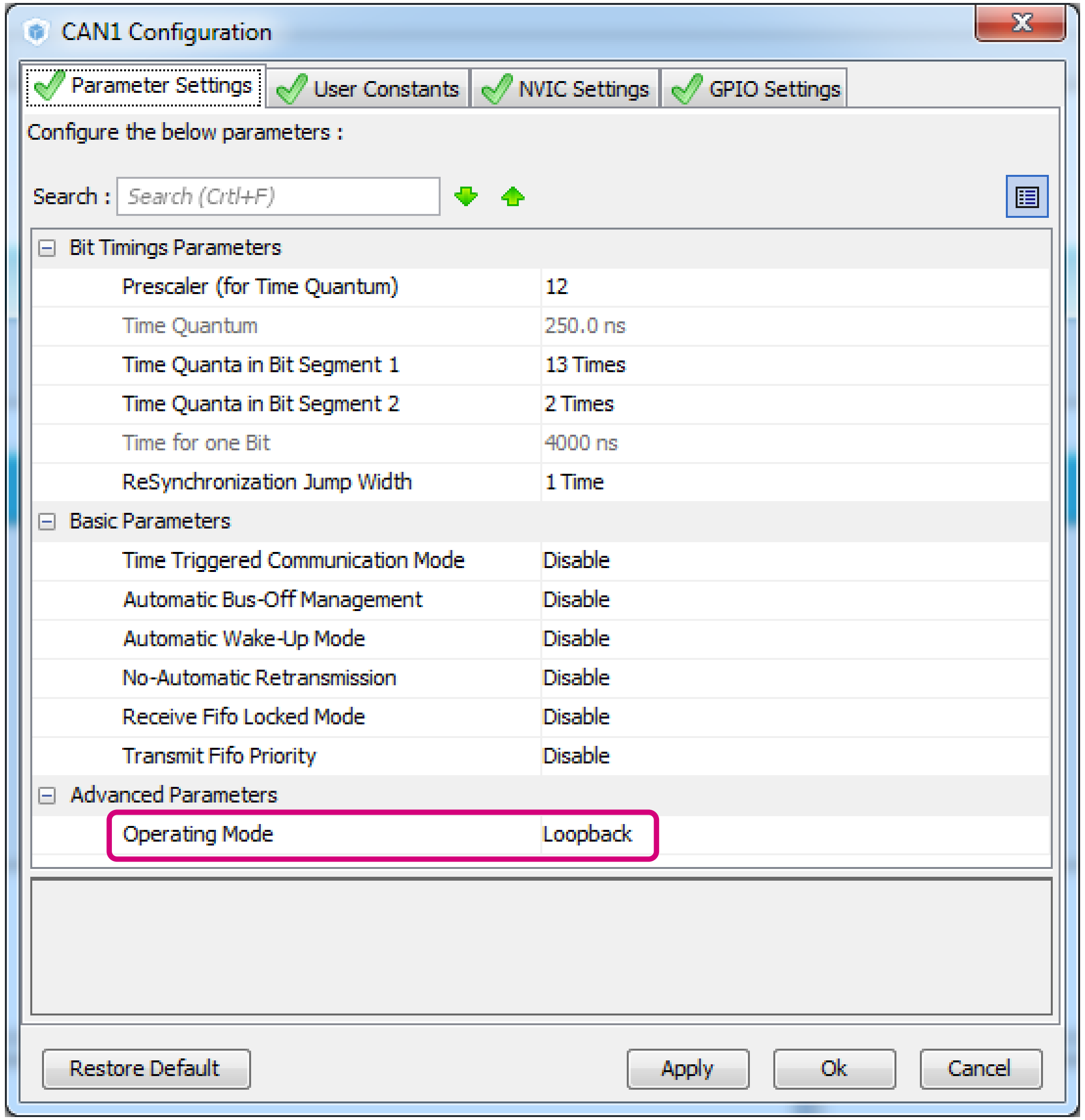


STM32CubeMX STM32L4_CAN.ioc: STM32L476VGTx

STM32CubeMX

configuration of CAN mode

- Select Parameter Settings tab
  - Change Operating Mode as Loopback or Silent
- Press **Ok** to confirm the configuration



CAN1 Configuration

Configure the below parameters :

Search :

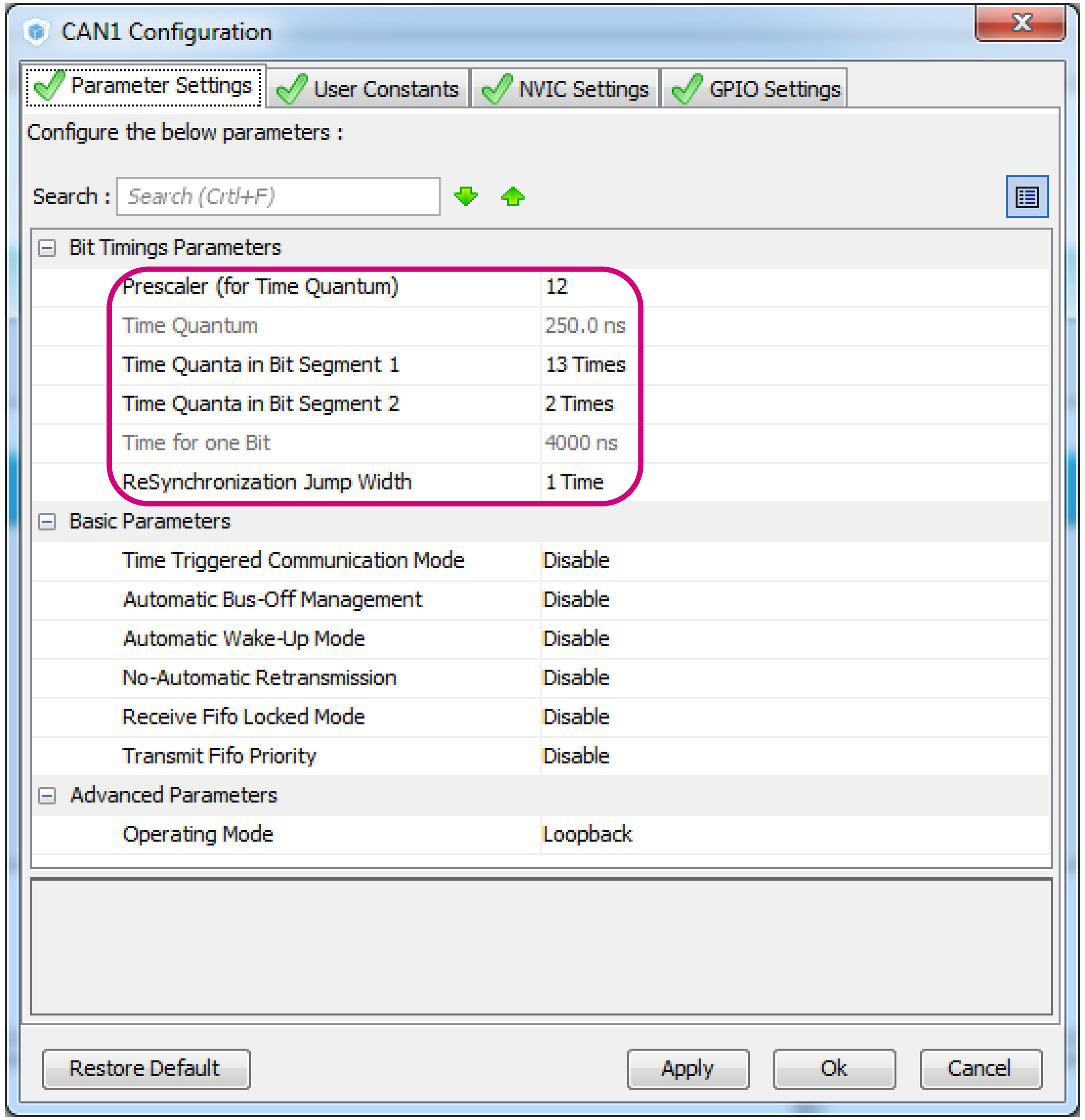
Bit Timings Parameters	
Prescaler (for Time Quantum)	12
Time Quantum	250.0 ns
Time Quanta in Bit Segment 1	13 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	4000 ns
ReSynchronization Jump Width	1 Time
Basic Parameters	
Time Triggered Communication Mode	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
No-Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Loopback

life.augmented

STM32CubeMX

configuration of CAN baudrate

- Select Parameter Settings tab
  - Fill in Bit Timing Parameters to set CAN baudrate
- Press **Ok** to confirm the configuration

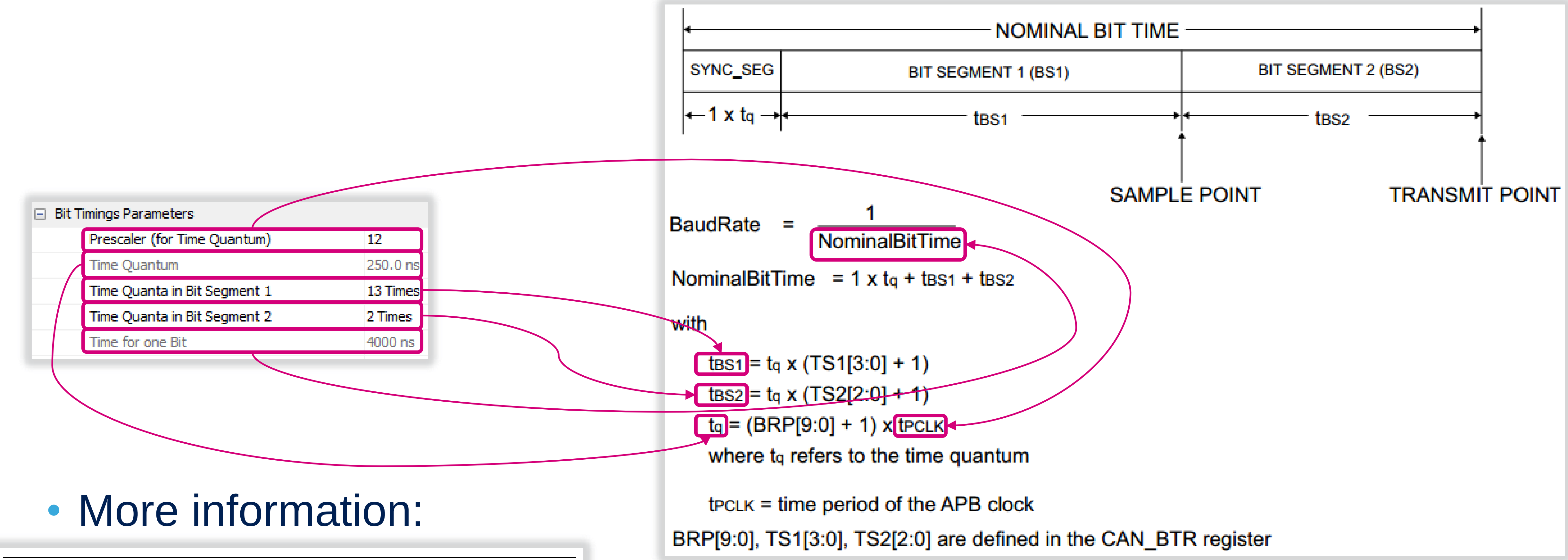


Configure the below parameters :

Bit Timings Parameters	
Prescaler (for Time Quantum)	12
Time Quantum	250.0 ns
Time Quanta in Bit Segment 1	13 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	4000 ns
ReSynchronization Jump Width	1 Time
Basic Parameters	
Time Triggered Communication Mode	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
No-Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Loopback

How to understand parameters, which have impact on CAN baudrate?

- Formula used to calculate CAN baudrate



⊟ Bit Timings Parameters

Prescaler (for Time Quantum)	12
Time Quantum	250.0 ns
Time Quanta in Bit Segment 1	13 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	4000 ns

$$\text{BaudRate} = \frac{1}{\boxed{\text{NominalBitTime}}}$$

$$\text{NominalBitTime} = 1 \times t_q + t_{BS1} + t_{BS2}$$

with

$$\boxed{t_{BS1}} = t_q \times (\text{TS1[3:0]} + 1)$$

$$\boxed{t_{BS2}} = t_q \times (\text{TS2[2:0]} + 1)$$

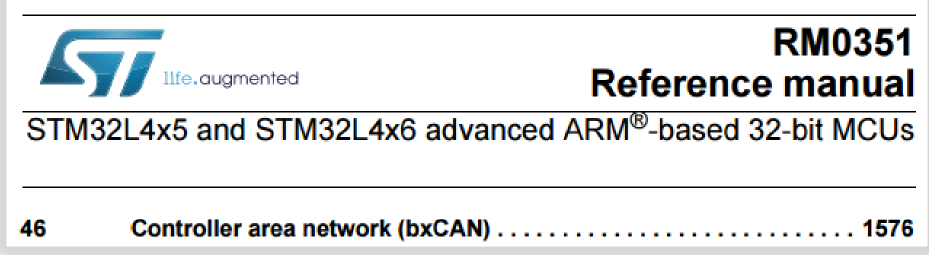
$$\boxed{t_q} = (\text{BRP[9:0]} + 1) \times \boxed{t_{PCLK}}$$

where t_q refers to the time quantum

t_{PCLK} = time period of the APB clock

BRP[9:0], TS1[3:0], TS2[2:0] are defined in the CAN_BTR register

- More information:



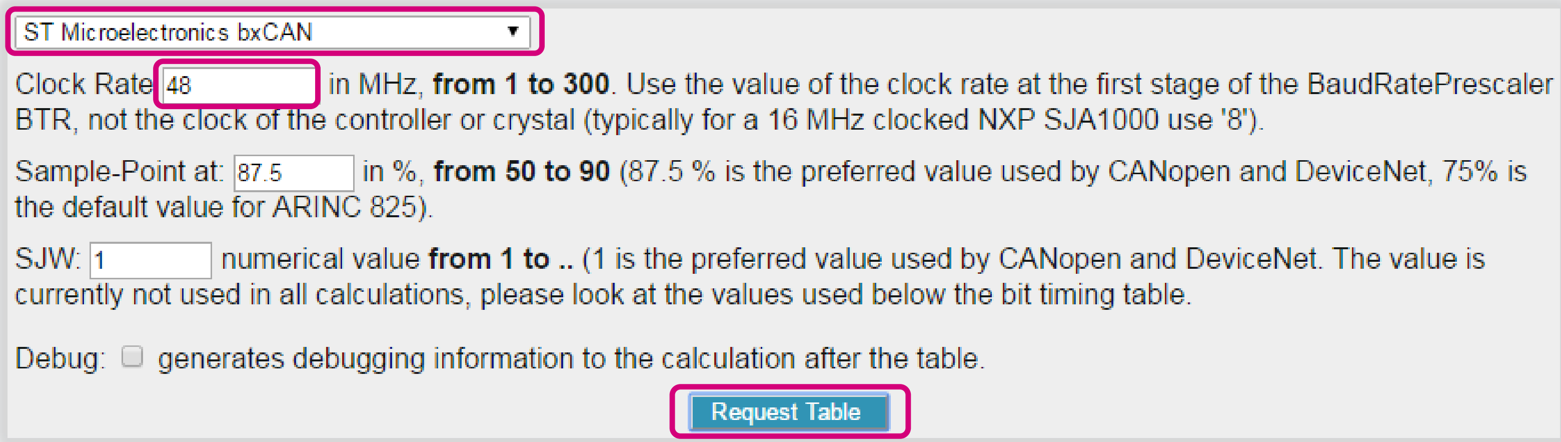
RM0351
Reference manual

STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs

Easy configuration of CAN baudrate

- http://www.bittiming.can-wiki.info/ webpage allows to obtain CAN baudrate configuation parameters automatically

1. Select STMicroelectronics as a CAN controller vendor
2. Select MCU's system clock
3. Click on **Request Table** button
4. Find desired CAN baudrate in the table and copy **clock prescaler**, **SB1** and **SB2** into the CubeMX



ST Microelectronics bxCAN ▾

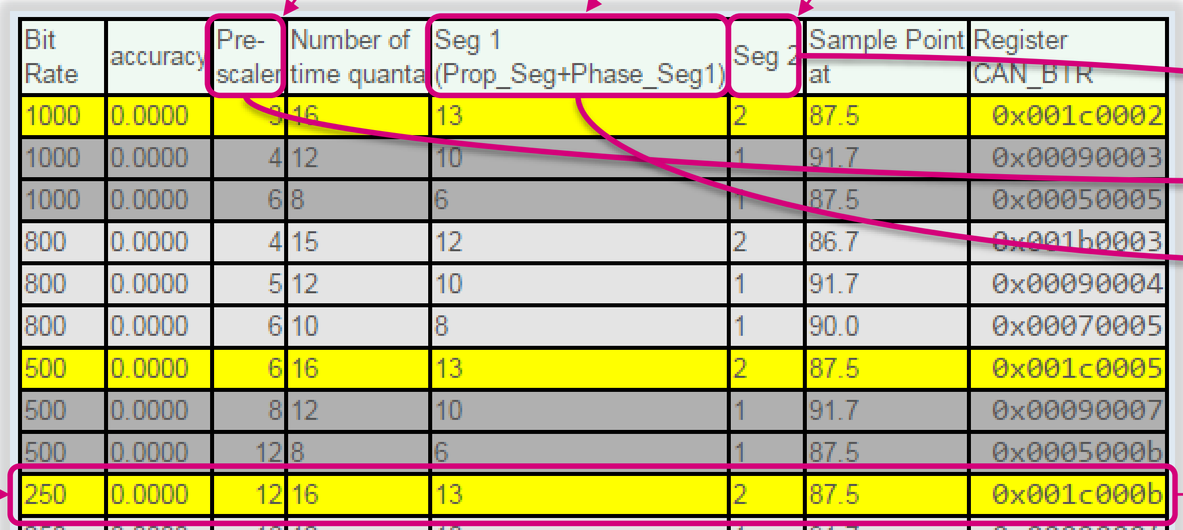
Clock Rate: 48 in MHz, **from 1 to 300**. Use the value of the clock rate at the first stage of the BaudRatePrescaler BTR, not the clock of the controller or crystal (typically for a 16 MHz clocked NXP SJA1000 use '8').

Sample-Point at: 87.5 in %, **from 50 to 90** (87.5 % is the preferred value used by CANopen and DeviceNet, 75% is the default value for ARINC 825).

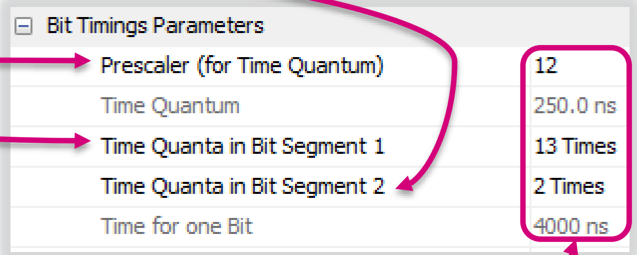
SJW: 1 numerical value **from 1 to ..** (1 is the preferred value used by CANopen and DeviceNet. The value is currently not used in all calculations, please look at the values used below the bit timing table.

Debug: □ generates debugging information to the calculation after the table.

Example for CAN baudrate 250 kbit/s



Bit Rate	accuracy	Pre-scaler	Number of time quanta	Seg 1 (Prop_Seg+Phase_Seg1)	Seg 2	Sample Point at	Register CAN_BTR
1000	0.0000	3	16	13	2	87.5	0x001c0002
1000	0.0000	4	12	10	1	91.7	0x00090003
1000	0.0000	6	8	6	1	87.5	0x00050005
800	0.0000	4	15	12	2	86.7	0x001b0003
800	0.0000	5	12	10	1	91.7	0x00090004
800	0.0000	6	10	8	1	90.0	0x00070005
500	0.0000	6	16	13	2	87.5	0x001c0005
500	0.0000	8	12	10	1	91.7	0x00090007
500	0.0000	12	8	6	1	87.5	0x0005000b
250	0.0000	12	16	13	2	87.5	0x001c000b



⊟ Bit Timings Parameters

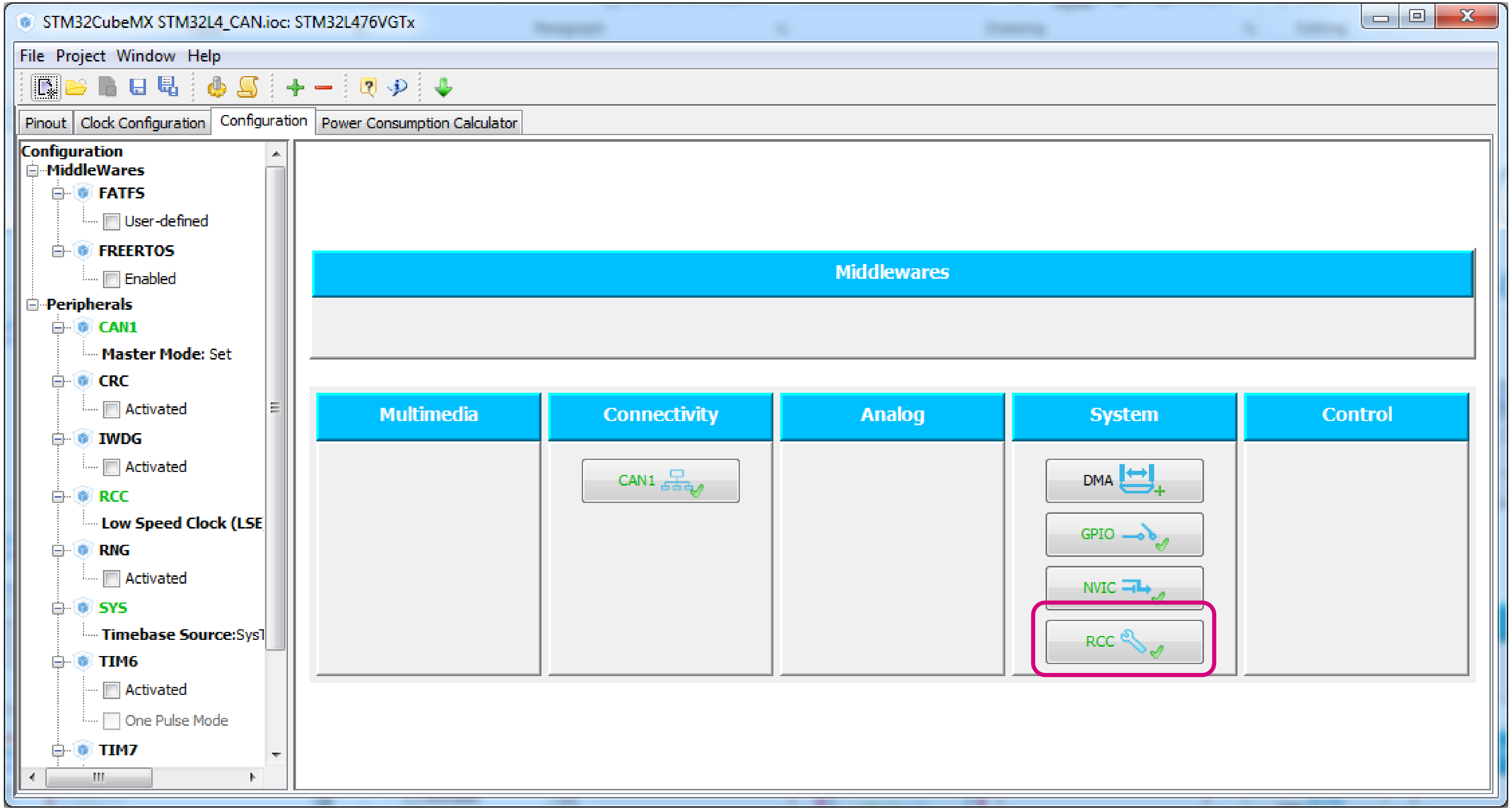
Prescaler (for Time Quantum)	12
Time Quantum	250.0 ns
Time Quanta in Bit Segment 1	13 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	4000 ns

life.augmented

STM32CubeMX

Configure clock

- Go to Configuration tab and select RCC peripheral



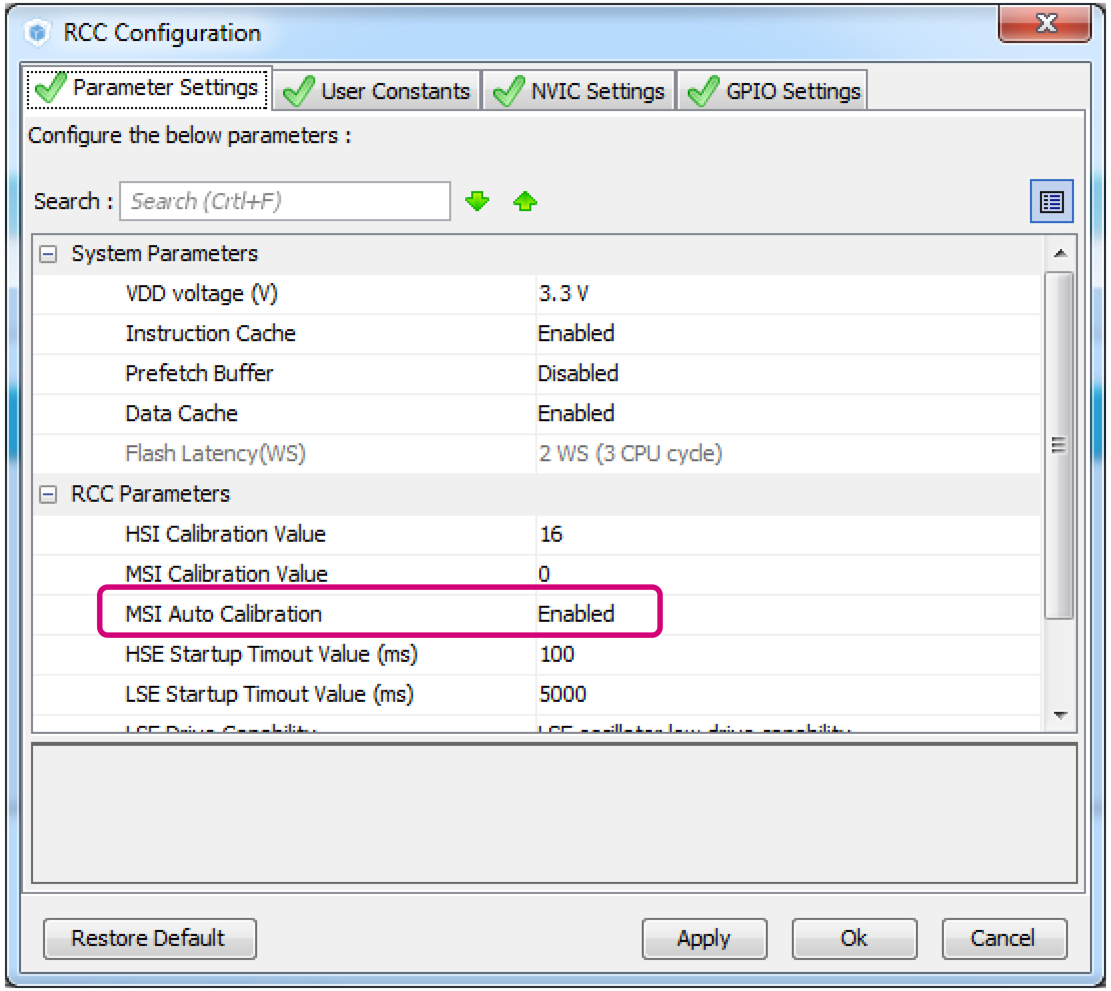
STM32CubeMX STM32L4_CAN.ioc: STM32L476VGTx

Configuration
- MiddleWares
  - FATFS
    - □ User-defined
  - FREERTOS
    - □ Enabled
- Peripherals
  - CAN1
    - Master Mode: Set
  - CRC
    - □ Activated
  - IWDG
    - □ Activated
  - RCC
    - Low Speed Clock (LSE
  - RNG
    - □ Activated
  - SYS
    - Timebase Source:SysT
  - TIM6
    - □ Activated
    - □ One Pulse Mode
  - TIM7

STM32CubeMX

configuration of the MSI calibration with LSE

- Select Parameter Settings tab
  - Enable MSI Auto Calibration
- Press **Ok** to confirm the configuration



Configure the below parameters :

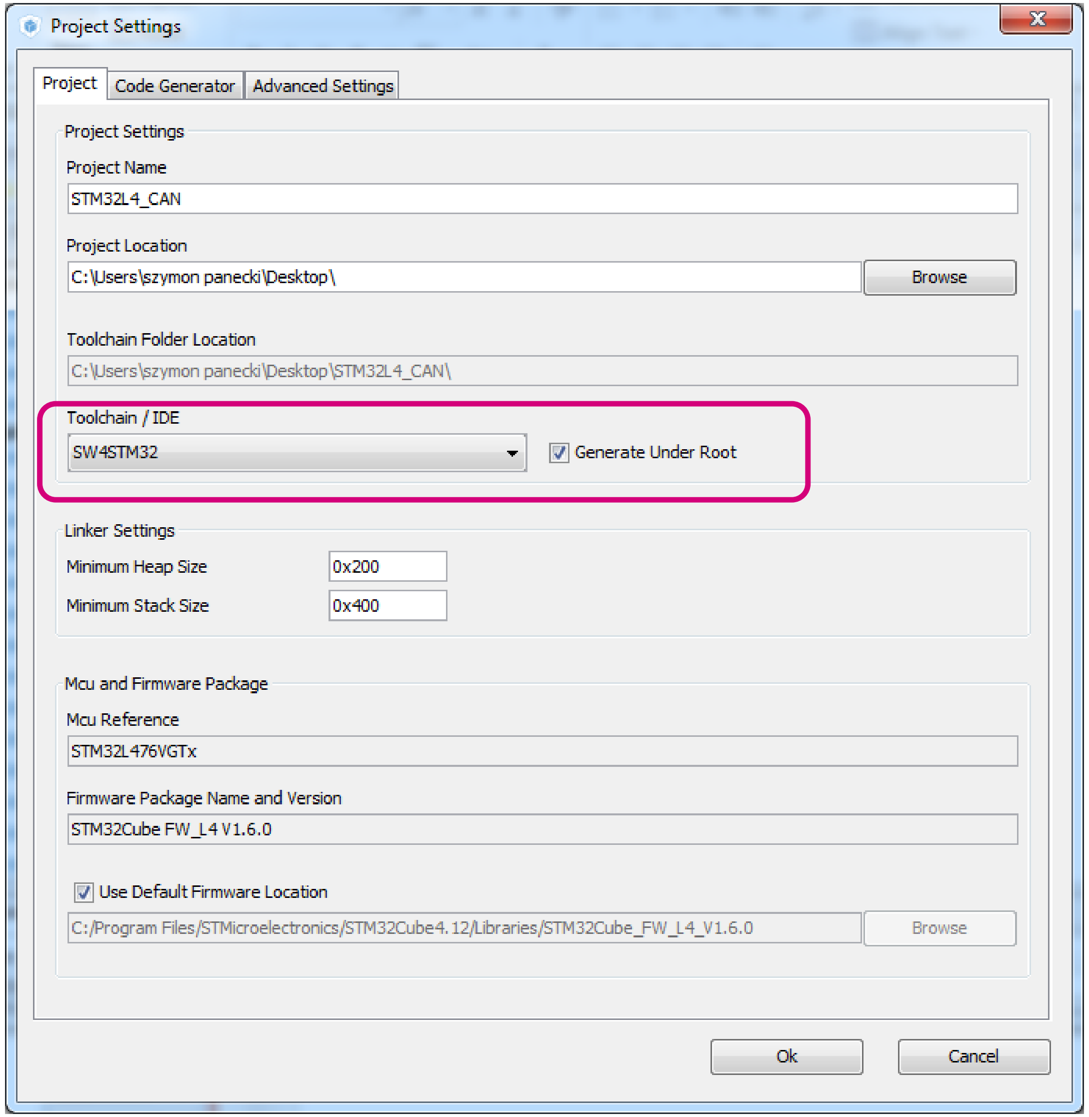
System Parameters	
VDD voltage (V)	3.3 V
Instruction Cache	Enabled
Prefetch Buffer	Disabled
Data Cache	Enabled
Flash Latency(WS)	2 WS (3 CPU cycle)
RCC Parameters	
HSI Calibration Value	16
MSI Calibration Value	0
MSI Auto Calibration	Enabled
HSE Startup Timout Value (ms)	100
LSE Startup Timout Value (ms)	5000
LSE Drive Capability	LSE oscillator low drive capability

STM32CubeMX

Project generation

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain

- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Settings

Project Name: STM32L4_CAN

Project Location: C:\Users\szymon panecki\Desktop\

Toolchain Folder Location: C:\Users\szymon panecki\Desktop\STM32L4_CAN\

Toolchain / IDE: SW4STM32

☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

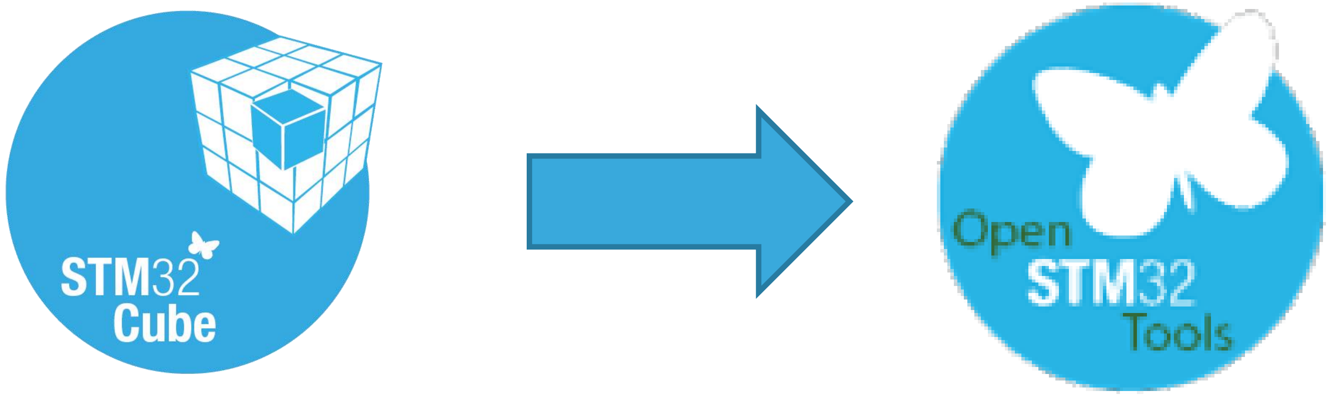
Mcu Reference: STM32L476VGTx

Firmware Package Name and Version: STM32Cube FW_L4 V1.6.0

☑ Use Default Firmware Location

C:/Program Files/STMicroelectronics/STM32Cube4.12/Libraries/STM32Cube_FW_L4_V1.6.0

life.augmented



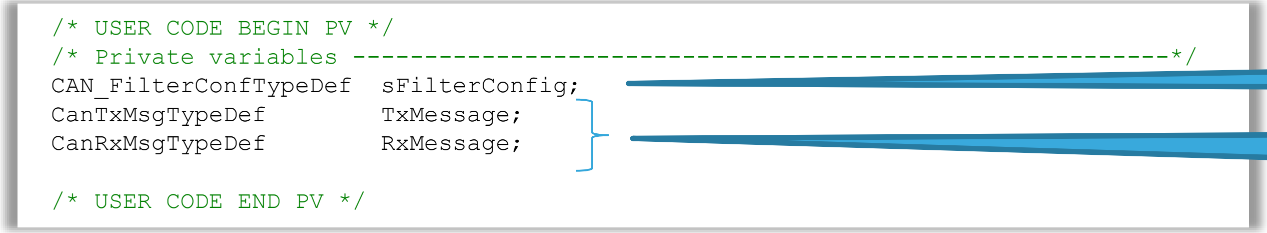
- After successful code generation by STM32CubeMX this is the right time to import it into SW4STM32 toolchain for further processing

Modifying the code

data declaration - main.c file

Tasks:

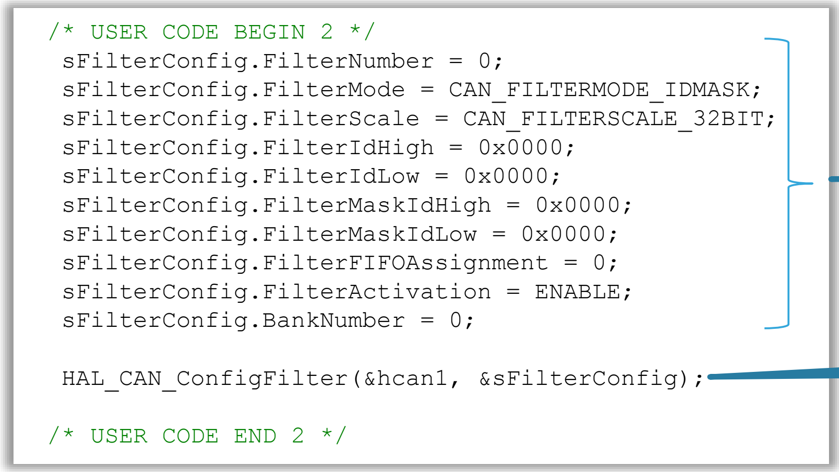
1. Create structures for managing CAN (filters, tranmission message, reception message)
2. Configure filters in the way, that all received messages are accepted



```
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
CAN_FilterConfTypeDef  sFilterConfig;
CanTxMsgTypeDef        TxMessage;
CanRxMsgTypeDef        RxMessage;

/* USER CODE END PV */
```

CAN filter structure

CAN transmission and reception structuresstructure



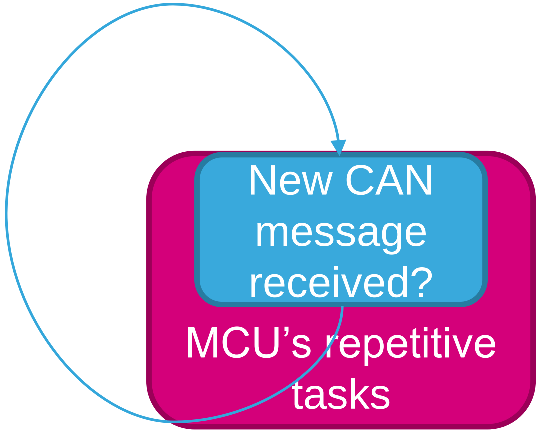
```
/* USER CODE BEGIN 2 */
 sFilterConfig.FilterNumber = 0;
 sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
 sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
 sFilterConfig.FilterIdHigh = 0x0000;
 sFilterConfig.FilterIdLow = 0x0000;
 sFilterConfig.FilterMaskIdHigh = 0x0000;
 sFilterConfig.FilterMaskIdLow = 0x0000;
 sFilterConfig.FilterFIFOAssignment = 0;
 sFilterConfig.FilterActivation = ENABLE;
 sFilterConfig.BankNumber = 0;

 HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig);

/* USER CODE END 2 */
```

CAN filter strucutre items configuration

CAN filter configuration function call

- Implementation of CAN message reception by pooling

Pooling approach

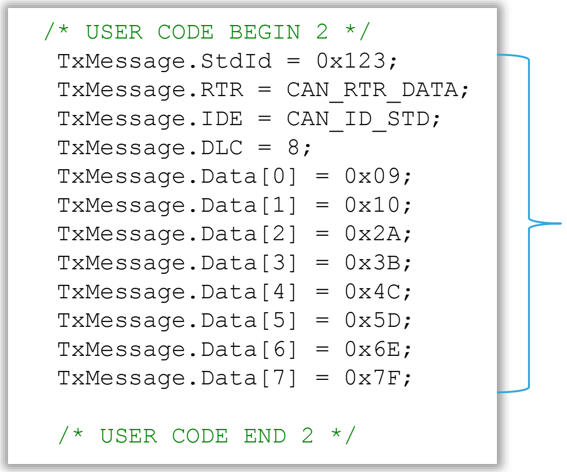


Modifying the code

message transmission and reception - main.c file

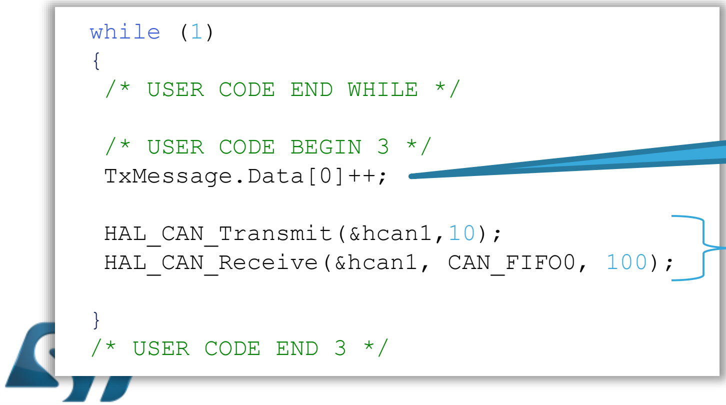
Tasks:

1. Fill in structure for CAN message transmission
2. In infinite loop call two functions: to send and receive CAN message



```
/* USER CODE BEGIN 2 */
 TxMessage.StdId = 0x123;
 TxMessage.RTR = CAN_RTR_DATA;
 TxMessage.IDE = CAN_ID_STD;
 TxMessage.DLC = 8;
 TxMessage.Data[0] = 0x09;
 TxMessage.Data[1] = 0x10;
 TxMessage.Data[2] = 0x2A;
 TxMessage.Data[3] = 0x3B;
 TxMessage.Data[4] = 0x4C;
 TxMessage.Data[5] = 0x5D;
 TxMessage.Data[6] = 0x6E;
 TxMessage.Data[7] = 0x7F;

 /* USER CODE END 2 */
```

CAN message strucutre items configuration



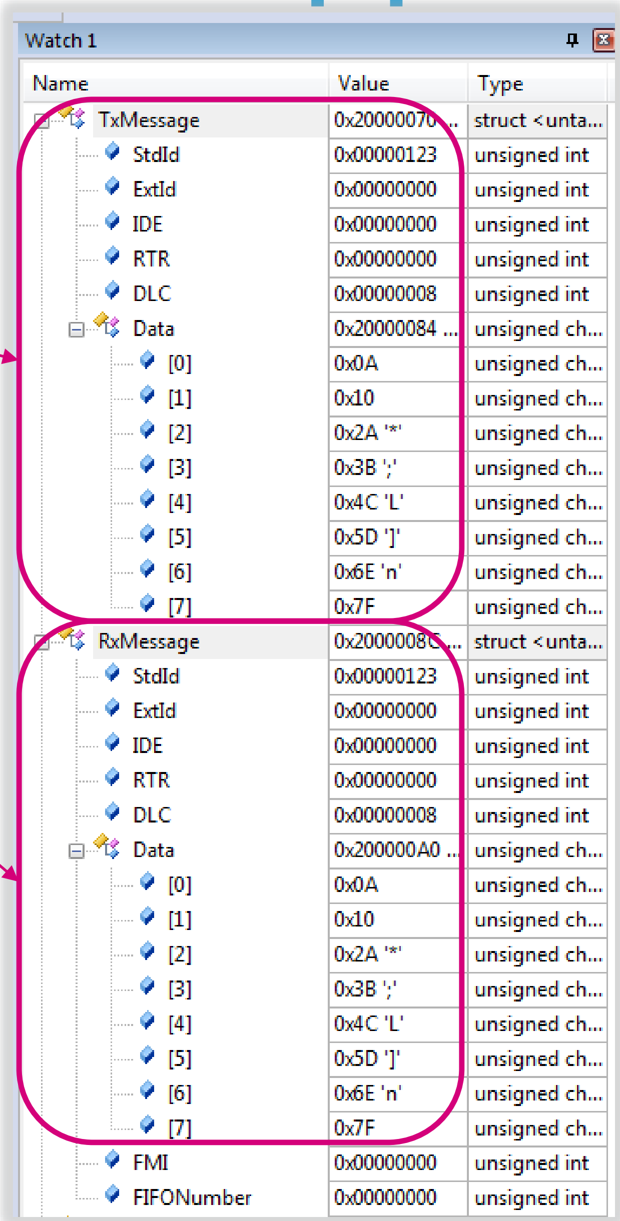
```
while (1)
{
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
 TxMessage.Data[0]++;

 HAL_CAN_Transmit(&hcan1,10);
 HAL_CAN_Receive(&hcan1, CAN_FIFO0, 100);

}
/* USER CODE END 3 */
```

Incrementation of CAN message strucutre's data item with each loop iteration

Call of functions: to send and receive CAN message

Running the application

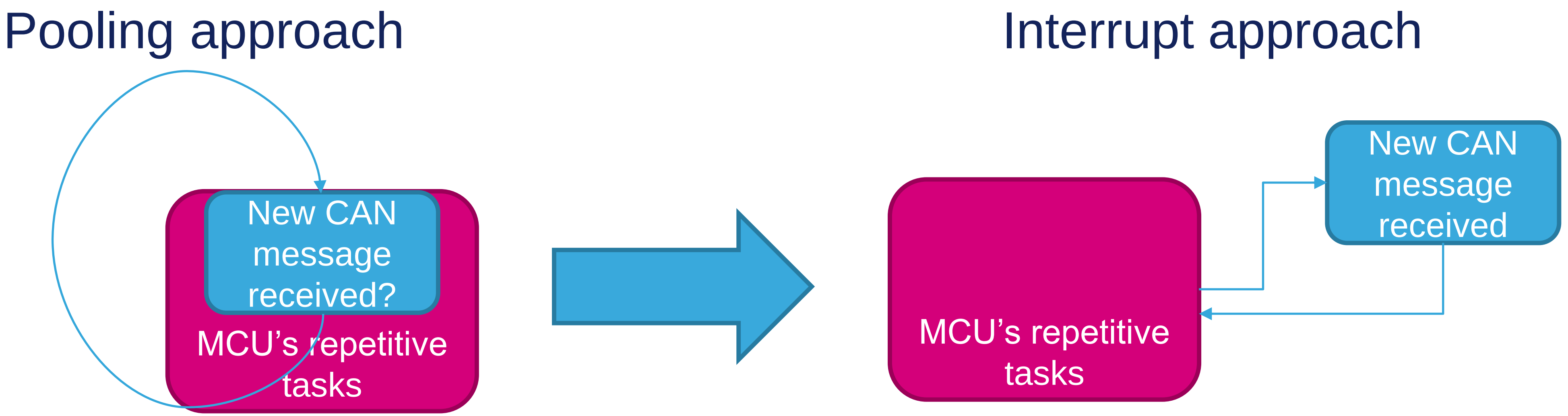
- In debug session observe content of the
  - **TxMessage** transmission buffer
  - **RxMessage** reception buffer



Watch 1

Name	Value	Type
TxMessage	0x20000070 ...	struct <unta...
StdId	0x00000123	unsigned int
ExtId	0x00000000	unsigned int
IDE	0x00000000	unsigned int
RTR	0x00000000	unsigned int
DLC	0x00000008	unsigned int
Data	0x20000084 ...	unsigned ch...
[0]	0x0A	unsigned ch...
[1]	0x10	unsigned ch...
[2]	0x2A '*'	unsigned ch...
[3]	0x3B ';'	unsigned ch...
[4]	0x4C 'L'	unsigned ch...
[5]	0x5D ']'	unsigned ch...
[6]	0x6E 'n'	unsigned ch...
[7]	0x7F	unsigned ch...
RxMessage	0x2000008C ...	struct <unta...
StdId	0x00000123	unsigned int
ExtId	0x00000000	unsigned int
IDE	0x00000000	unsigned int
RTR	0x00000000	unsigned int
DLC	0x00000008	unsigned int
Data	0x200000A0 ...	unsigned ch...
[0]	0x0A	unsigned ch...
[1]	0x10	unsigned ch...
[2]	0x2A '*'	unsigned ch...
[3]	0x3B ';'	unsigned ch...
[4]	0x4C 'L'	unsigned ch...
[5]	0x5D ']'	unsigned ch...
[6]	0x6E 'n'	unsigned ch...
[7]	0x7F	unsigned ch...
FMI	0x00000000	unsigned int
FIFONumber	0x00000000	unsigned int

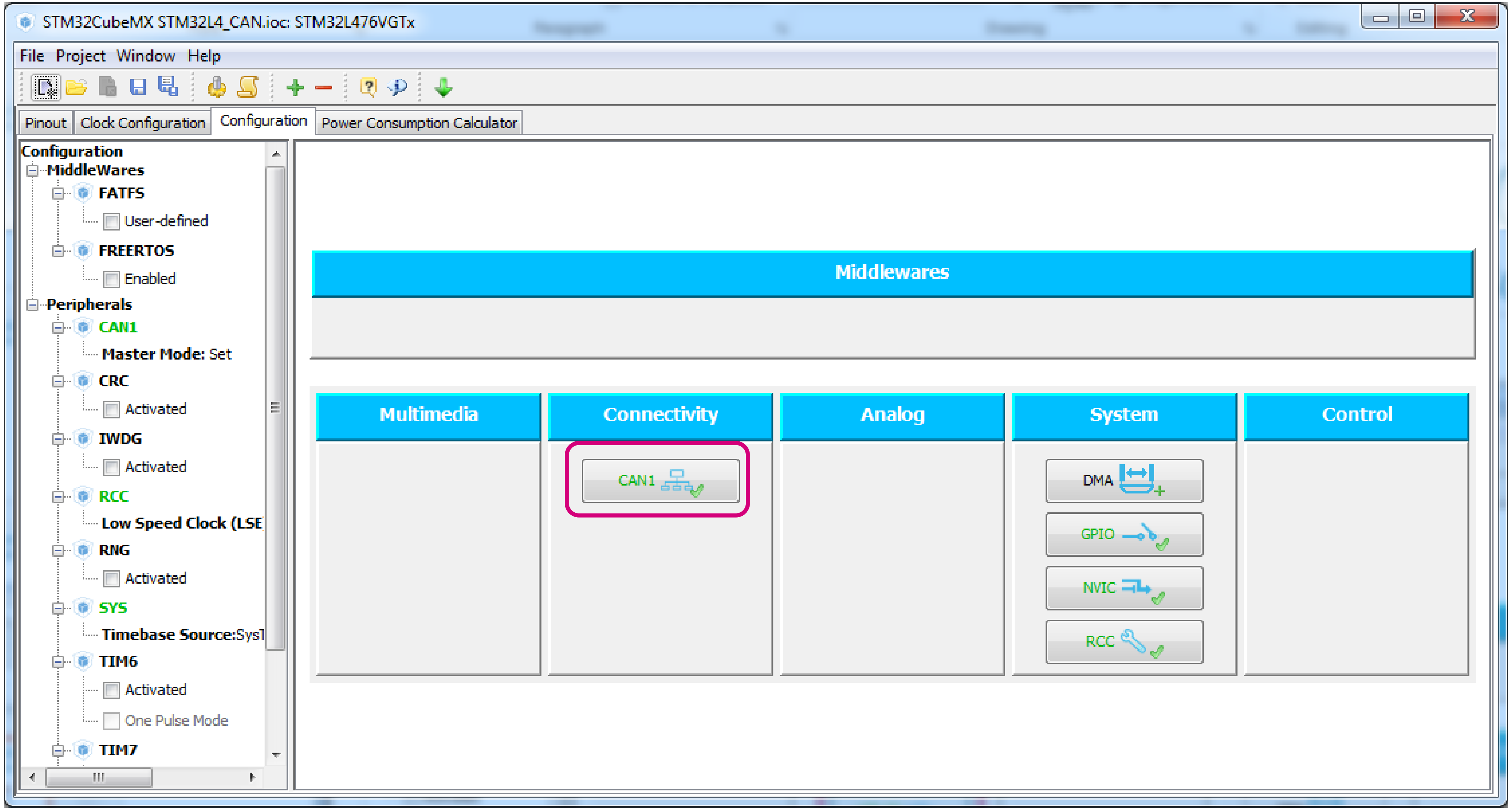
- Modification of application in order to replace CAN message reception by pooling with CAN message reception with interrupt



STM32CubeMX

Configure CAN

- Go to Configuration tab and select CAN peripheral



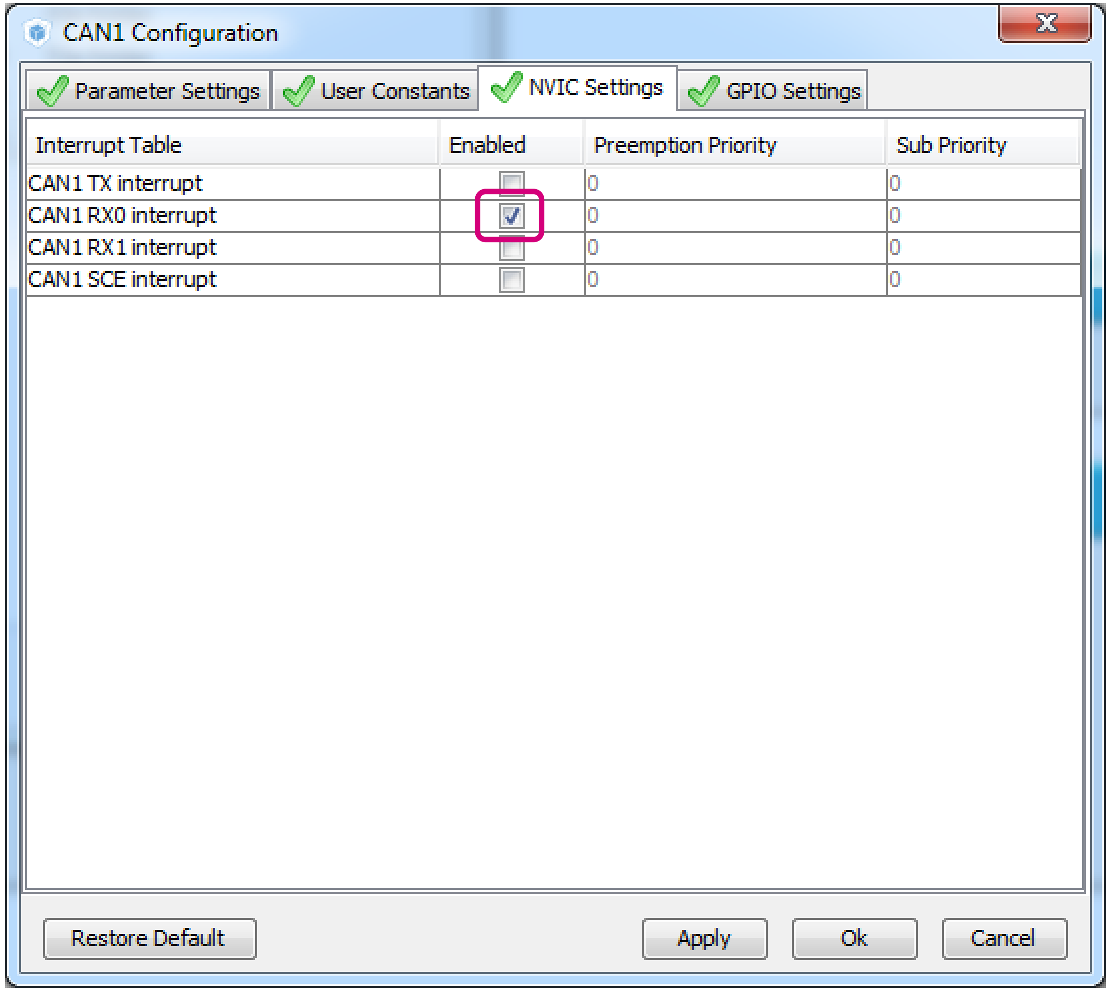
STM32CubeMX STM32L4_CAN.ioc: STM32L476VGTx

Configuration
- MiddleWares
  - FATFS
    - □ User-defined
  - FREERTOS
    - □ Enabled
- Peripherals
  - CAN1
    - Master Mode: Set
  - CRC
    - □ Activated
  - IWDG
    - □ Activated
  - RCC
    - Low Speed Clock (LSE
  - RNG
    - □ Activated
  - SYS
    - Timebase Source:SysT
  - TIM6
    - □ Activated
    - □ One Pulse Mode
  - TIM7

STM32CubeMX

enabling of CAN receive interrupt

- Select NVIC Settings tab
  - Enable CAN1RX0 interrupt
- Press **Ok** to confirm the configuration



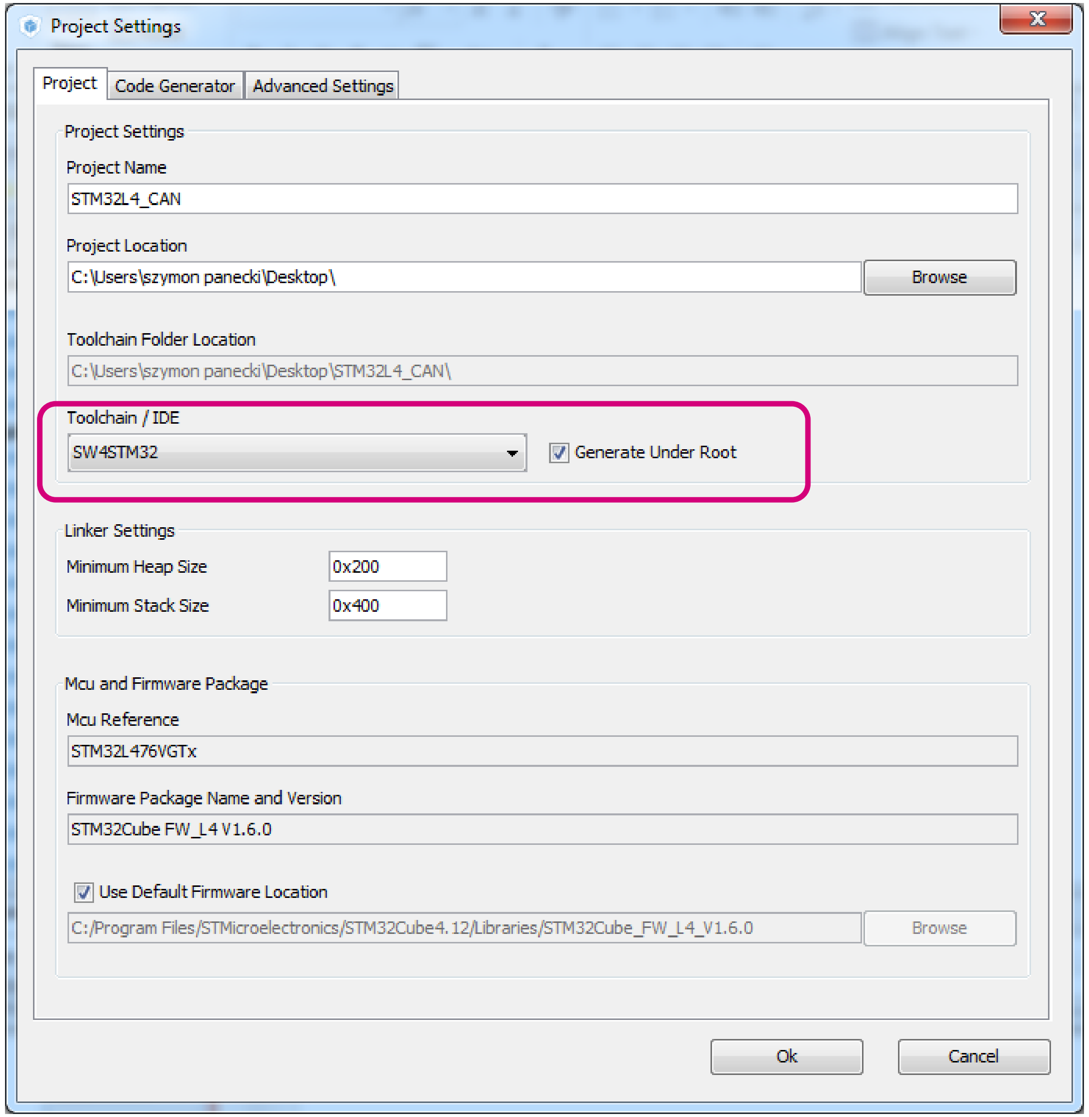
Interrupt Table	Enabled	Preemption Priority	Sub Priority
CAN1 TX interrupt	□	0	0
CAN1 RX0 interrupt	☑	0	0
CAN1 RX1 interrupt	□	0	0
CAN1 SCE interrupt	□	0	0

STM32CubeMX

Project generation

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain

- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Name: STM32L4_CAN

Project Location: C:\Users\szymon panecki\Desktop\

Toolchain Folder Location: C:\Users\szymon panecki\Desktop\STM32L4_CAN\

Toolchain / IDE: SW4STM32 ☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Mcu and Firmware Package

Mcu Reference: STM32L476VGTx

Firmware Package Name and Version: STM32Cube FW_L4 V1.6.0

☑ Use Default Firmware Location

C:/Program Files/STMicroelectronics/STM32Cube4.12/Libraries/STM32Cube_FW_L4_V1.6.0

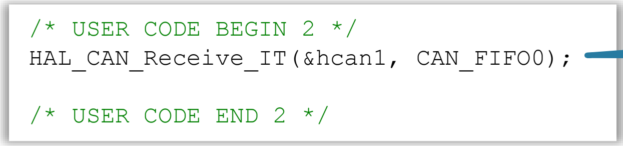
life.augmented

Modifying the code

message transmission and reception - main.c file

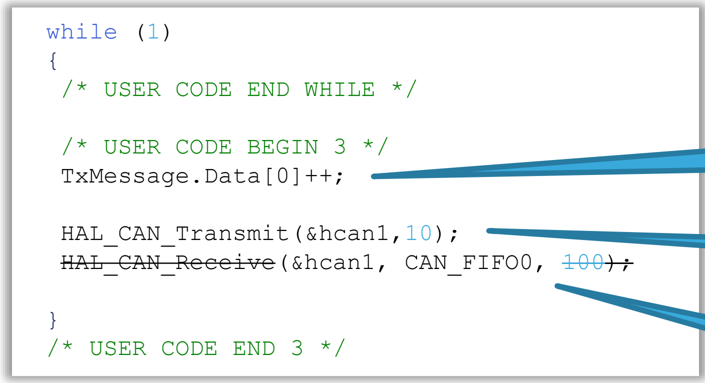
Tasks:

1. Call a function to enable CAN reception interrupt
2. Remove call of function, which receives CAN messages in infinie while loop



```
/* USER CODE BEGIN 2 */
HAL_CAN_Receive_IT(&hcan1, CAN_FIFO0);

/* USER CODE END 2 */
```

Function call before while(1) loop to enable CAN reception interrupt



```
while (1)
{
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
 TxMessage.Data[0]++;

 HAL_CAN_Transmit(&hcan1,10);
 ~~HAL_CAN_Receive~~(&hcan1, CAN_FIFO0, ~~100);~~

}
/* USER CODE END 3 */
```

Incrementation of CAN message strucutre’s data item with each loop iteration

Call of function to send CAN message

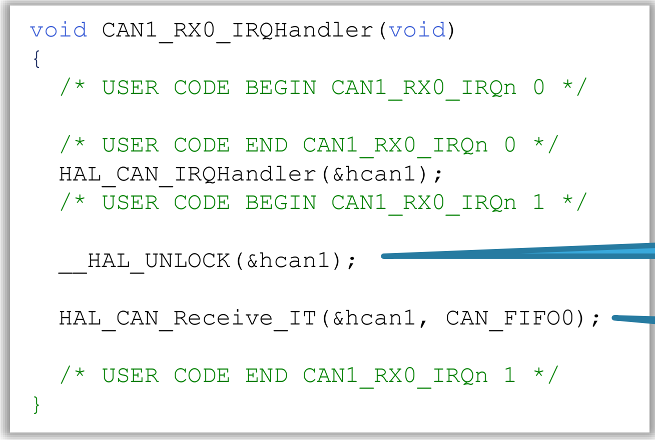
Comparing to pooling approach, call of this function should be removed, as CAN message reception will now be implemented in interrupt routine

Modifying the code

message reception - stm32l4xx_it.c file

Tasks:

1. In CAN reception interrupt handler call function to receive CAN message
2. In CAN reception interrupt handler call function to UNLOCK HAL after each interrupt generation



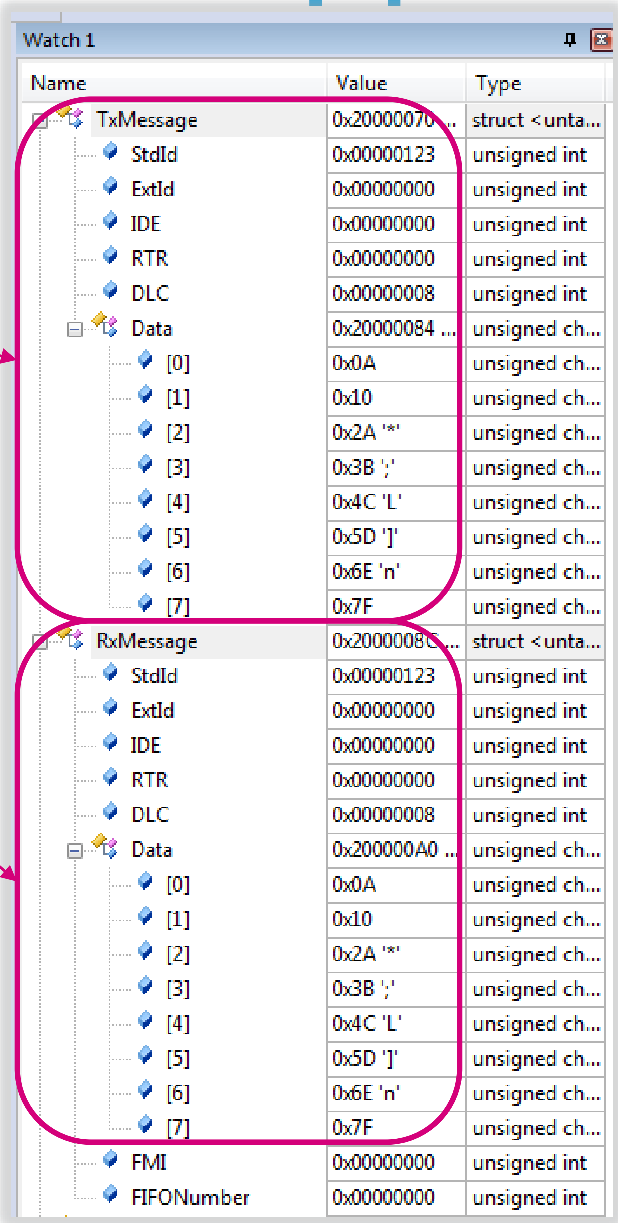
```
void CAN1_RX0_IRQHandler(void)
{
  /* USER CODE BEGIN CAN1_RX0_IRQn 0 */

  /* USER CODE END CAN1_RX0_IRQn 0 */
  HAL_CAN_IRQHandler(&hcan1);
  /* USER CODE BEGIN CAN1_RX0_IRQn 1 */

  __HAL_UNLOCK(&hcan1);

  HAL_CAN_Receive_IT(&hcan1, CAN_FIFO0);

  /* USER CODE END CAN1_RX0_IRQn 1 */
}
```

Call this function to release manually HAL for CAN structure

Call of function to receive CAN message in interrupt

Running the application

- In debug session observe content of the
  - **TxMessage** transmission buffer
  - **RxMessage** reception buffer



Watch 1

Name	Value	Type
TxMessage	0x20000070 ...	struct <unta...
StdId	0x00000123	unsigned int
ExtId	0x00000000	unsigned int
IDE	0x00000000	unsigned int
RTR	0x00000000	unsigned int
DLC	0x00000008	unsigned int
Data	0x20000084 ...	unsigned ch...
[0]	0x0A	unsigned ch...
[1]	0x10	unsigned ch...
[2]	0x2A '*'	unsigned ch...
[3]	0x3B ';'	unsigned ch...
[4]	0x4C 'L'	unsigned ch...
[5]	0x5D ']'	unsigned ch...
[6]	0x6E 'n'	unsigned ch...
[7]	0x7F	unsigned ch...
RxMessage	0x2000008C ...	struct <unta...
StdId	0x00000123	unsigned int
ExtId	0x00000000	unsigned int
IDE	0x00000000	unsigned int
RTR	0x00000000	unsigned int
DLC	0x00000008	unsigned int
Data	0x200000A0 ...	unsigned ch...
[0]	0x0A	unsigned ch...
[1]	0x10	unsigned ch...
[2]	0x2A '*'	unsigned ch...
[3]	0x3B ';'	unsigned ch...
[4]	0x4C 'L'	unsigned ch...
[5]	0x5D ']'	unsigned ch...
[6]	0x6E 'n'	unsigned ch...
[7]	0x7F	unsigned ch...
FMI	0x00000000	unsigned int
FIFONumber	0x00000000	unsigned int

Enjoy!

www.st.com/mcu