S32G-VNP-GLDBOX

REAL TIME DRIVER

EXAMPLE ENABLEMENT GUIDE



SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.

CONTENTS

- Hands on UART Real Time Driver example
- Hands on ETH Real Time Driver example
- Hands on CAN Real Time Driver example

HARDWARE REQUIREMENT AND SOFTWARE INSTALLATION

Hardware Requirement

- S32G-VNP-RDB2
- S32 Debug Probe
- AD/DC power supply
- Serial port cable for UART example

Software Installation

- Install S32DS 3.4 according to S32G-VNP-RDB2 Software Enablement Guide
- Install SW32_RTD_4.4_1.0.0 HF1(RTD) according to S32G-VNP-RDB2 Software Enablement Guide

Hands on UART Example



SECURE CONNECTIONS
FOR A SMARTER WORLD



PUBLIC

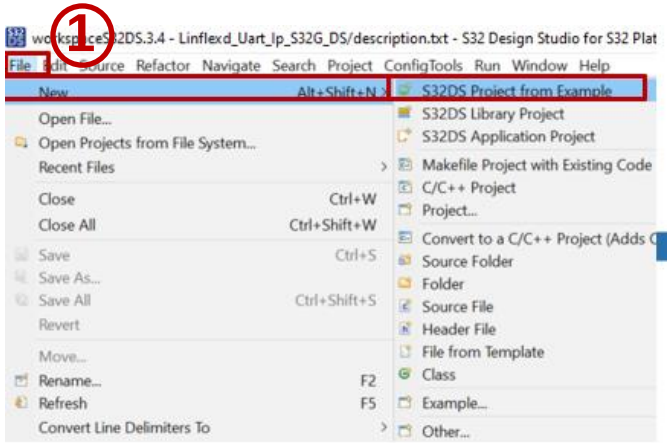
NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.

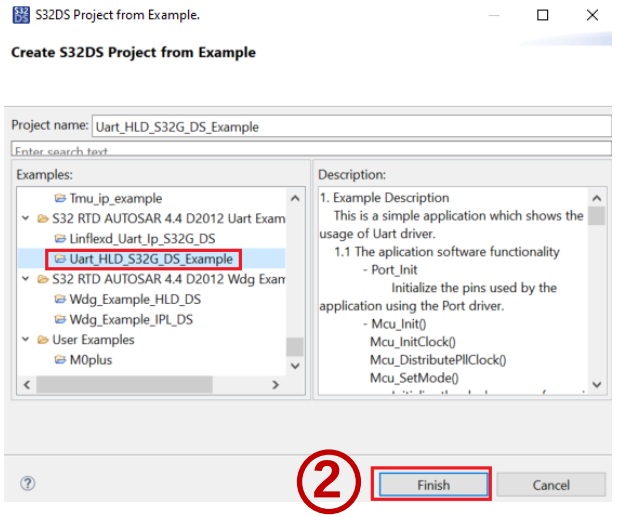
HANDS ON UART: OBJECTIVE

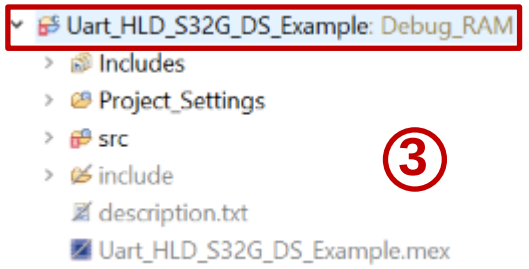
- How to import the UART example into S32DS
- How to configure the clock of UART via S32DS
- How to configure the UART setting via S32DS
- How to debug the UART example with S32 debug probe

HANDS ON UART: IMPORT UART EXAMPLE PROJECT

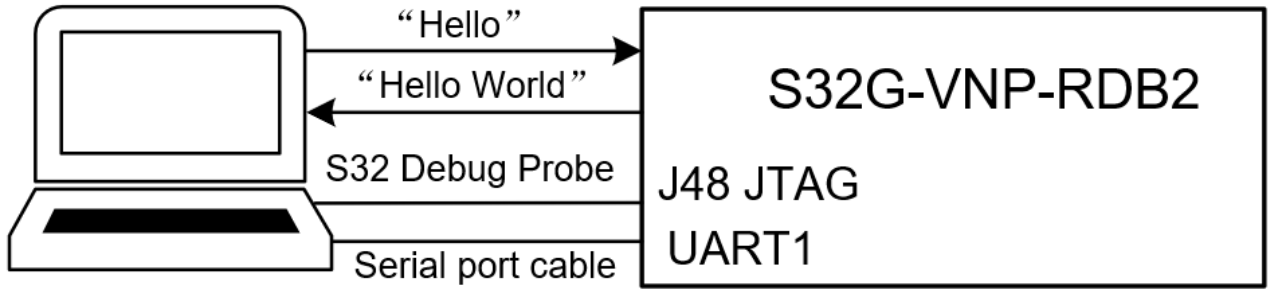
Open S32DS3.4, go to “File -> New -> S32DS Project From Example”. Select “**Uart_HLD_S32G_DS_Example**” example, Then click on “Finish”. The project should now be copied into current workspace.





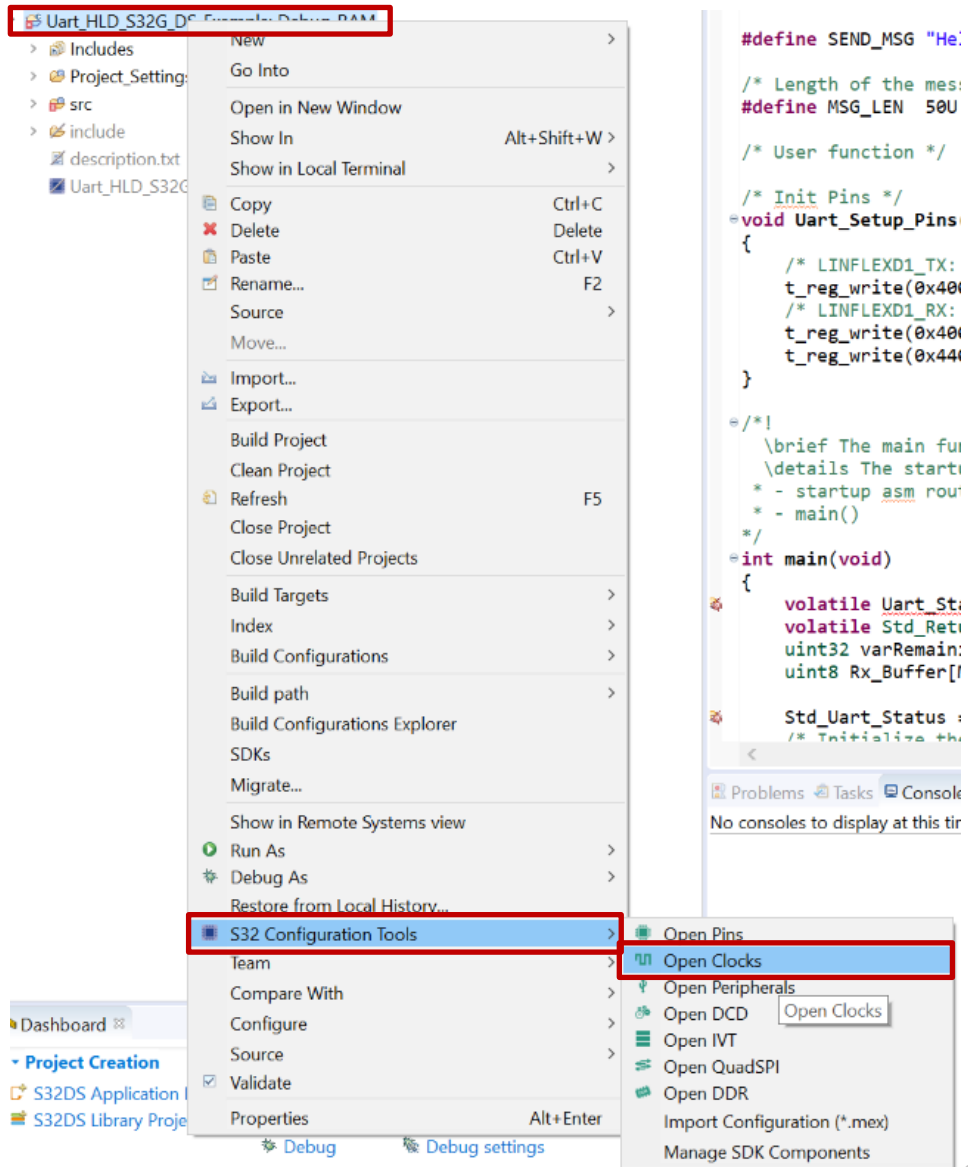


The purpose of “Uart_HLD_S32G_DS_Example” example is a simple application which shows the usage of UART driver.



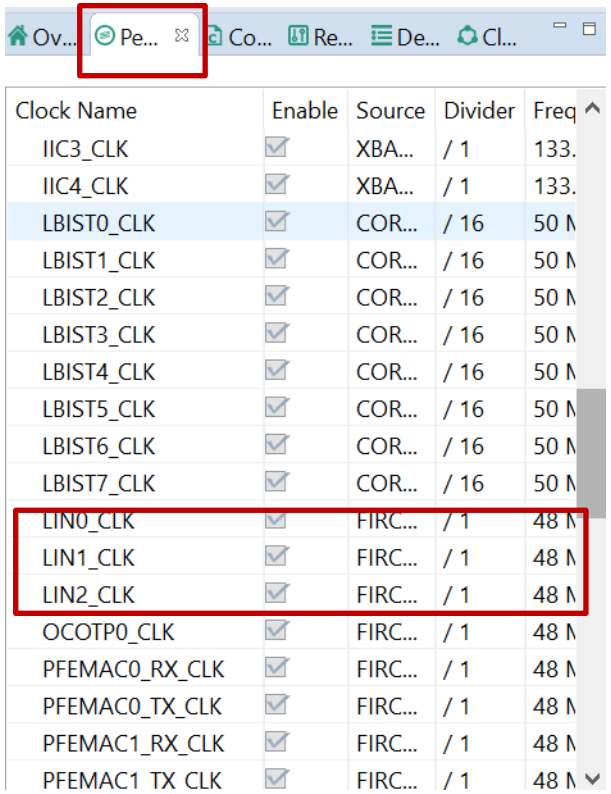
HANDS ON UART: CLOCK CONFIGURATION 1

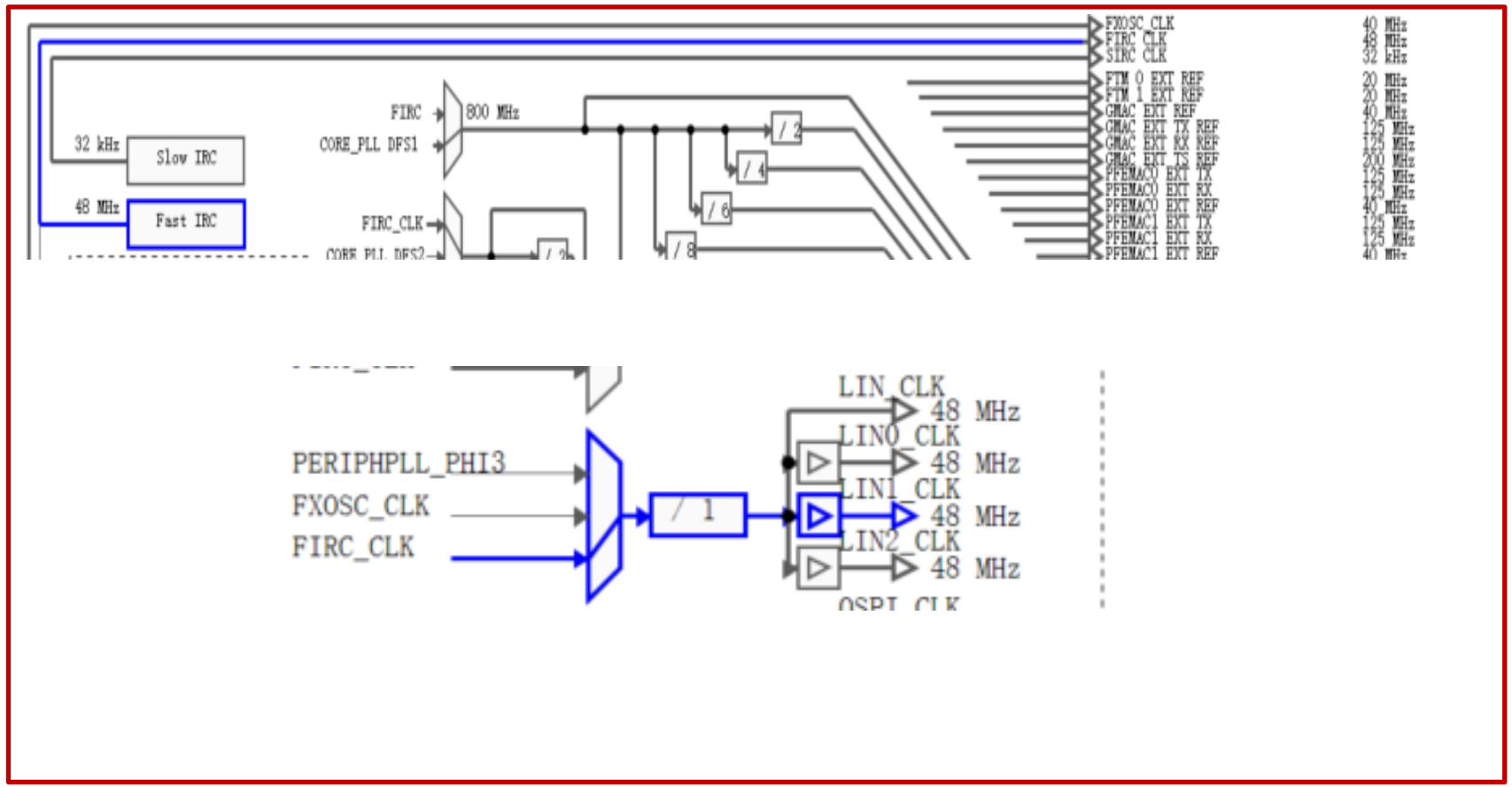
- Go to desired configuration tool:
  - Right click on Project,
  - Select S32 Configuration Tool…
  - Select Open Clocks



HANDS ON UART: CLOCK CONFIGURATION 2

Open the **Peripheral Clock View**, Double click the Lin module. The **Clocks Diagram** will show the power tree. In Uart_HLD_S32G_DS_Example project. The default clock configuration of UART is 48 MHZ which comes from FIRC directly

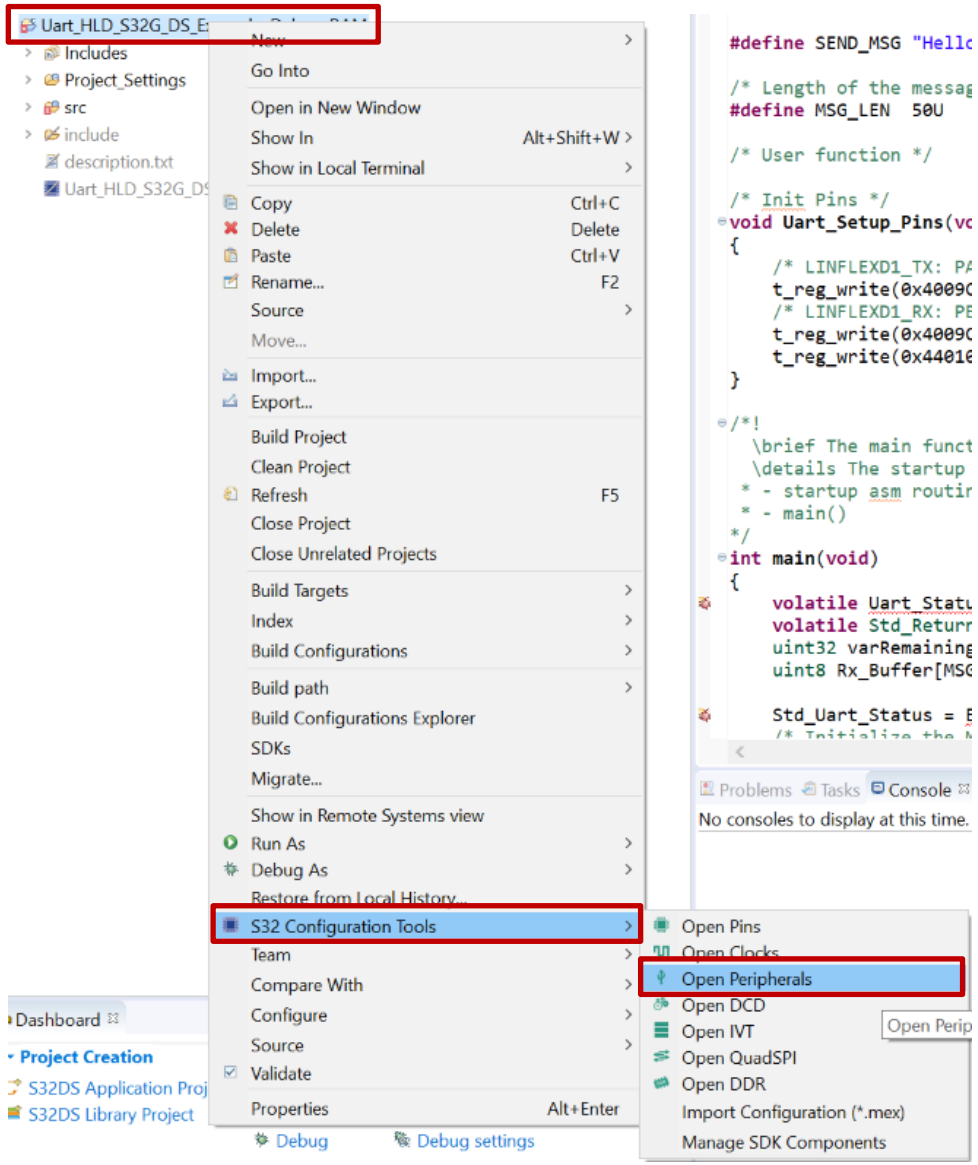




HANDS ON UART: UART CONFIGURATION 1

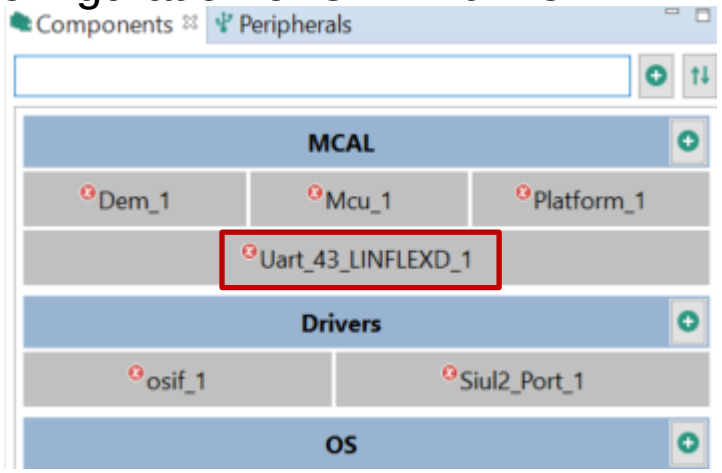
Open the Clocks Diagram:

- Right click on Project,
- Select S32 Configuration Tool…
- Select Peripherals



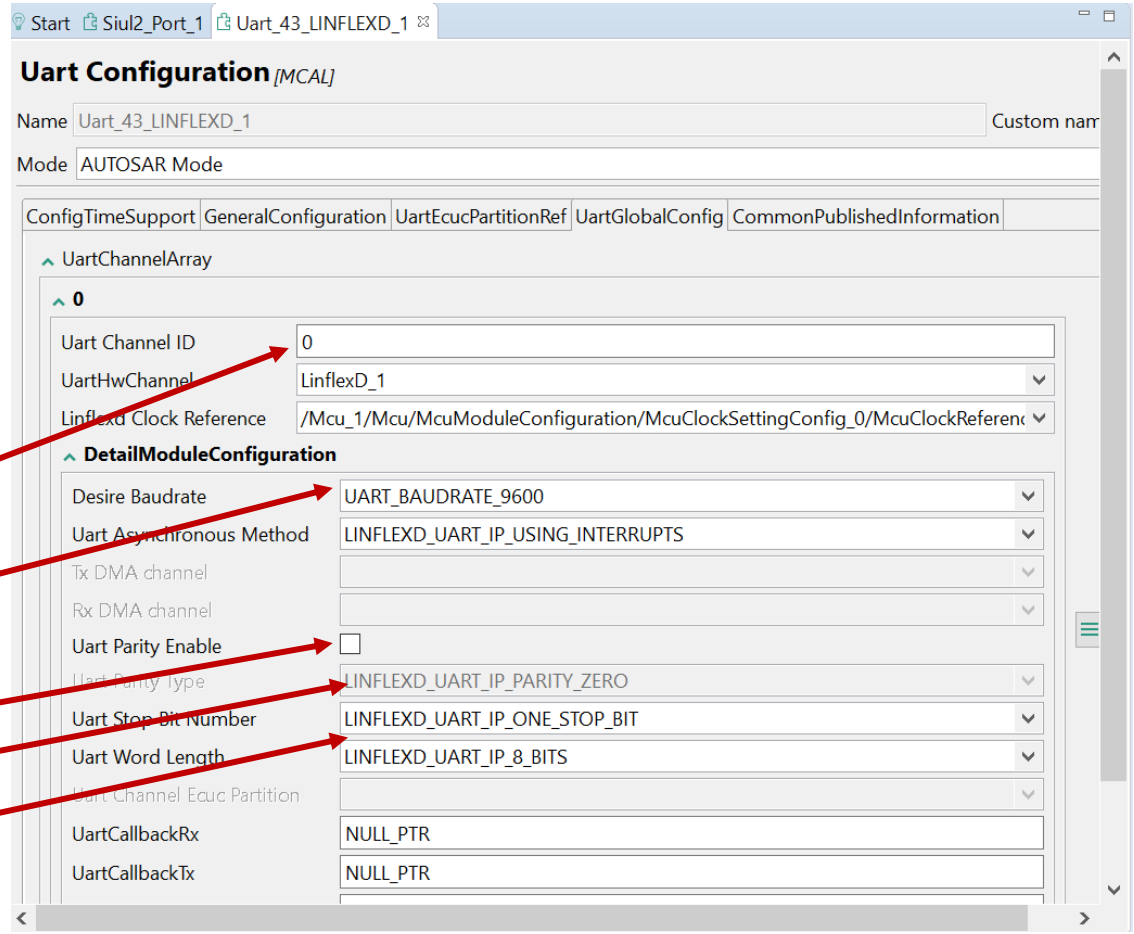
HANDS ON UART: UART CONFIGURATION 2

The **Components** shows all drivers which used by this example, the **UART_43_LINFLEXD_1** includes the configuration of UART driver



UART default configuration:

- - Select correct COM Port
- - Select Baudrate of 9600
- - Select none parity checking
- - Select 1 stop bits
- - Select 8 data bits



HANDS ON UART: UPDATE CODE

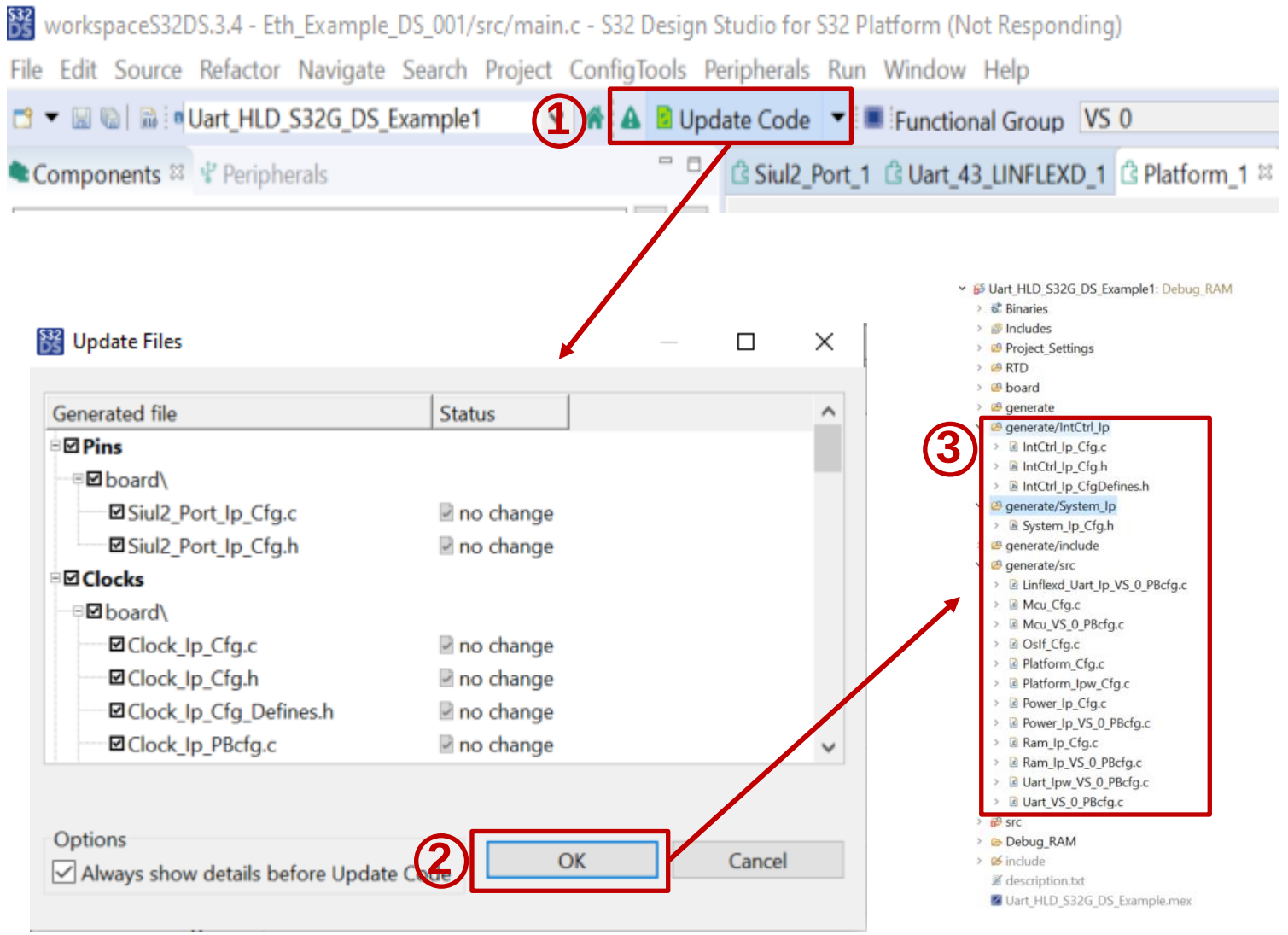
Generate code method:

1.Click on any configuration tool, like Pins

Then click **Update Code** (ensure desired project is selected!)

2.The Update Files window pops up. It shows the detail update information. Click **ok** button.

3.The configuration .c and .h file will be generated at “generate” folder.



HANDS ON UART: APPLICATION CODE 1

Open the main.c file in S32DS

```
/
int main(void)
{
    volatile Uart_StatusType Uart_Status;
    volatile Std_ReturnType Std_Uart_Status;
    uint32 varRemainingBytes;
    uint8 Rx_Buffer[MSG_LEN];

    Std_Uart_Status = E_NOT_OK;
    /* Initialize the Mcu driver */
     Mcu_Init(NULL_PTR);
     /* Initialize the clock tree and apply PLL as system clock */
     Mcu_InitClock(McuClockSettingConfig_0);
     while ( MCU_PLL_LOCKED != Mcu_GetPllStatus() )
     {
         /* Busy wait until the System PLL is locked */
     }

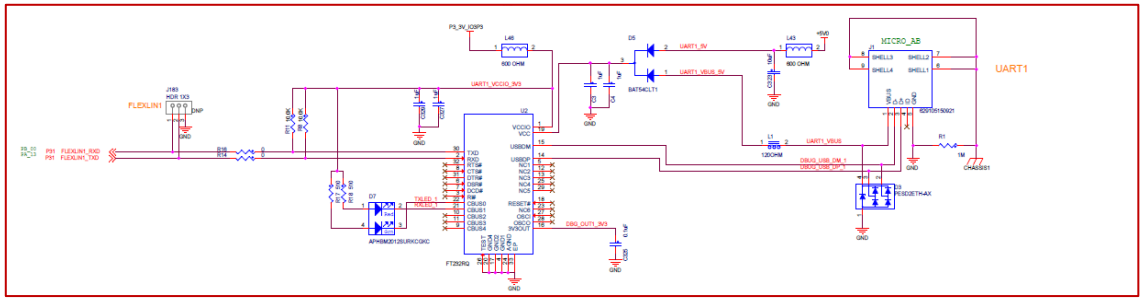
     /* Distribute the PLL across the clock tree */
     Mcu_DistributePllClock();

     /* Apply a mode configuration */
     Mcu_SetMode(McuModeSettingConf_0);

    /* Initialize all pins using the transceiver */
    Uart_Setup_Pins();

    /* Initialize IRQs */
    Platform_Init(NULL_PTR);
    Platform_InstallIrqHandler(LINFLEXD1_IRQn, LINFLEXD1_UART_IRQHandler, NULL_PTR);
```

MCU clock initiation




```
/* Init Pins */
void Uart_Setup_Pins(void)
{
    /* LINFLEXD1_TX: PA_13 */
    t_reg_write(0x4009C274, 0x00200002);
    /* LINFLEXD1_RX: PB_00 */
    t_reg_write(0x4009C280, 0x00080000);
    t_reg_write(0x44010DC0, 0x02);
}
```

In Uart_HLD_S32G_DS_Example Project.
Initialization of pins is writing register directly.

HANDS ON UART: APPLICATION CODE 2

```
while (1)
{
    /* Receive and store data byte by byte until new line character is received,
    * or the buffer becomes full
    */
    (void)Uart_AsyncReceive(UART_CHANNEL, Rx_Buffer, strlen(EXPECT_RX_MSG));
    /* Wait for transfer to be completed */
    while(Uart_GetStatus(UART_CHANNEL, &varRemainingBytes, UART_RECEIVE) == UART_OPERATION_ONGOING);

    /* Check the status */
    Uart_Status = Uart_GetStatus(UART_CHANNEL, &varRemainingBytes, UART_RECEIVE);

    if (Uart_Status != UART_NO_ERROR)
    {
        /* If an error occurred, send the error message and exit the loop */
        (void)Uart_AsyncSend(UART_CHANNEL, (const uint8 *)ERROR_MSG, strlen(ERROR_MSG));
        while(Uart_GetStatus(UART_CHANNEL, &varRemainingBytes, UART_SEND) == UART_OPERATION_ONGOING);
        break;
    }

    /* Send the received data back */
    Std_Uart_Status = Uart_AsyncSend(UART_CHANNEL, (const uint8 *)SEND_MSG, strlen(SEND_MSG));
    while(Uart_GetStatus(UART_CHANNEL, &varRemainingBytes, UART_SEND) == UART_OPERATION_ONGOING);
    break;
}
```

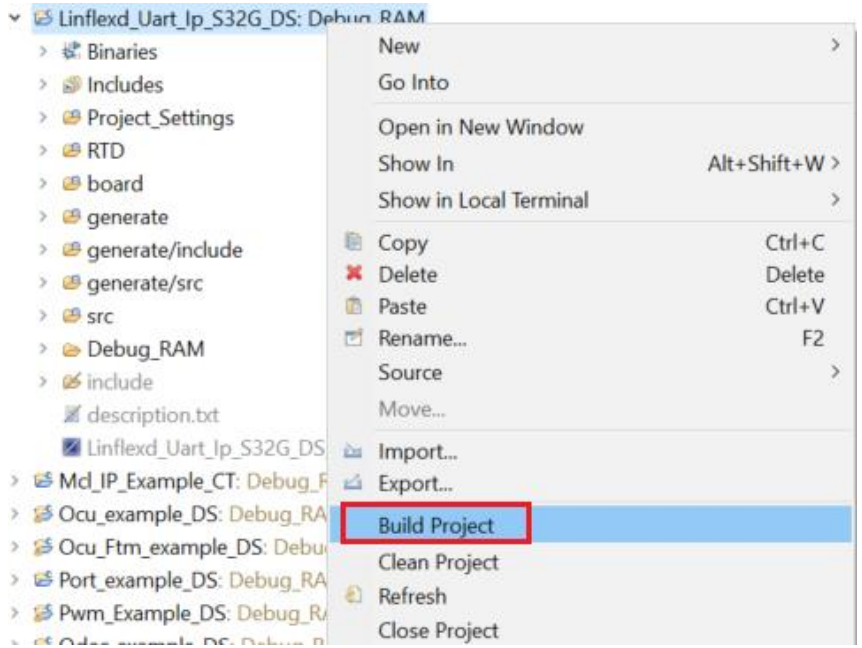
Receive data from user

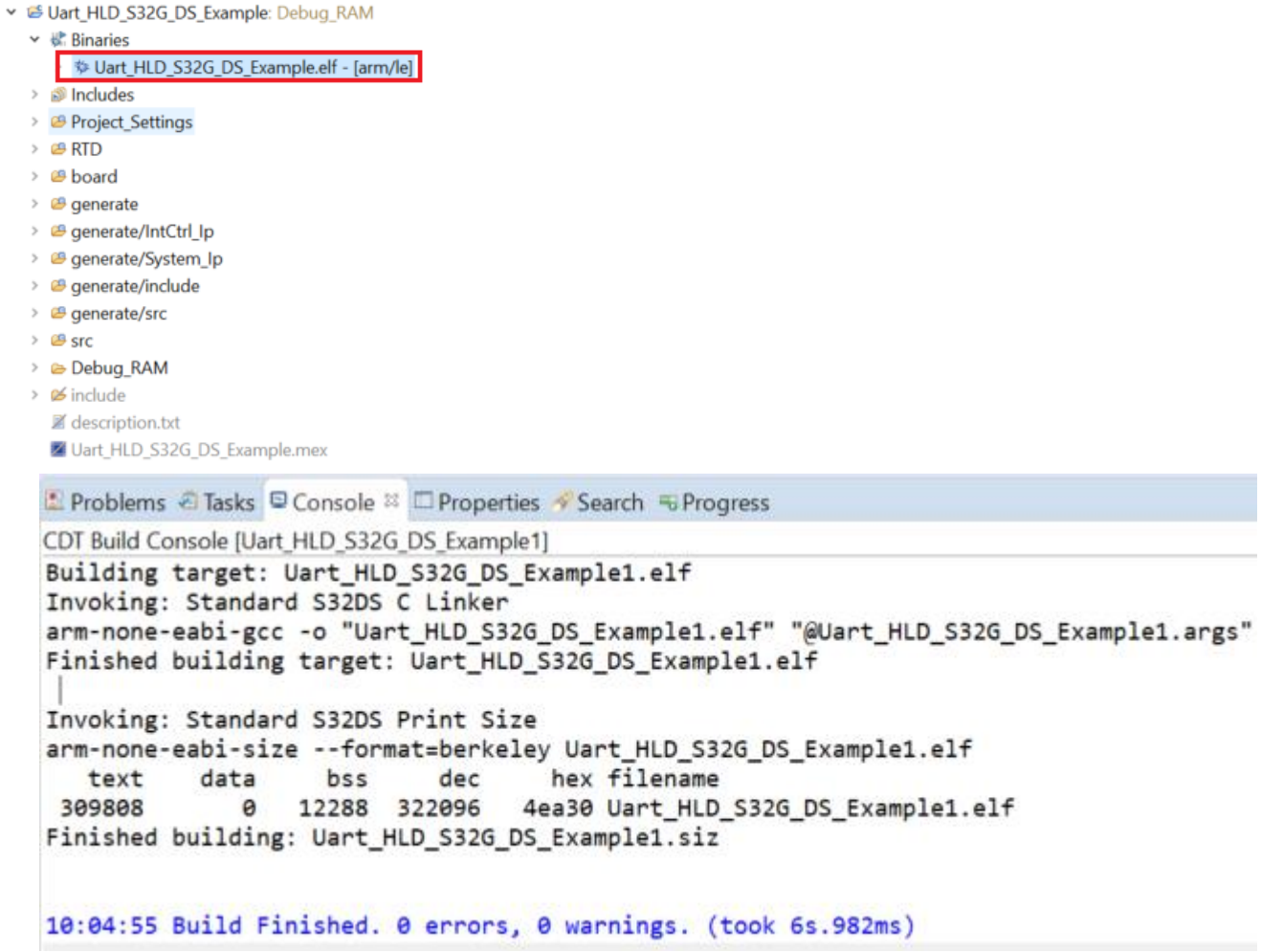
Echo the received data back

HANDS ON UART: BUILD AND DEBUG 1

Build the target :

- Right click on Project,
- Select Build Project
- Print Build information on Console window
- Uart_HLD_S32G_DS_Example1.elf is generated

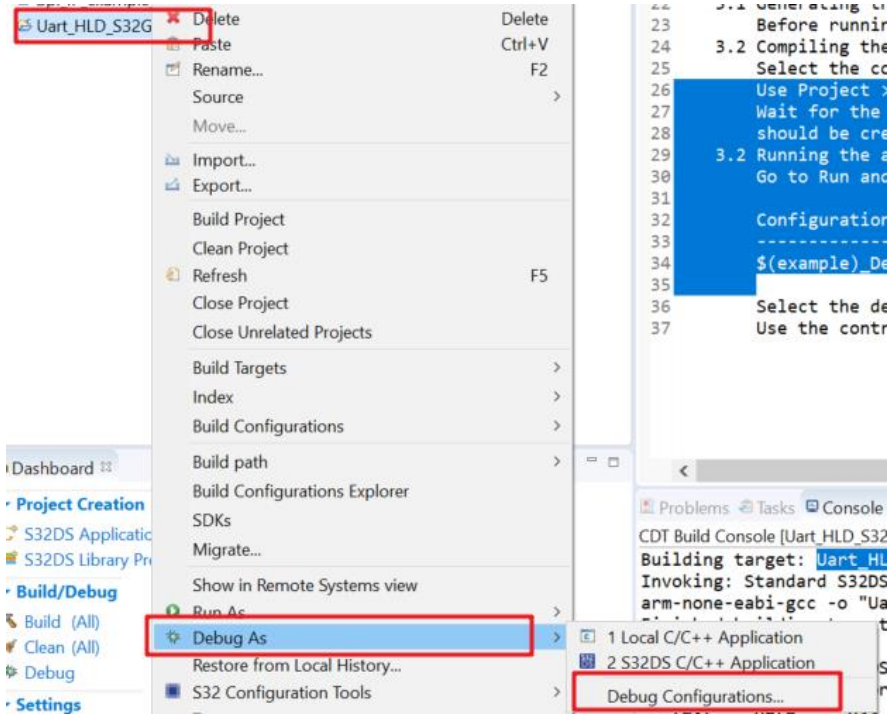




HANDS ON UART: BUILD AND DEBUG 2

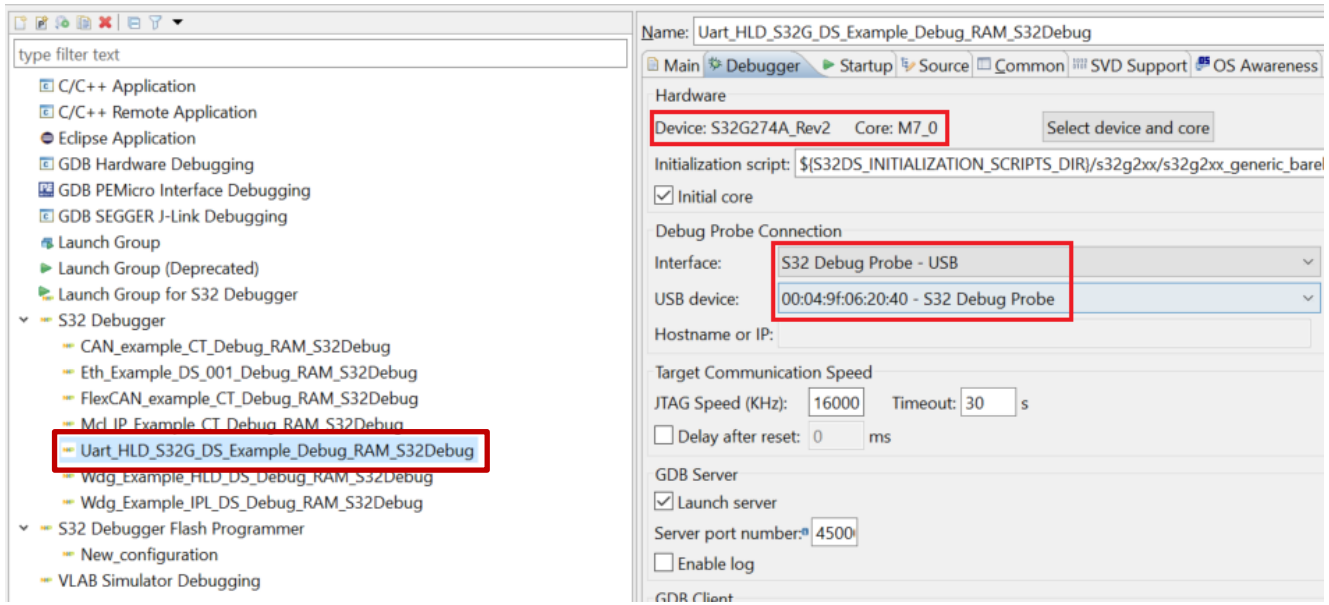
Go to debug configuration:

- Right click on Project,
- Select the Debug As
- Click Configurations



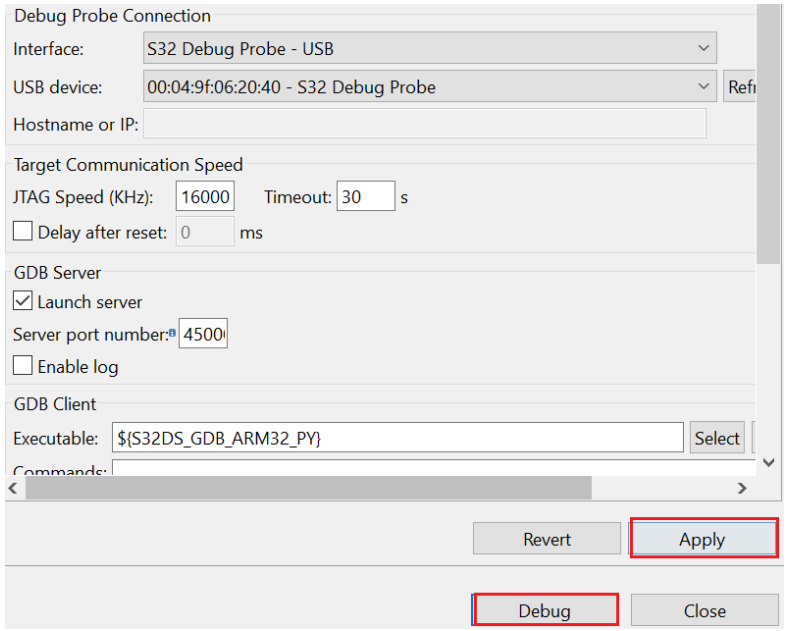
Debug configuration set:

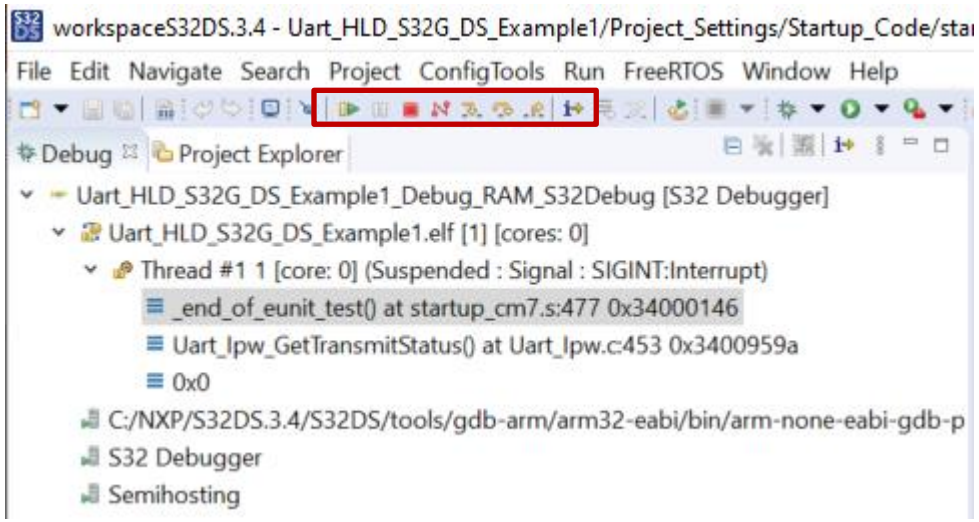
- Click target project ,
- Select the target device
- Select target S32 Debug Probe



HANDS ON UART : DEBUG AND RUN

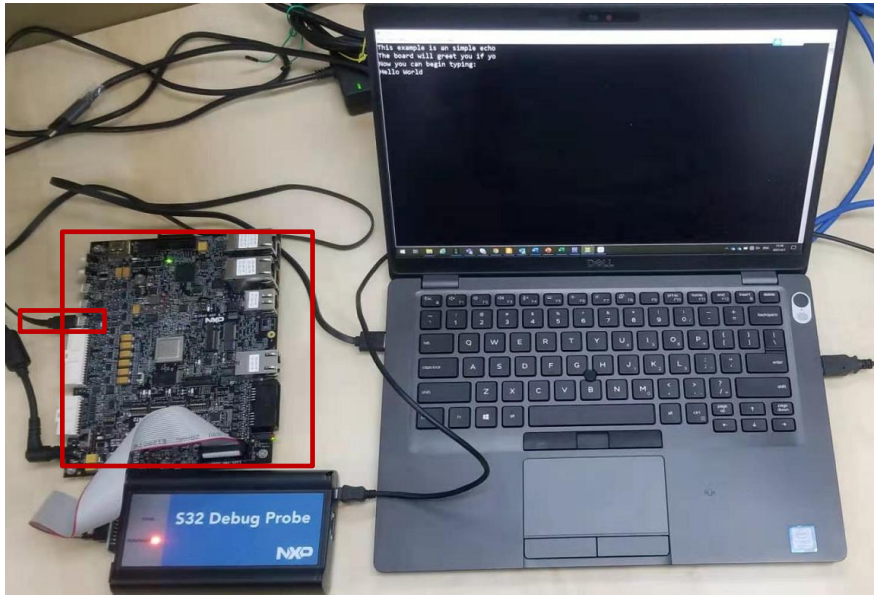
Click on “Apply”, then click on “Debug”. the perspective will jump to the Debug Perspective, and you can use the controls to control the program flow.

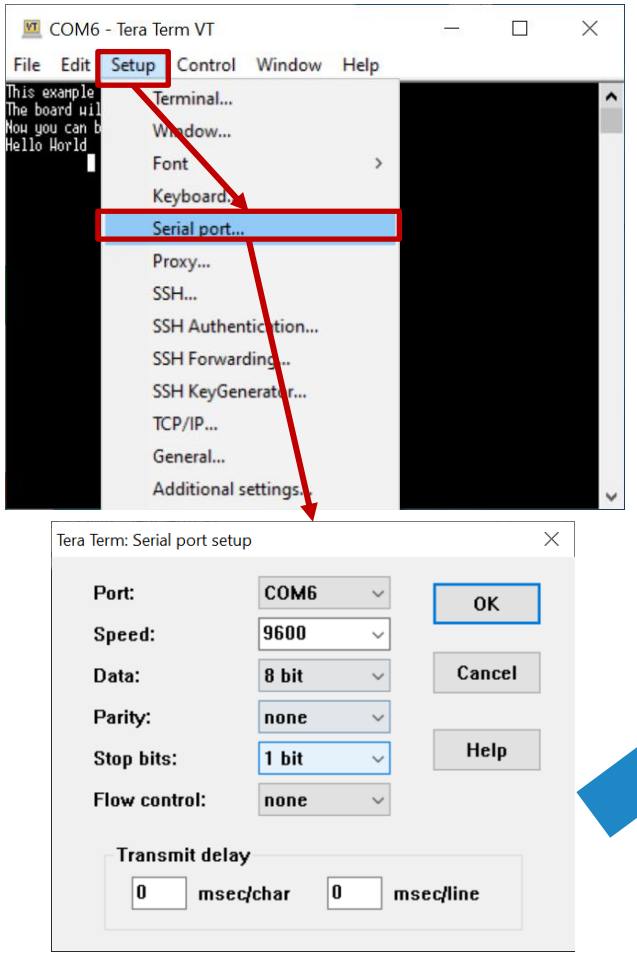


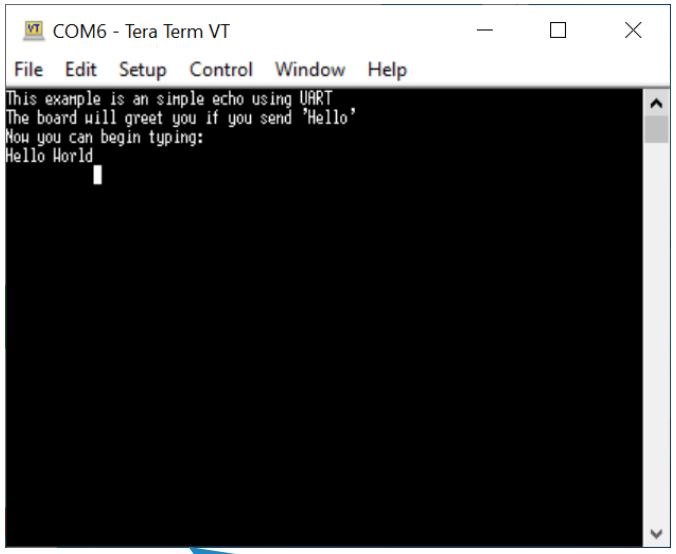


HANDS ON UART: TEST RESULT

- Connect the PC and UART1
- Open Tera Term and Set the serial port
- the terminal software will show the below messages. input “Hello”, UART output “Hello World”







Hands on ETH Example



SECURE CONNECTIONS
FOR A SMARTER WORLD



PUBLIC

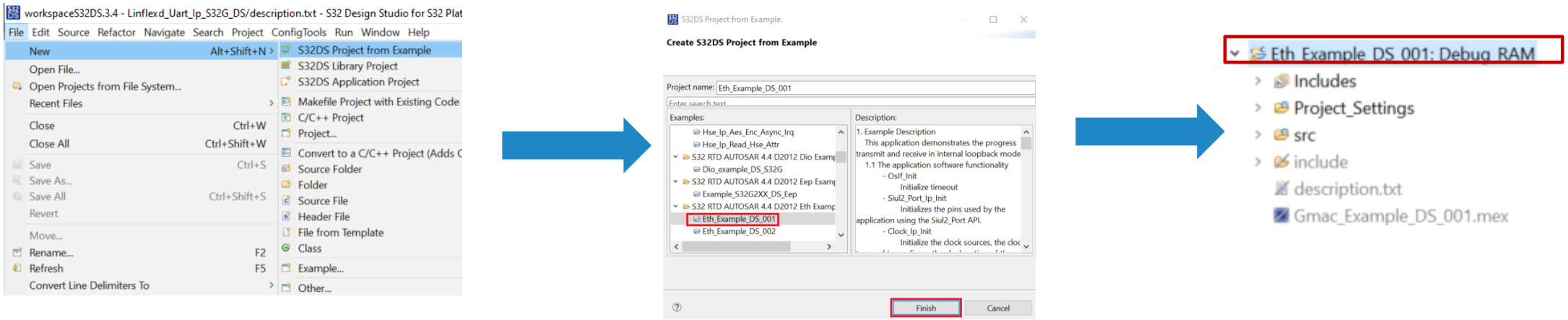
NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.

HANDS ON ETH – OBJECTIVE

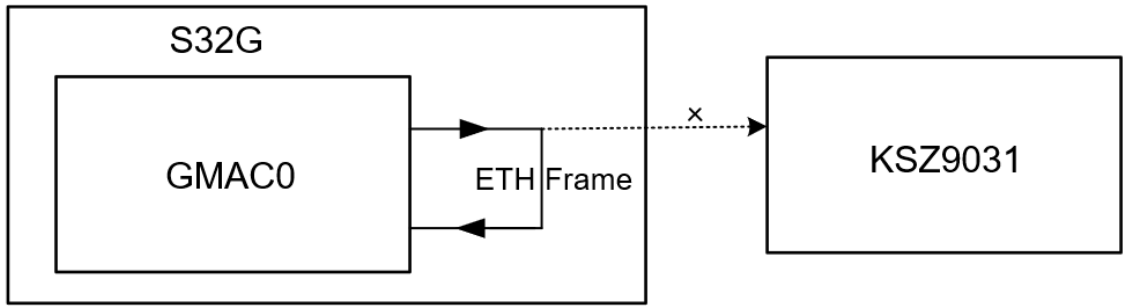
- How to import the ETH example into S32DS
- How to configure the clock of ETH via S32DS
- How to configure the port of ETH via S32DS
- How to use the ETH module to transmit/receive ETH frame
- How to debug the ETH example with S32 debug probe

HANDS ON ETH: IMPORT ETH EXAMPLE PROJECT

Open S32 Design Studio, go to “File -> New -> S32DS Project From Example”. Select “**Eth_Example_DS_001**” example, then click on “Finish”. The project is copied into current workspace.



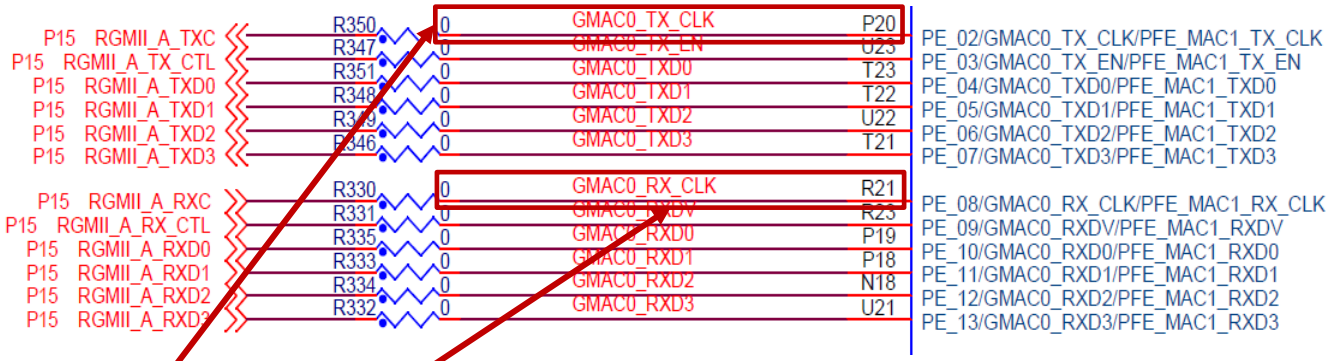
This “Gmac_Example_DS_001” example demonstrates the GMAC transmit and receive in internal loopback mode. The ETH frame is transmitted back directly through GMAC, and the frame will not be transmitted to PHY.

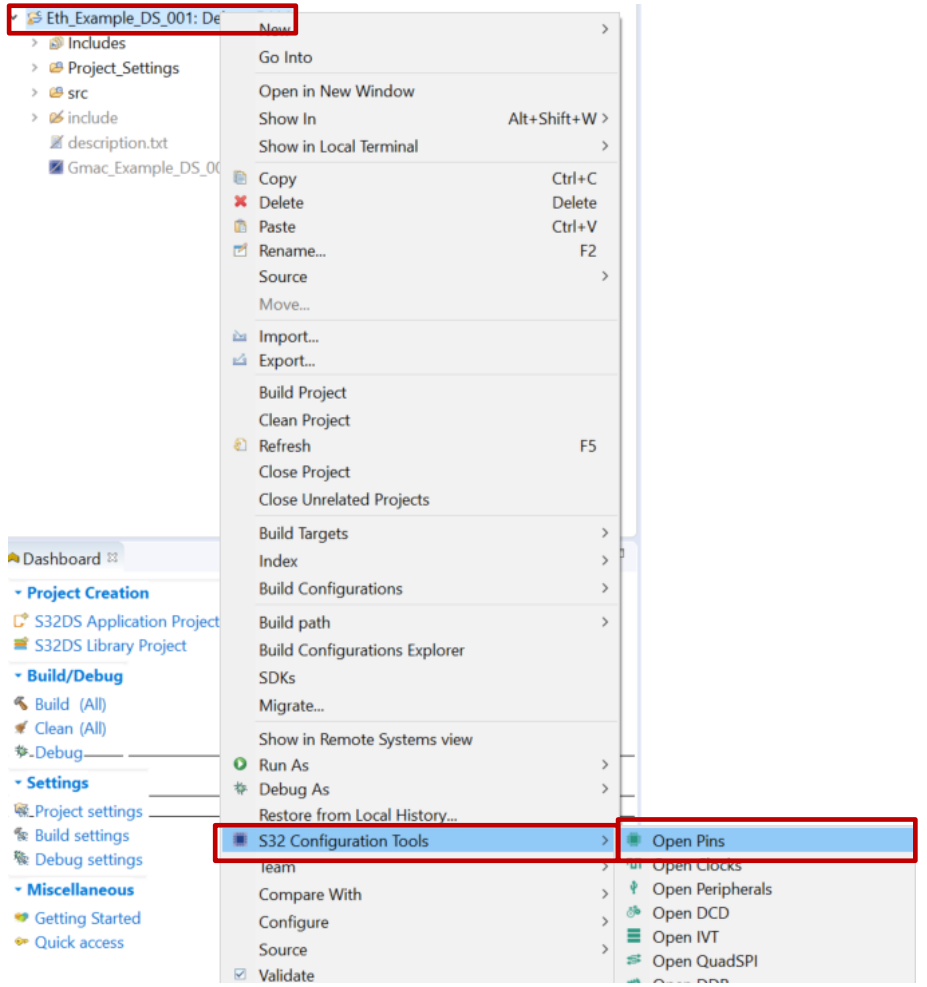


HANDS ON ETH : PORT CONFIGURATION

Go to desired configuration tool:

- Right click on Project,
- Select S32 Configuration Tool…
- Select Open Pins
- Configure pins to provide the external clock to Tx, Rx signals





Routed Pins for BOARD... 3

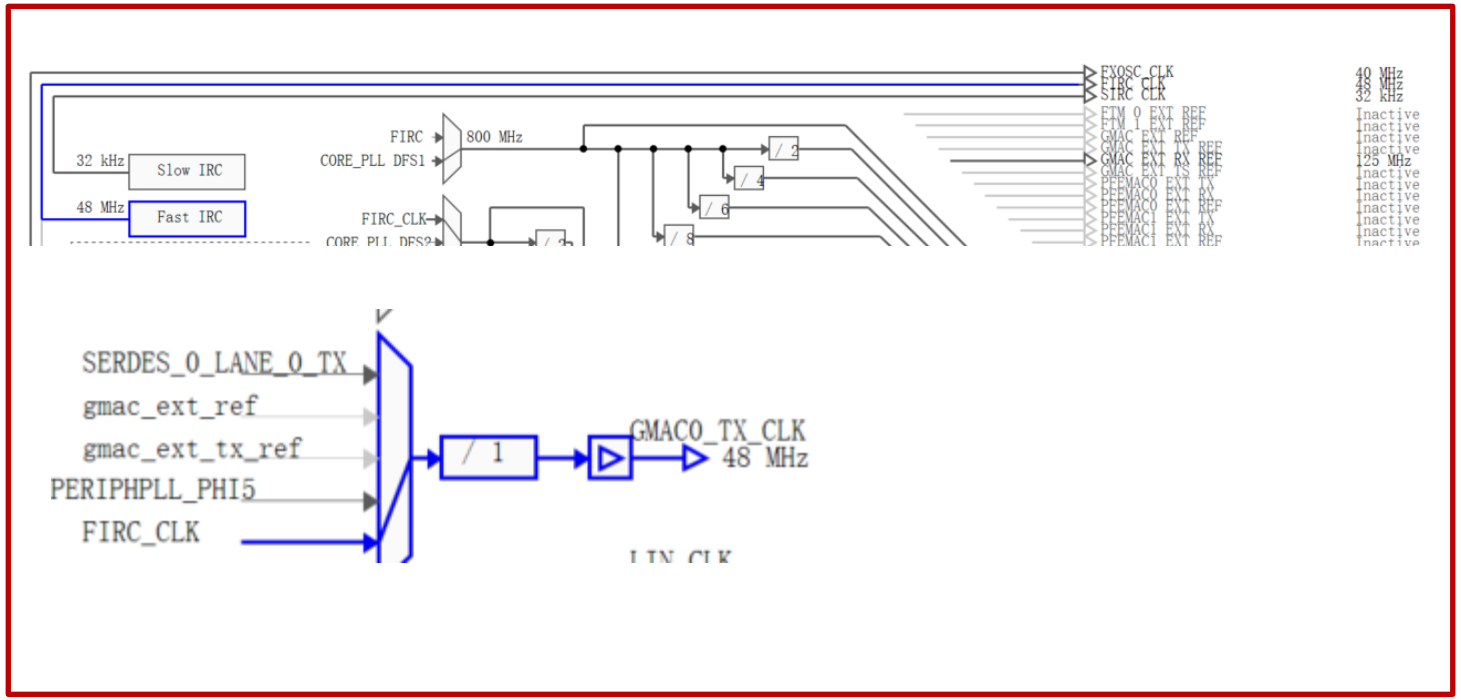
#	Peripheral	Signal	Route to	Label	Identifier	Power group	Direction	Output Buffer	Open Drain	Input Buffer	Slew Rate Control	Pullup Enab
P20	GMAC_0	tx_clk_o	PE_02		n/a	VDD_IO_GMAC0 (0V)	Output	Enabled	Disabled	Disabled	SIUL2_0: FAST pad: 166MHz(1.8V), 150MHz(3.3V) - GPIO pad: 166MHz(1.8V), 50Mhz(3.3V) / SIUL2_1: 150MHz or lower	Disabled
R21	GMAC_0	rx_clk_i	PE_08		n/a	VDD_IO_GMAC0 (0V)	Input	Disabled	Disabled	Enabled	SIUL2_0: FAST pad: 166MHz(1.8V), 150MHz(3.3V) - GPIO pad: 166MHz(1.8V), 50Mhz(3.3V) / SIUL2_1: 150MHz or lower	Disabled

HANDS ON ETH : CLOCK CONFIGURATION

Open the **Peripheral Clock View**, Double click the GMAC0 module. The **Clocks Diagram** shows the power tree of GMAC module

Overview | Periphera... | Code Pre... | Registers | Details | Cloc

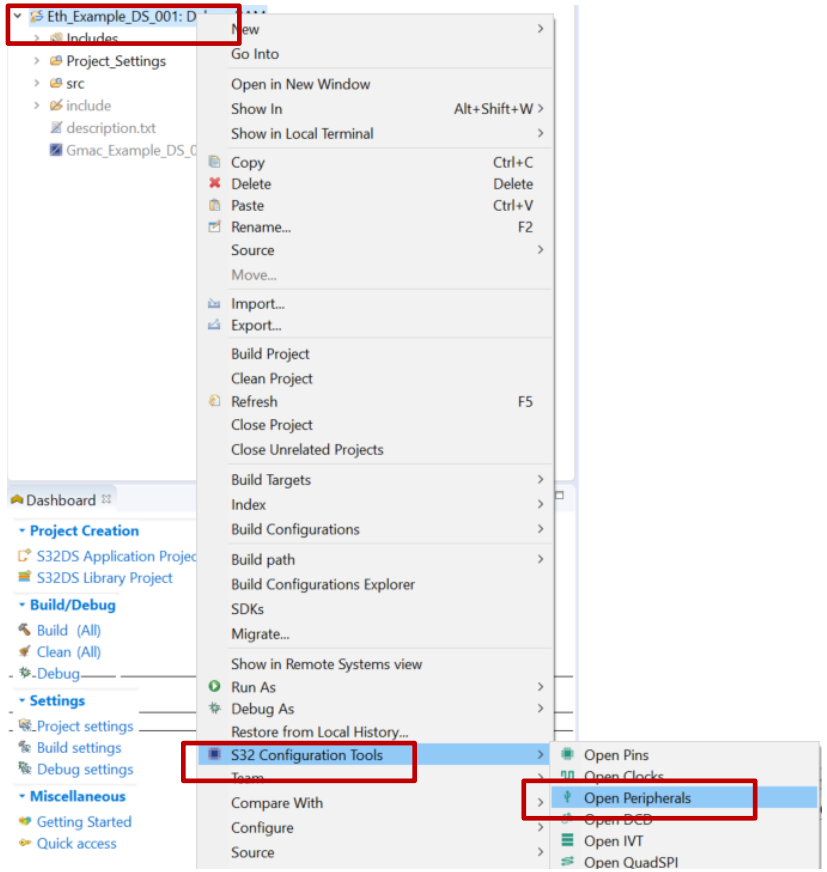
Clock Name	Enable	Source	Divider	Frequency	Monitor
EIM3_CLK	☑	XBA...	/ 1	66.66... ...	
EIM_CLK	☑	XBA...	/ 1	66.66... ...	
ERM0_CLK	☑	XBA...	/ 1	66.66... ...	
FLEXCAN0_CLK	☑	FIRC...	/ 1	48 MHz	
FLEXCAN1_CLK	☑	FIRC...	/ 1	48 MHz	
FLEXCAN2_CLK	☑	FIRC...	/ 1	48 MHz	
FLEXCAN3_CLK	☑	FIRC...	/ 1	48 MHz	
FRAY0_CLK	☑	FIRC...	/ 1	48 MHz	
FTIMER0_CLK	☑	FIRC...	/ 1	48 MHz	
FTIMER1_CLK	☑	FIRC...	/ 1	48 MHz	
GMAC0_RX_CLK	☑	FIRC...	/ 1	48 MHz	
GMAC0_TS_CLK	☑	FIRC...	/ 1	48 MHz	
GMAC0_TX_CLK	☑	FIRC...	/ 1	48 MHz	
IIC0_CLK	☑	XBA...	/ 1	133.33.....	
IIC1_CLK	☑	XBA...	/ 1	133.33.....	
IIC2_CLK	☑	XBA...	/ 1	133.33.....	
IIC3_CLK	☑	XBA...	/ 1	133.33.....	



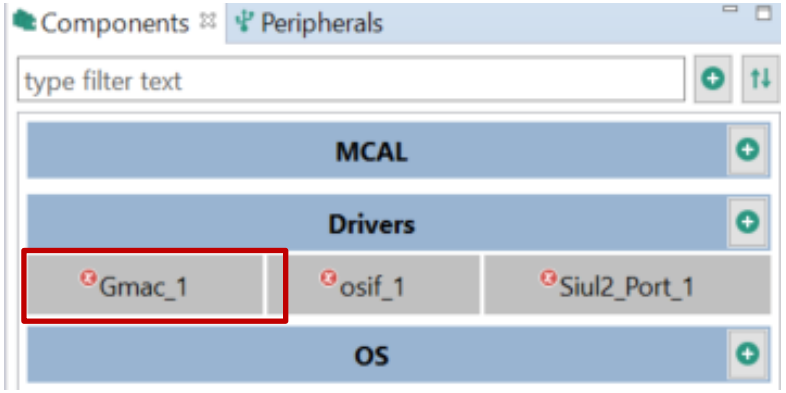
HANDS ON ETH: ETH CONFIGURATION

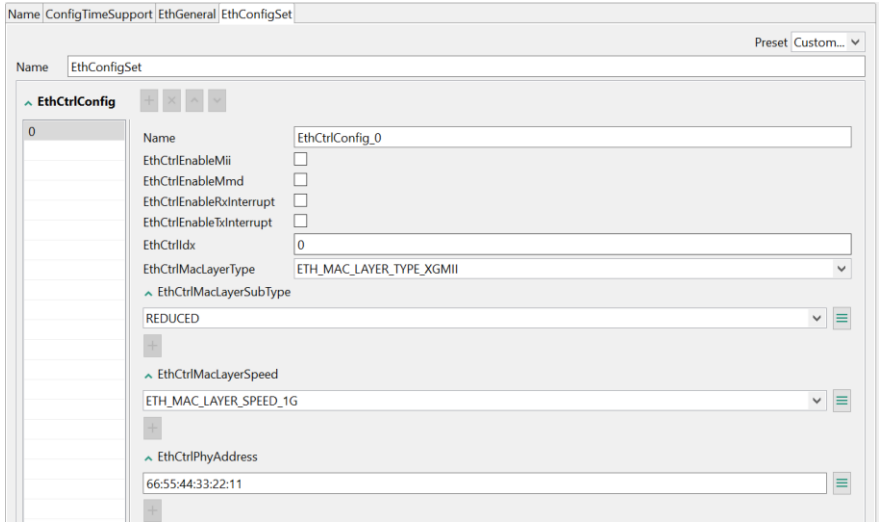
Open the peripheral configuration:

- Right click on Project,
- Select S32 Configuration Tool…
- Select Peripherals



Select **Components** to find out **GMAC_1** Driver and double click





HANDS ON ETH: UPDATE CODE

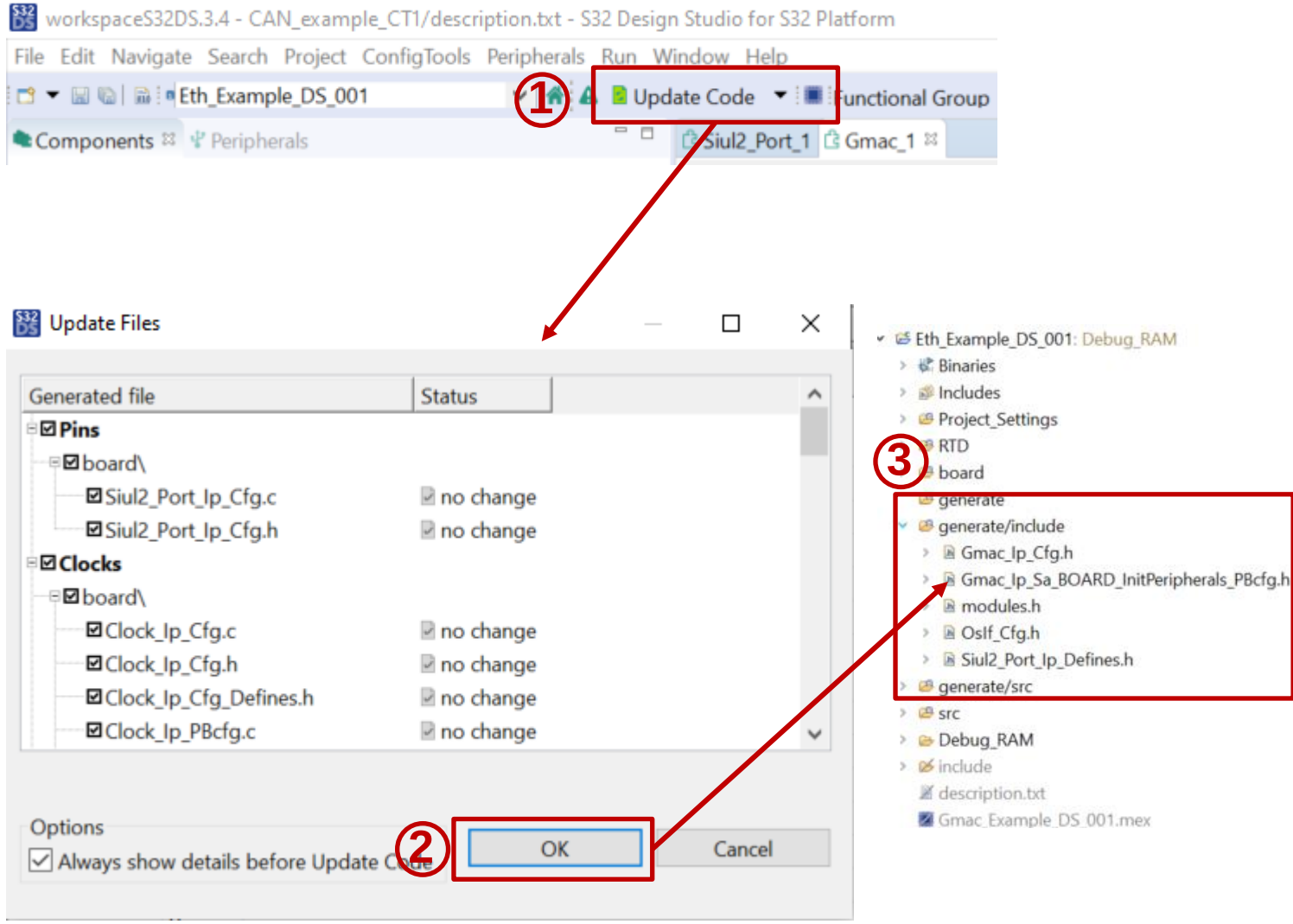
Generate code method:

1.Click on any configuration tool, like Pins

Then click **Update Code** (ensure desired project is selected!)

2.The Update Files widow pops up. It shows the detail update information. Click **ok** button.

3.The configuration .c and .h file will be generated at “generate” folder.



HANDS ON ETH: APPLICATION CODE 1

```
int main(void)
{
    Gmac_Ip_TxOptionsType txOptions = {TRUE, GMAC_CRC_AND_PAD_INSERTION, GMAC_CHECKSUM_INSERTION_DISABLE};
    Gmac_Ip_BufferType txBuffer = {0};
    Gmac_Ip_BufferType rxBuffer = {0};
    Gmac_Ip_TxInfoType txInfo;
    Gmac_Ip_RxInfoType rxInfo;
    Gmac_Ip_StatusType status;
    uint8 macAddr[6U] = {0U};
    uint8 i;
    uint8 j = 0U;
    boolean result = TRUE;

    OsIf_Init(NULL_PTR);

    Siul2_Port_Ip_Init(NUM_OF_CONFIGURED_PINS0, g_pin_mux_InitConfigArr0);

    Clock_Ip_Init(&Mcu_aClockConfigPB[0]);

    Gmac_Ip_Init(INST_GMAC_0, &Gmac_0_ConfigPB_BOARD_INITPERIPHERALS);

    /* Setup the frame with the Mac address and size */
    Gmac_Ip_GetMacAddr(INST_GMAC_0, macAddr);

    /* Request a buffer of at least 64 bytes */
    txBuffer.length = 64U;
    if ((GMAC_STATUS_SUCCESS != Gmac_Ip_GetTxBuff(INST_GMAC_0, 0U, &txBuffer, NULL_PTR)) || (txBuffer.length < 64U))
    {
        result = FALSE;
    }

    for (i = 0U; i < 12U; i++)
    {
        *((uint8 *)(&txBuffer.data[0U] + i)) = macAddr[0 + j];
        if (j < 5U)
        {
            j++;
        }
        else
        {
            j = 0U;
        }
    }
```

Initialize pins to provide the external clock to Tx, Rx signals via the function Siul2_Port_Ip_Init
Initialize clock to Tx, Rx signals via the function Clock_Ip_Init

Enable controller, initialize Tx and Rx buffer via the function Gmac_Ip_Init

initialize transmit buffer and Borrow transmit area to load frame via the function Gmac_Ip_GetTxBuff

HANDS ON ETH: APPLICATION CODE 2

```
/* Payload = Frame - (DstAddr + SrcAddr + EtherType/Length + FCS) */
*((uint32 *)(txBuffer.data + 13U)) = 64U - (6U + 6U + 2U + 4U);

/* Send the ETH frame */
txBuffer.length = 64U - 4U;     /* Don't count FCS, because it is automatically inserted by the controller in this example */
if (GMAC_STATUS_SUCCESS != Gmac_Ip_SendFrame(INST_GMAC_0, 0U, &txBuffer, &txOptions))
{
    result = FALSE;
}

/* Wait for the frame to be transmitted */
do {
    status = Gmac_Ip_GetTransmitStatus(INST_GMAC_0, 0U, &txBuffer, &txInfo);
} while (status == GMAC_STATUS_BUSY);

/* Check the frame status */
if ((GMAC_STATUS_SUCCESS != status) || (0U != txInfo.errMask))
{
    result = FALSE;
}

/* Wait for the frame to be received */
do {
    status = Gmac_Ip_ReadFrame(INST_GMAC_0, 0U, &rxBuffer, &rxInfo);
} while (status == GMAC_STATUS_RX_QUEUE_EMPTY);

/* Check the frame status */
if ((GMAC_STATUS_SUCCESS != status) || (0U != rxInfo.errMask))
{
    result = FALSE;
}

Gmac_Ip_ProvideRxBuff(INST_GMAC_0, 0U, &rxBuffer);

Gmac_Ip_DisableController(INST_GMAC_0);
```

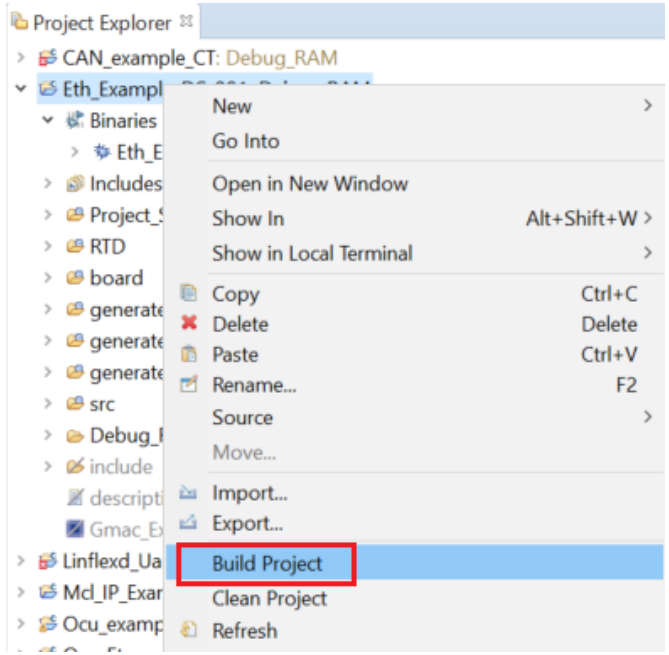
Trigger the transmit frame via Gmac_Ip_SendFrame

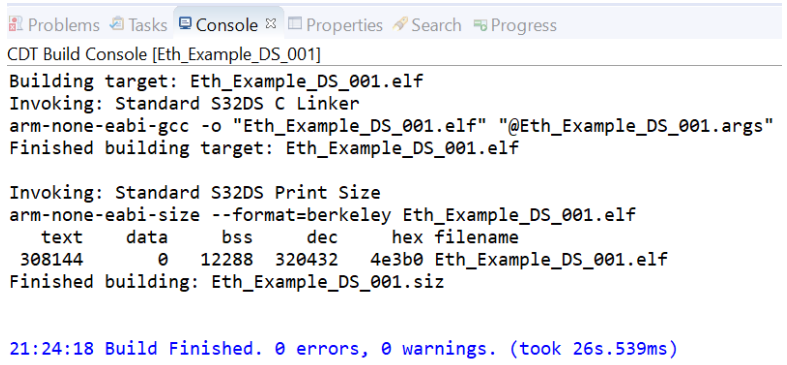
Verify frame is transmitted/ received

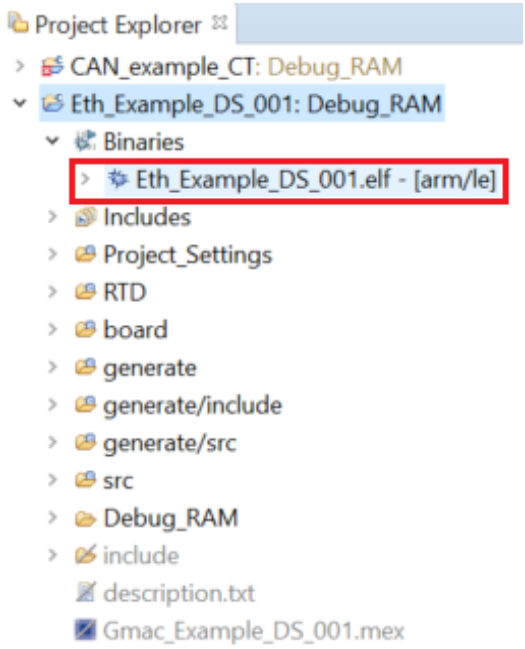
HANDS ON ETH: BUILD AND DEBUG 1

Build target Project:

- Right click on Project,
- Build Project
- The console print build information
- Eth_Example_DS_001.elf is created



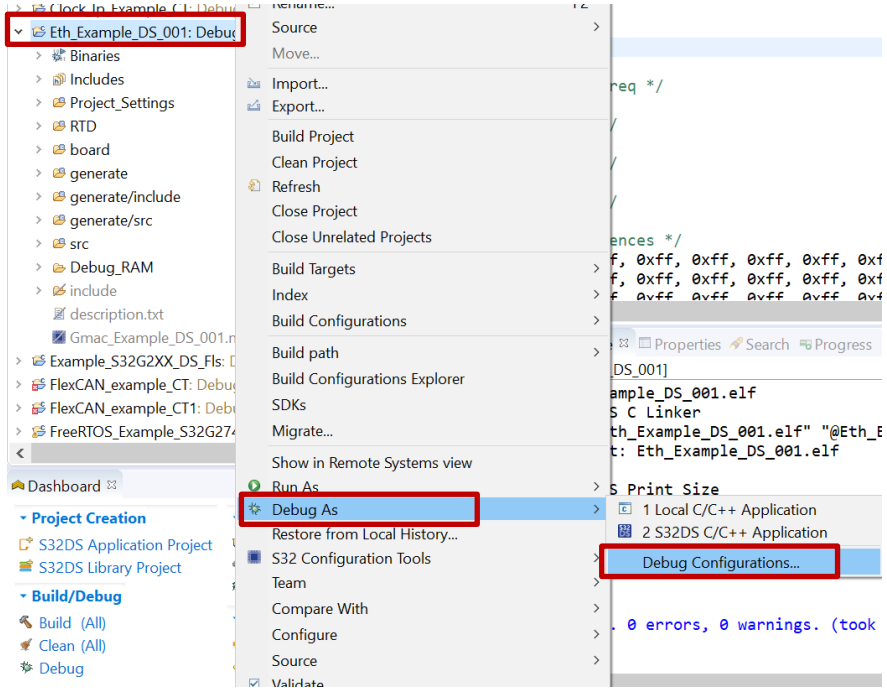




HANDS ON ETH: BUILD AND DEBUG 2

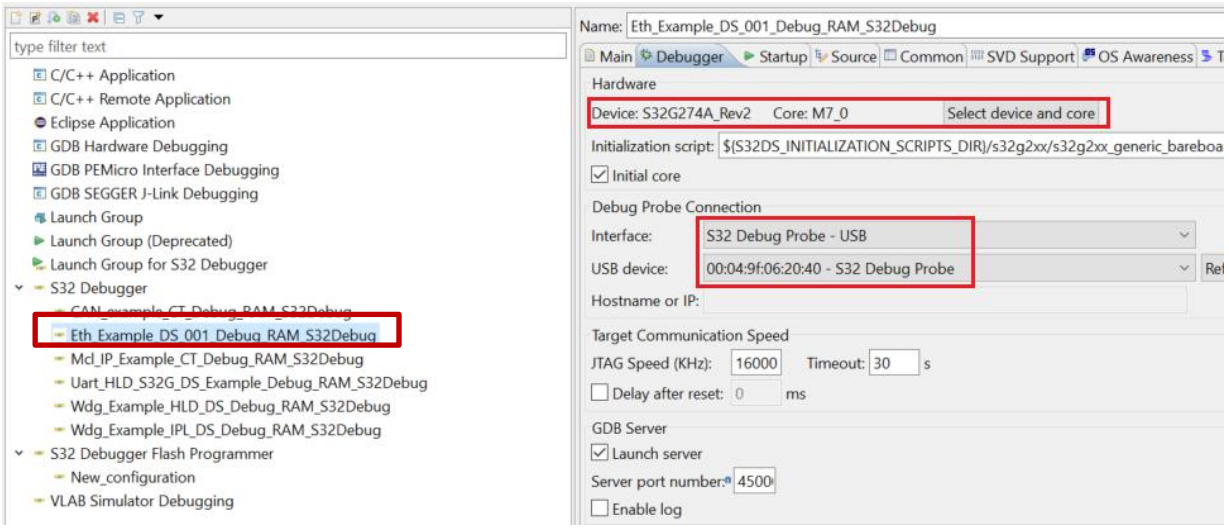
Go to debug configuration:

- Right click on Project,
- Select the Debug As
- Click Configurations



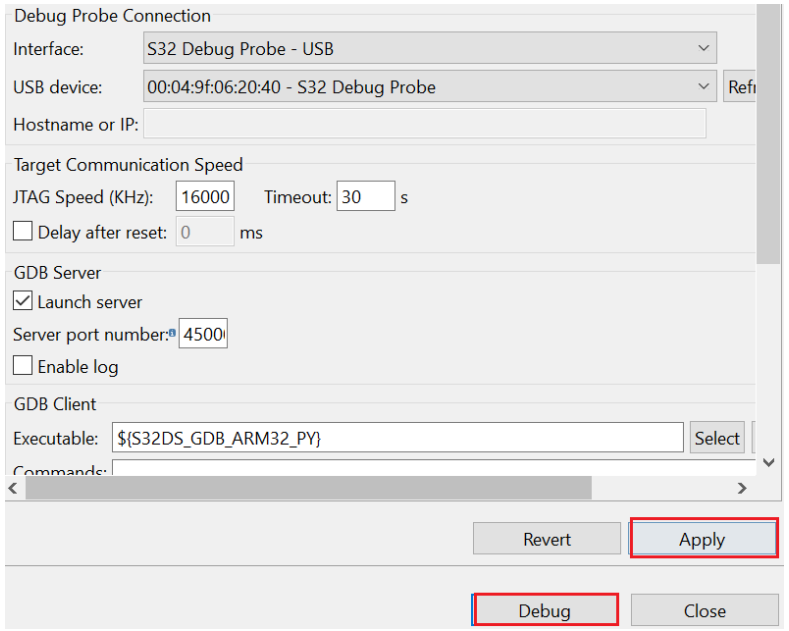
Debug configuration set:

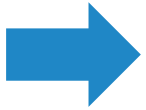
- Click target project ,
- Select the target device
- Select target S32 Debug Probe

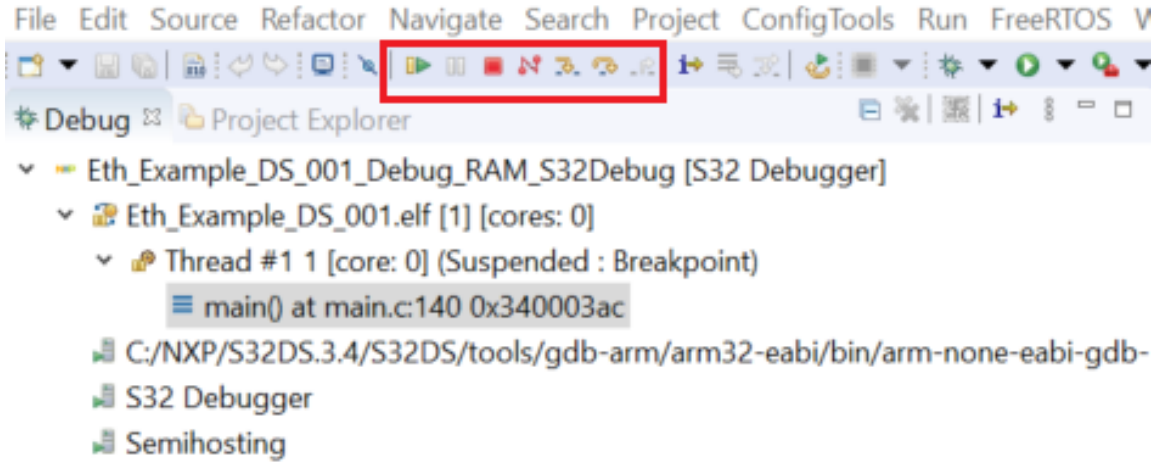


HANDS ON ETH: DEBUG AND RUN

Click on “Apply”, then click on “Debug”. the perspective will jump to the Debug Perspective, and you can use the controls to control the program flow.

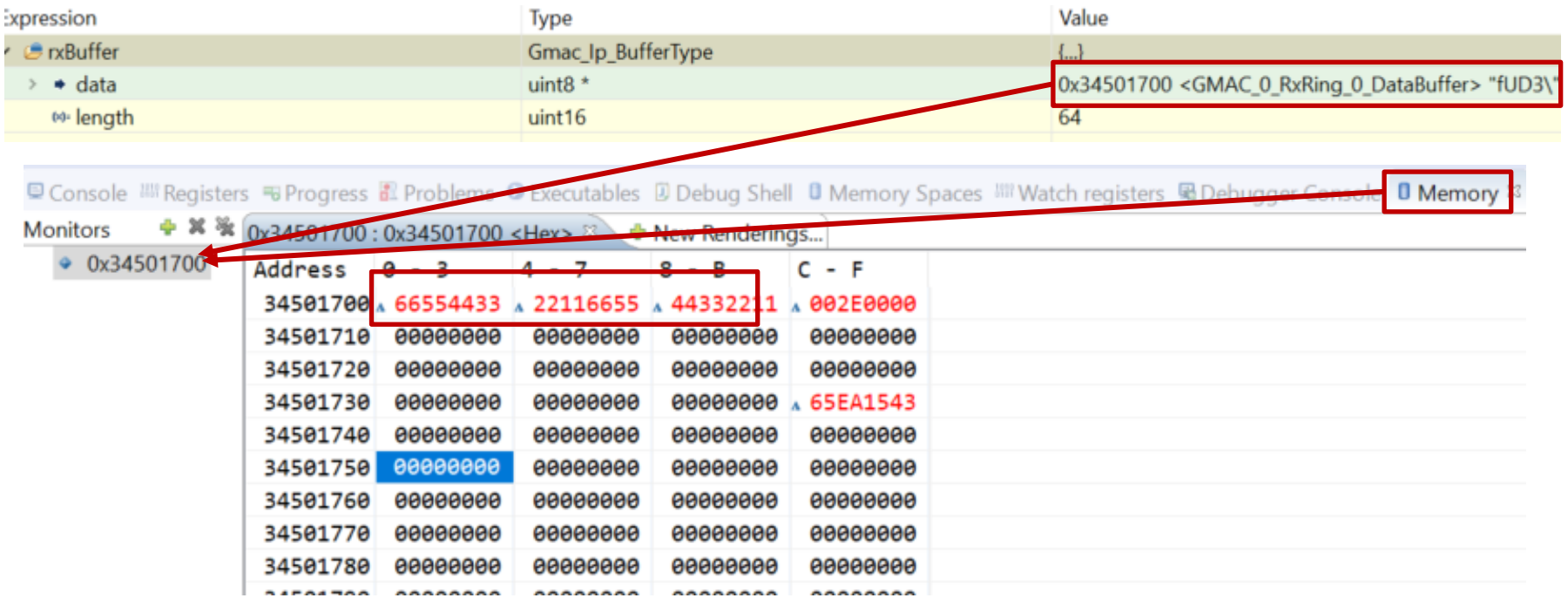






HANDS ON ETH: TEST RESULT

In this project. The eth frame of Transmit & receive in internal loopback mode. The rxBuffer shows the received frame.



Hands on CAN Example



SECURE CONNECTIONS
FOR A SMARTER WORLD



PUBLIC

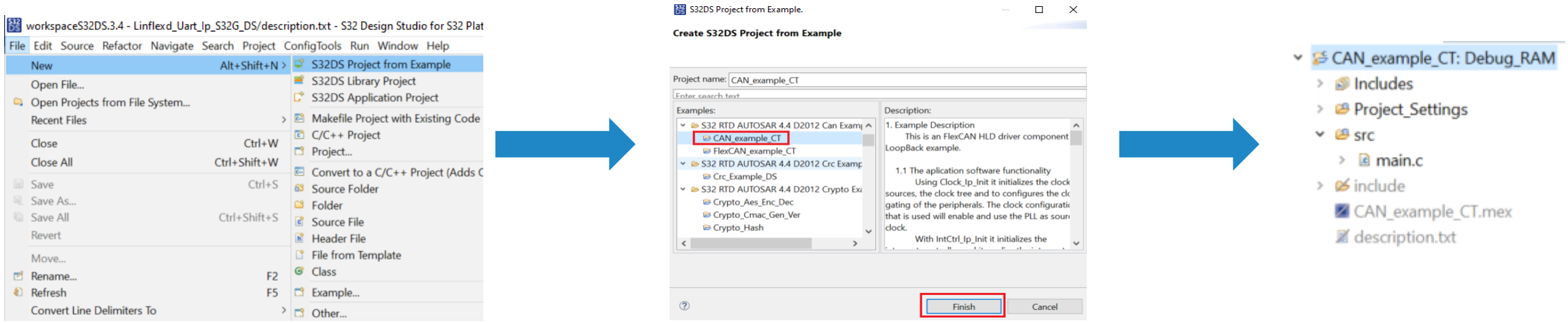
NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.

HANDS ON CAN – OBJECTIVE

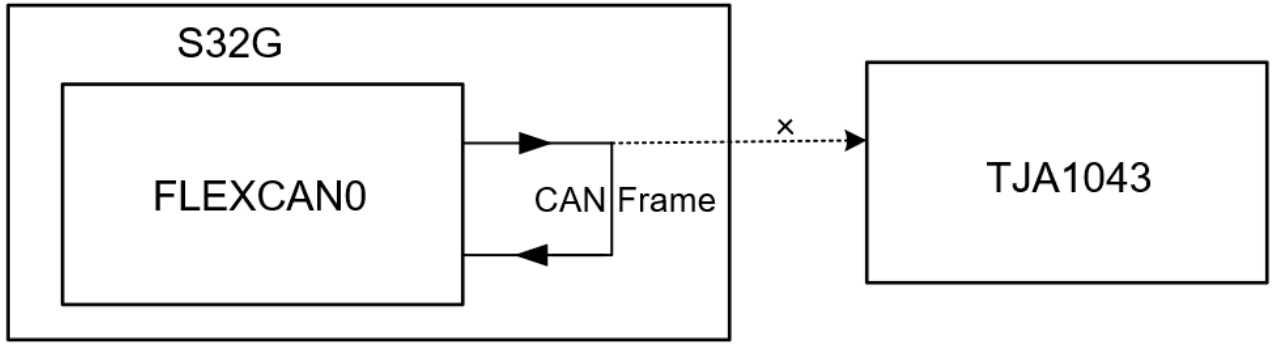
- How to import the CAN example into S32DS
- How to configure the clock of CAN via S32DS
- How to configure the port of CAN via S32DS
- How to modify the CAN loopback
- How to debug the CAN example with S32 debug probe

HANDS ON CAN : IMPORT CAN EXAMPLE PROJECT

Open S32DS3.4, go to “File -> New -> S32DS Project From Example”. Select “**CAN_example_CT**” example, then click on “Finish”. The project is copied into current workspace.

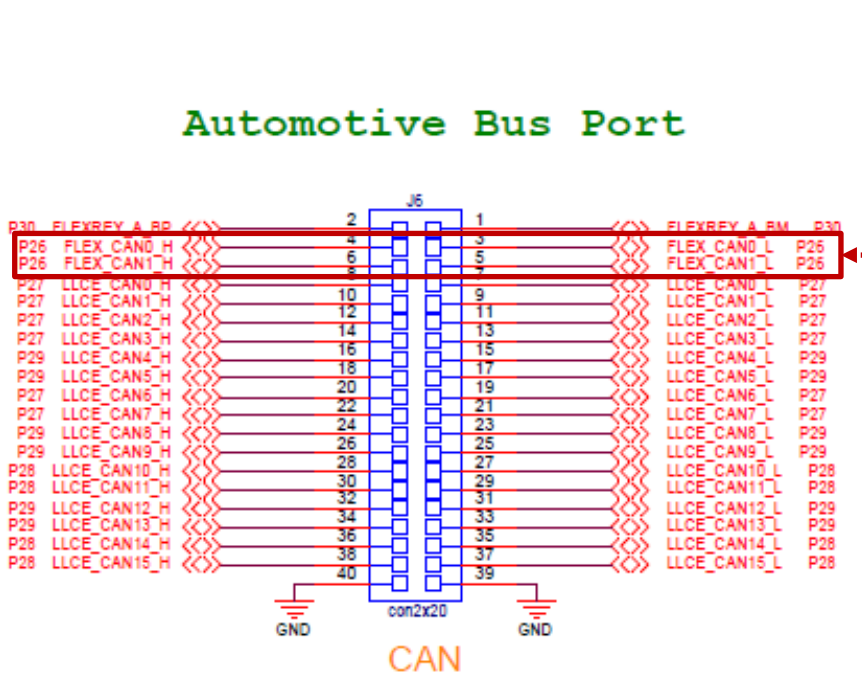


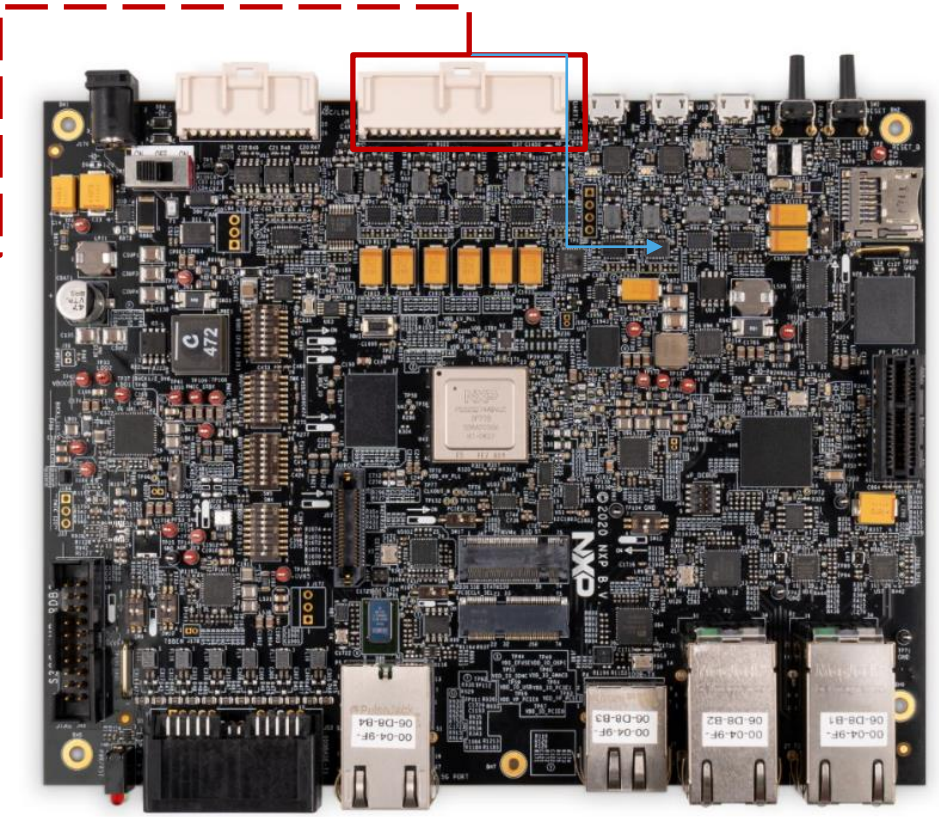
“CAN_example_CT” project is a FlexCAN HLD driver component LoopBack project.



HANDS ON CAN: THE PROPOSED DEMO NEED TO MODIFY

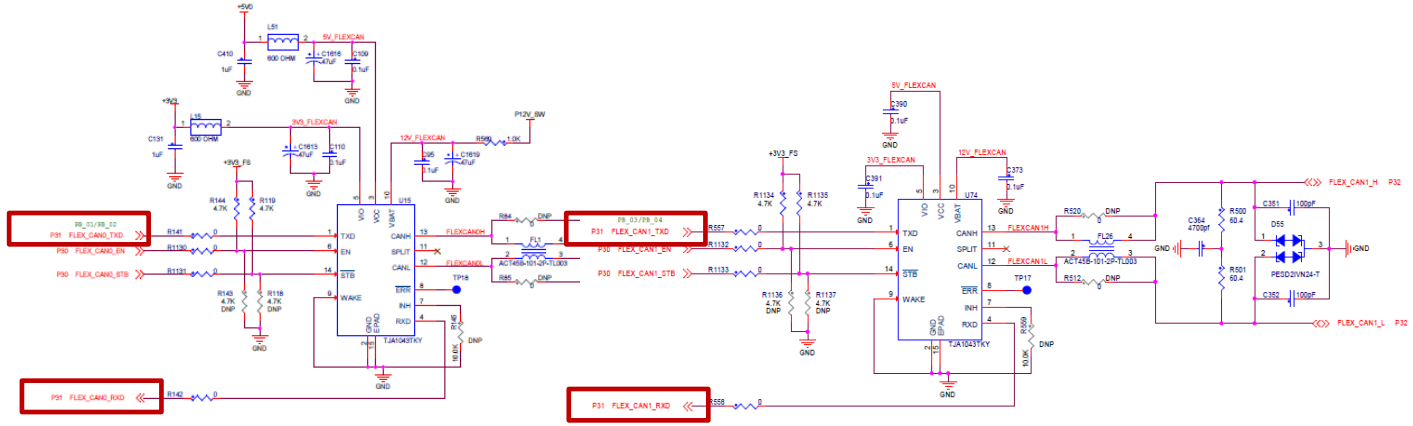
The “CAN_example_CT” project only support loopback model. modify this default project configuration to build transmit/receive CAN frame from FlexCAN_0 to FlexCAN_1

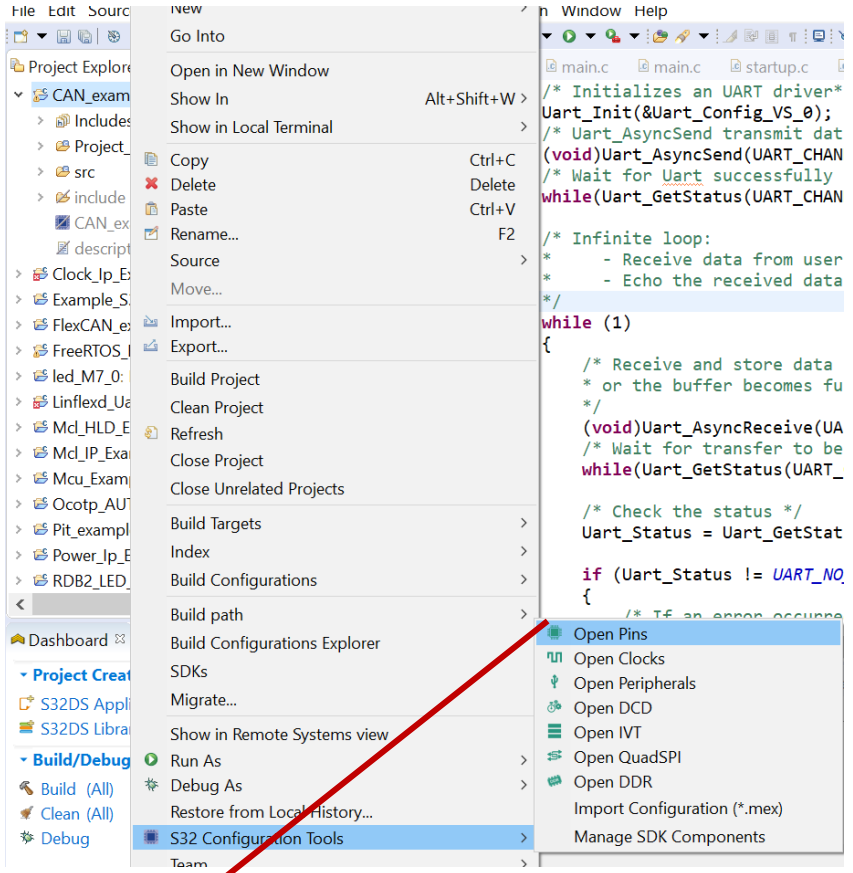




HANDS ON CAN: PORT CONFIGURATION 1

- Go to desired configuration tool:
  - Right click on Project,
  - Select S32 Configuration Tool…
  - Select Open Pins
  - Modify the Pins as the schematic of CAN0 and CAN 1





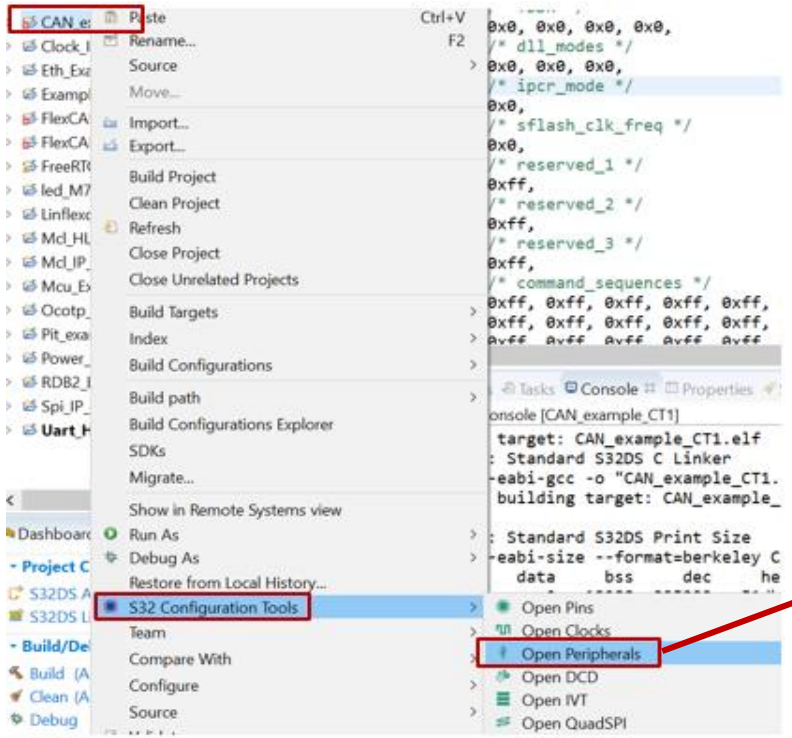
Routed Pins for BOARD... 4

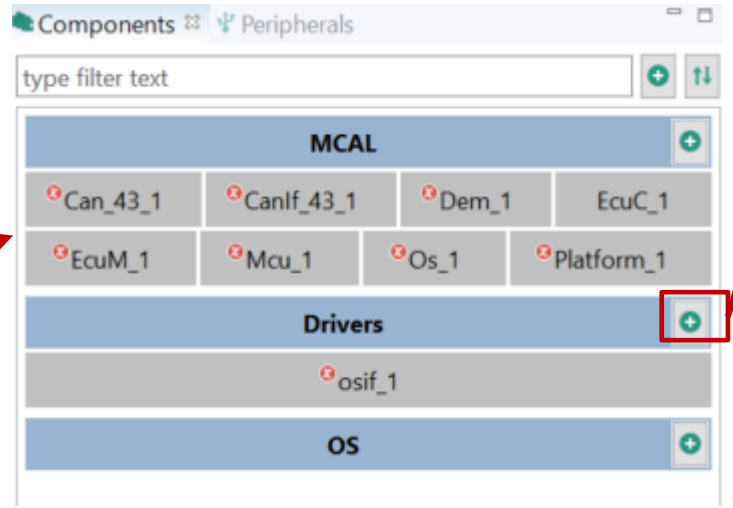
#	Peripheral	Signal	Route to	Label	Identifier	Power group	Direction	Output Buffer	Open Drain	Input Buffer
D7	CAN_0	rxd	PB_02		n/a	VDD_IO_B (0V)	Input	Disabled	Disabled	Enabled
E7	CAN_0	txd	PB_01		n/a	VDD_IO_B (0V)	Output	Enabled	Disabled	Disabled
E8	CAN_1	rxd	PB_04		n/a	VDD_IO_B (0V)	Input	Disabled	Disabled	Enabled
C6	CAN_1	txd	PB_03		n/a	VDD_IO_B (0V)	Output	Enabled	Disabled	Disabled

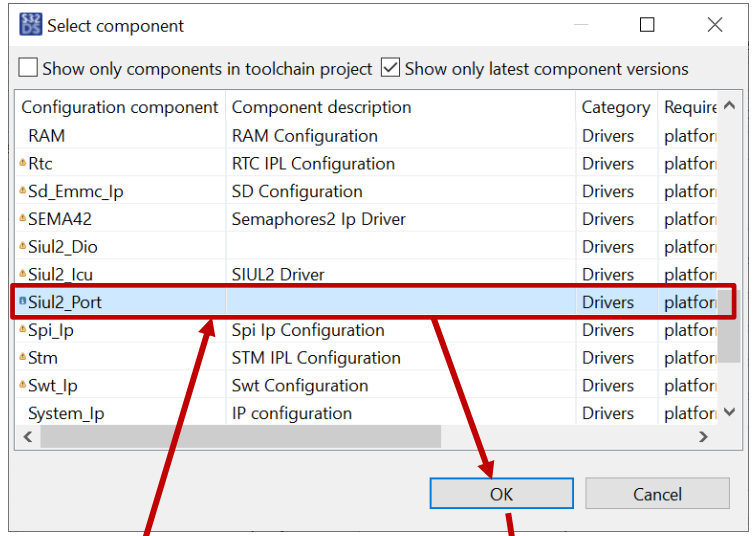
HANDS ON CAN: PORT CONFIGURATION 2

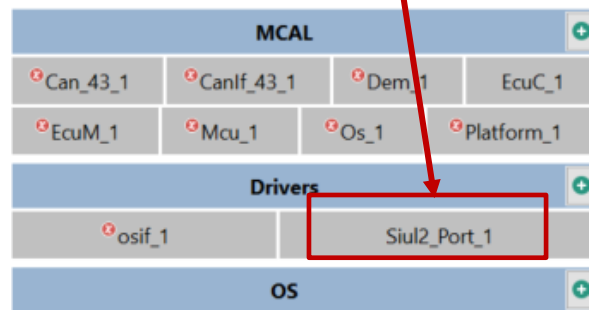
- Add the Port configuration:
  - Right click on Project,
  - Select S32 Configuration Tool…
  - Select Open Peripherals

- Click the plus button
- Click the Siul2_Port component
- The Siul2_Port_1 will be added





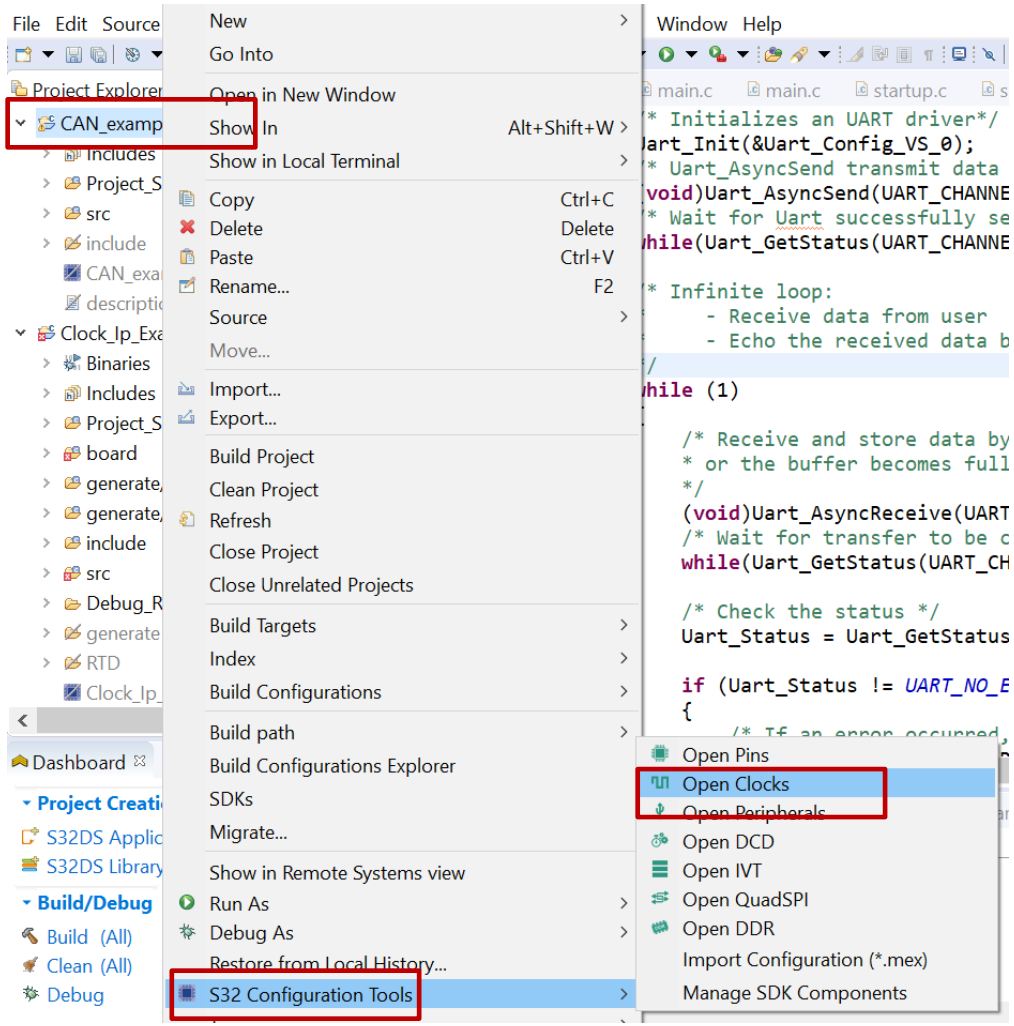




HANDS ON CAN: CLOCK CONFIGURATION 1

Go to desired configuration tool:

- Right click on Project,
- Select S32 Configuration Tool…
- Select Open Clocks

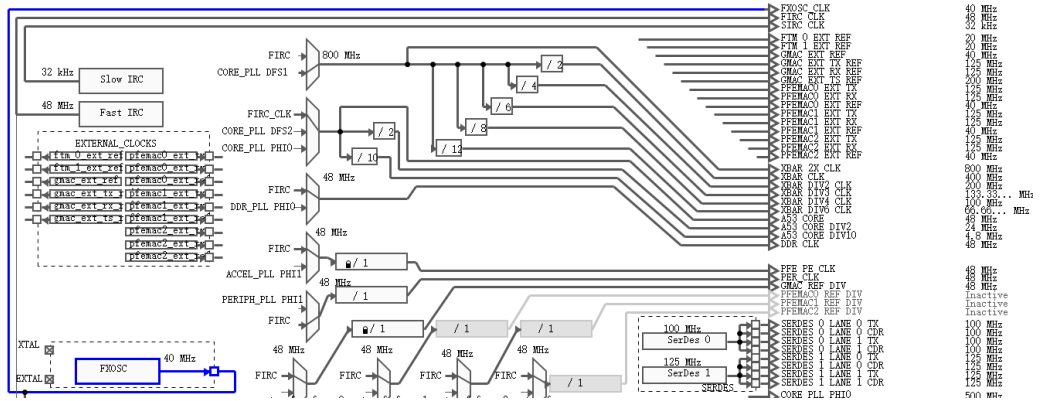


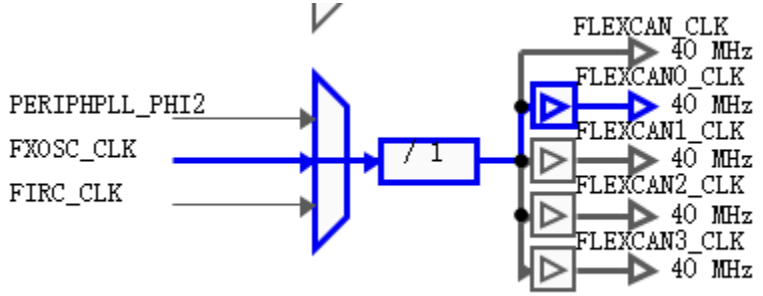
HANDS ON CAN: CLOCK CONFIGURATION 2

Open the **Peripheral Clock View**, double click the **FLEXCAN0_CLK**. The **Clocks Diagram** will show the power tree and the key node can be re-set. The default clock configuration of CAN is 40 MHZ. the CAN PE clock source comes from FXOSC

Ov... Pe... Co... Re... De... Cl...

Clock Name	Enable	Source	Divider	Freq
DMAMUX2_CLK	☑	XBA...	/ 1	133.
DMAMUX3_CLK	☑	XBA...	/ 1	133.
DMA_CRC0_CLK	☑	XBA...	/ 1	400
DMA_CRC1_CLK	☑	XBA...	/ 1	400
EIM0_CLK	☑	A53_...	/ 1	4.8
EIM1_CLK	☑	XBA...	/ 1	66.6
EIM2_CLK	☑	XBA...	/ 1	66.6
EIM3_CLK	☑	XBA...	/ 1	66.6
EIM_CLK	☑	XBA...	/ 1	66.6
ERM0_CLK	☑	XBA...	/ 1	66.6
FLEXCAN0_CLK	☑	FXO...	/ 1	40 M
FLEXCAN1_CLK	☑	FXO...	/ 1	40 M
FLEXCAN2_CLK	☑	FXO...	/ 1	40 M
FLEXCAN3_CLK	☑	FXO...	/ 1	40 M
FRAY0_CLK	☑	FIRC...	/ 1	48 M
FTIMER0_CLK	☑	FIRC...	/ 1	48 M
FTIMER1_CLK	☑	FIRC...	/ 1	48 M
GMAC0_RX_CLK	☑	FIRC...	/ 1	48 M

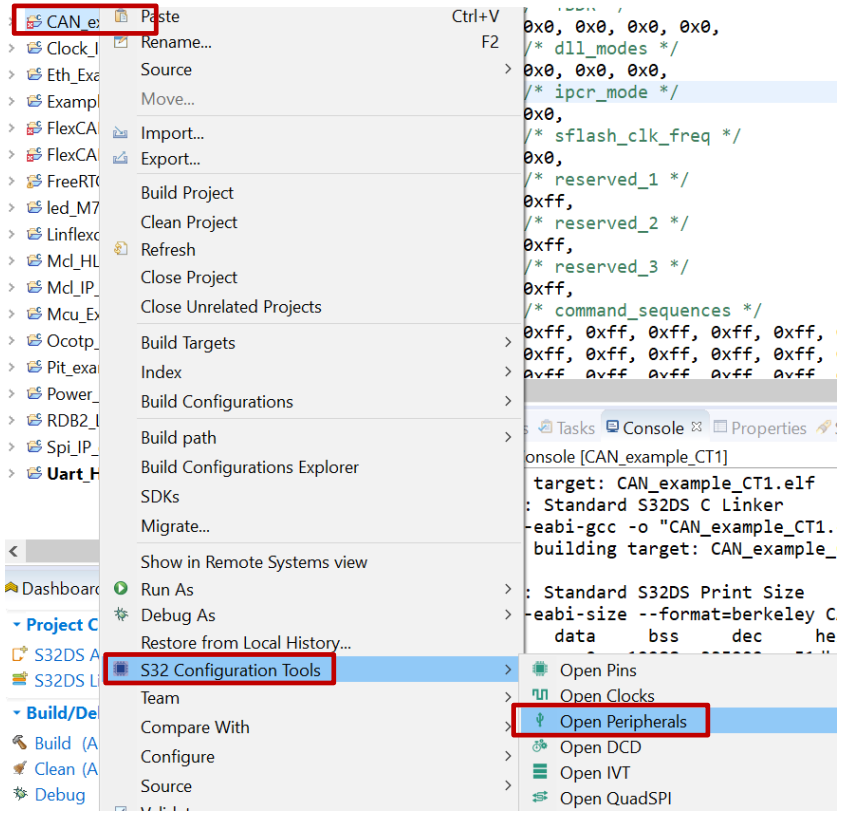


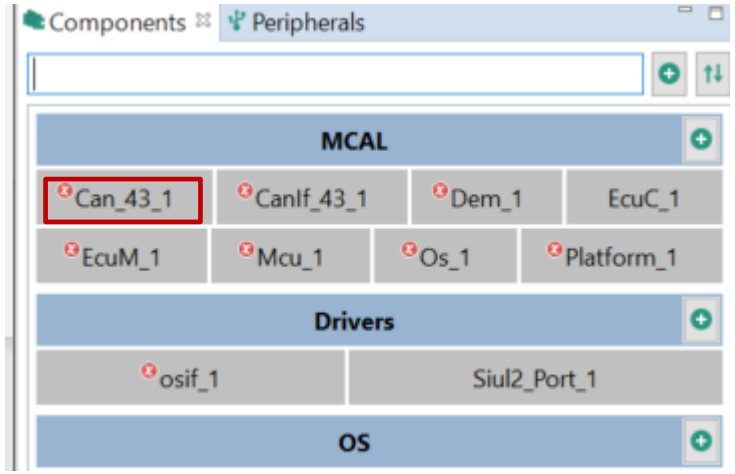


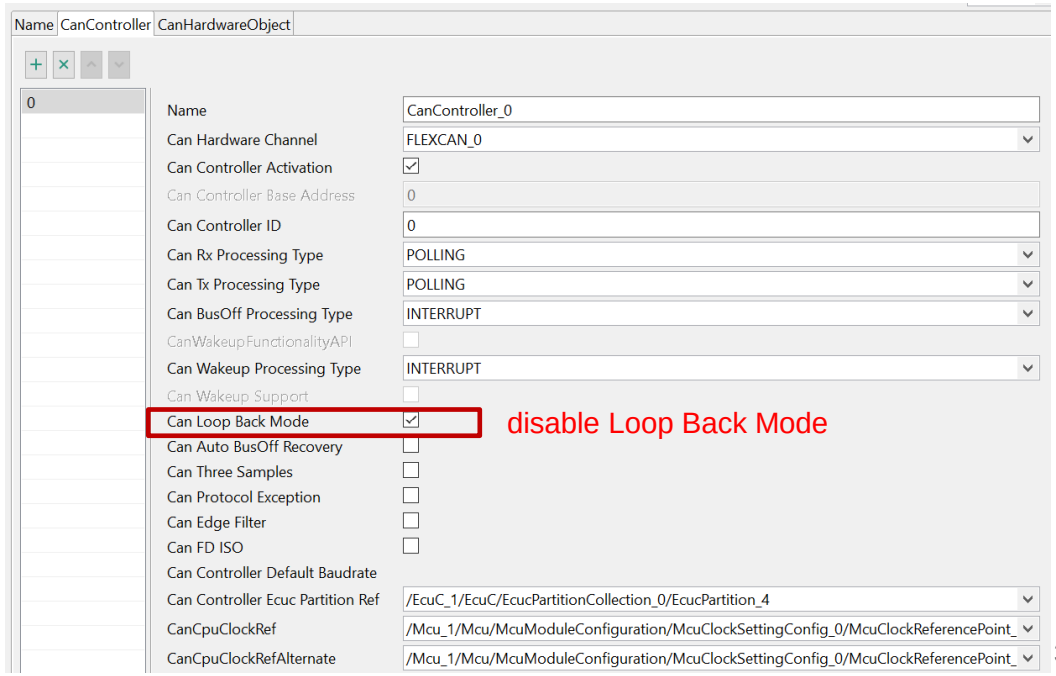
HANDS ON CAN: CAN CONFIGURATION 1

Open the Clocks Diagram:

- Right click on Project,
- Select S32 Configuration Tool…
- Select Peripherals



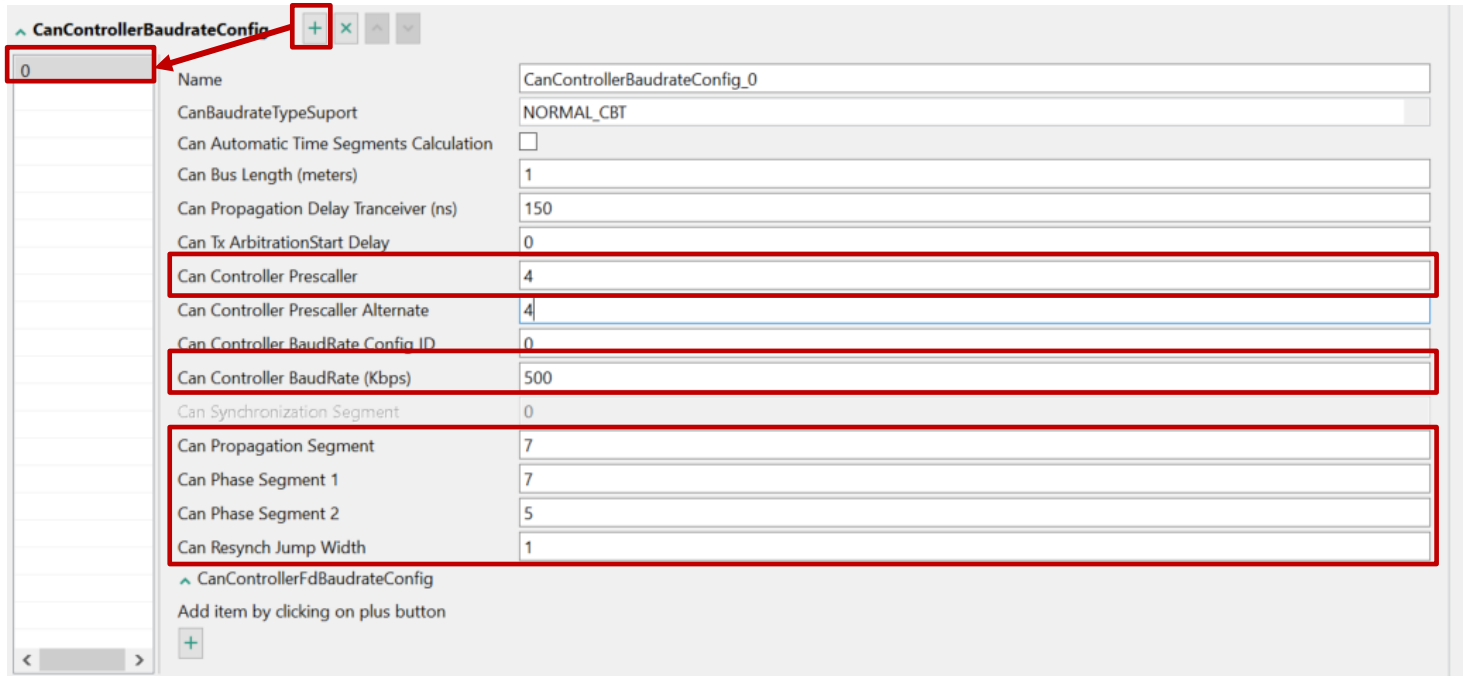




HANDS ON CAN: CAN CONFIGURATION 2

Configure the Baud rate as 500Kbps for Controller 0

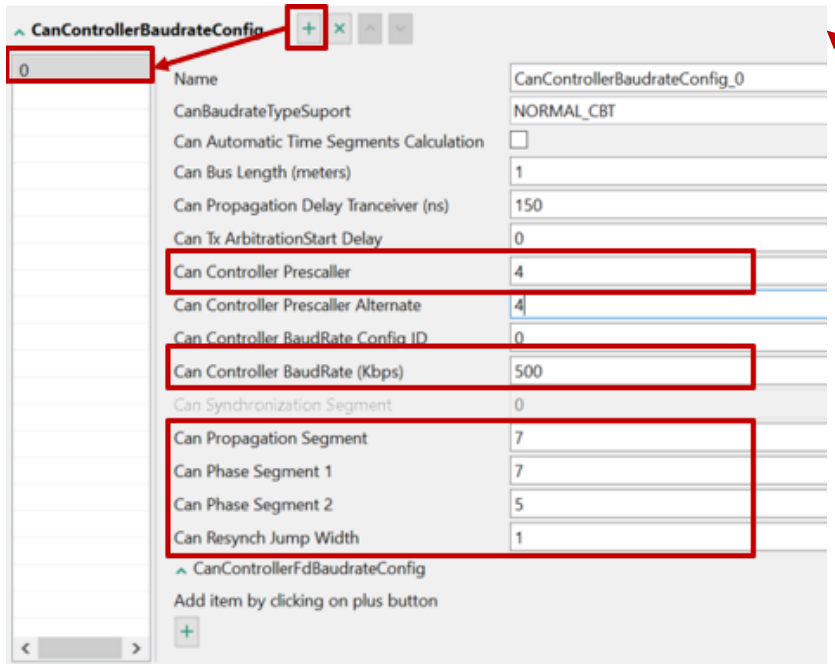
- TimeQuantum (seconds) = Prescaler / CanClockFrequency
- No. of CanTimeQuantas = (1 / CancontrollerBaudRate) / TimeQuantum
- No. of CanTimeQuantas = 1 + CanControllerPropSeg + CanControllerSeg1 + CanControllerSeg2



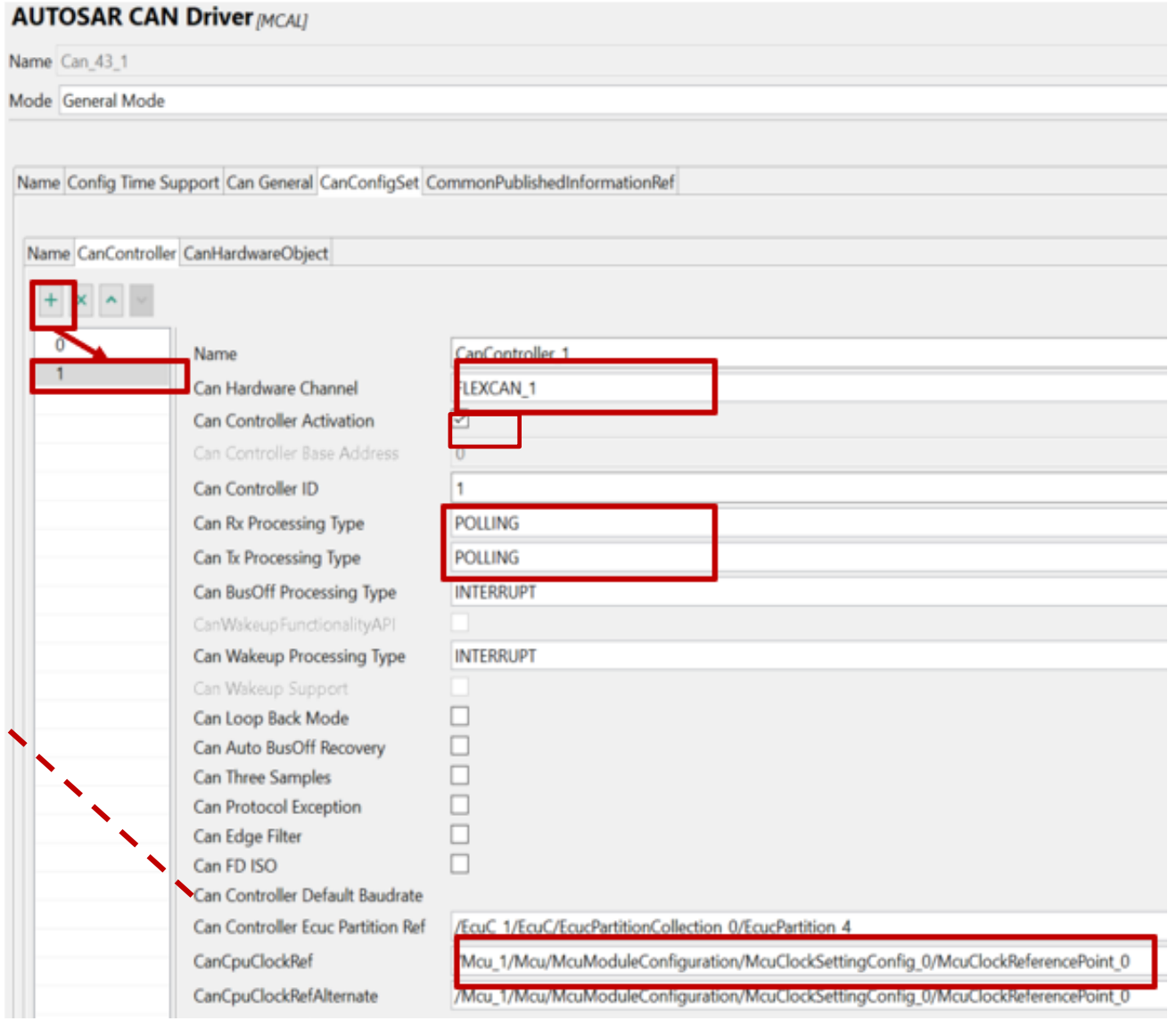
HANDS ON CAN: CAN CONFIGURATION 3

Open the peripheral configuration view

- add a new CanController for FLEXCAN_1
- Set Hardware Channel as FLEXCAN_1
- Set CAN Rx/TX Processing Type as POLLING
- Set CanCpuClockRef as 40Mhz
- Set Baudrate as 500kbps



Set Baudrate

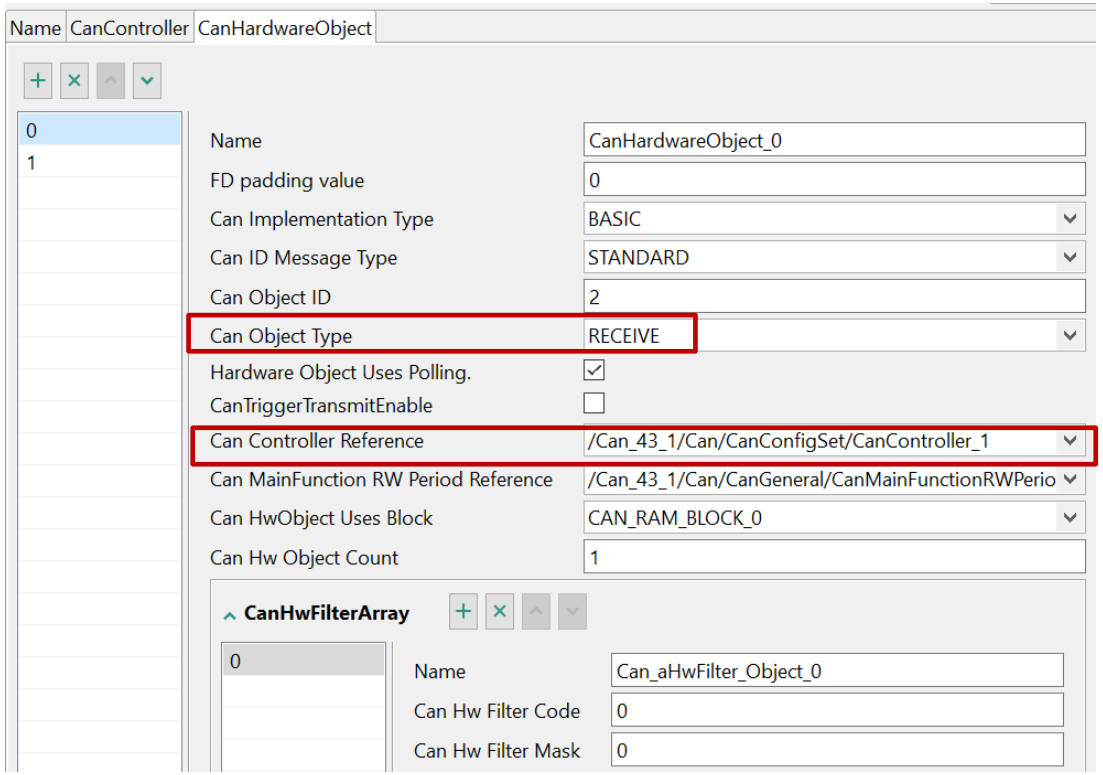


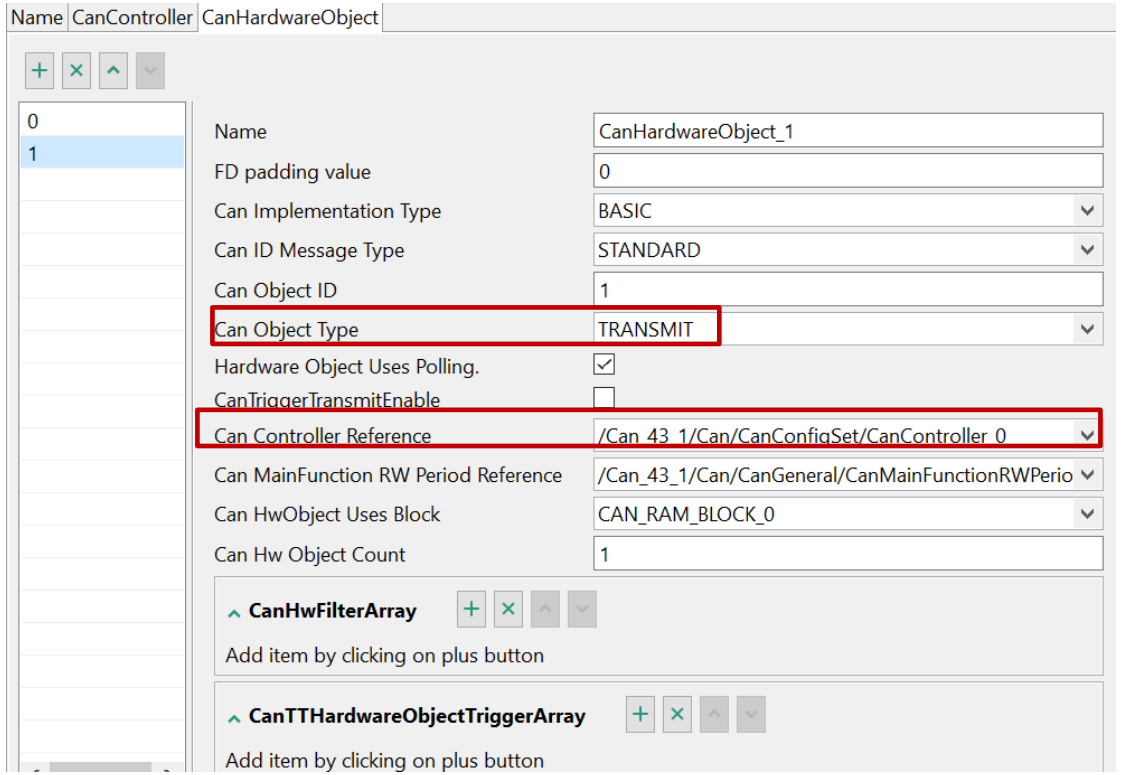
HANDS ON CAN: CAN CONFIGURATION 4

Modify the CanHardwareObjects Configuration for CanController 0 and CanController 1

Set the CanHardwareObjects_0 reference to CanController 1

Set the CanHardwareObjects_1 reference to CanController 0





HANDS ON ETH: UPDATE CODE

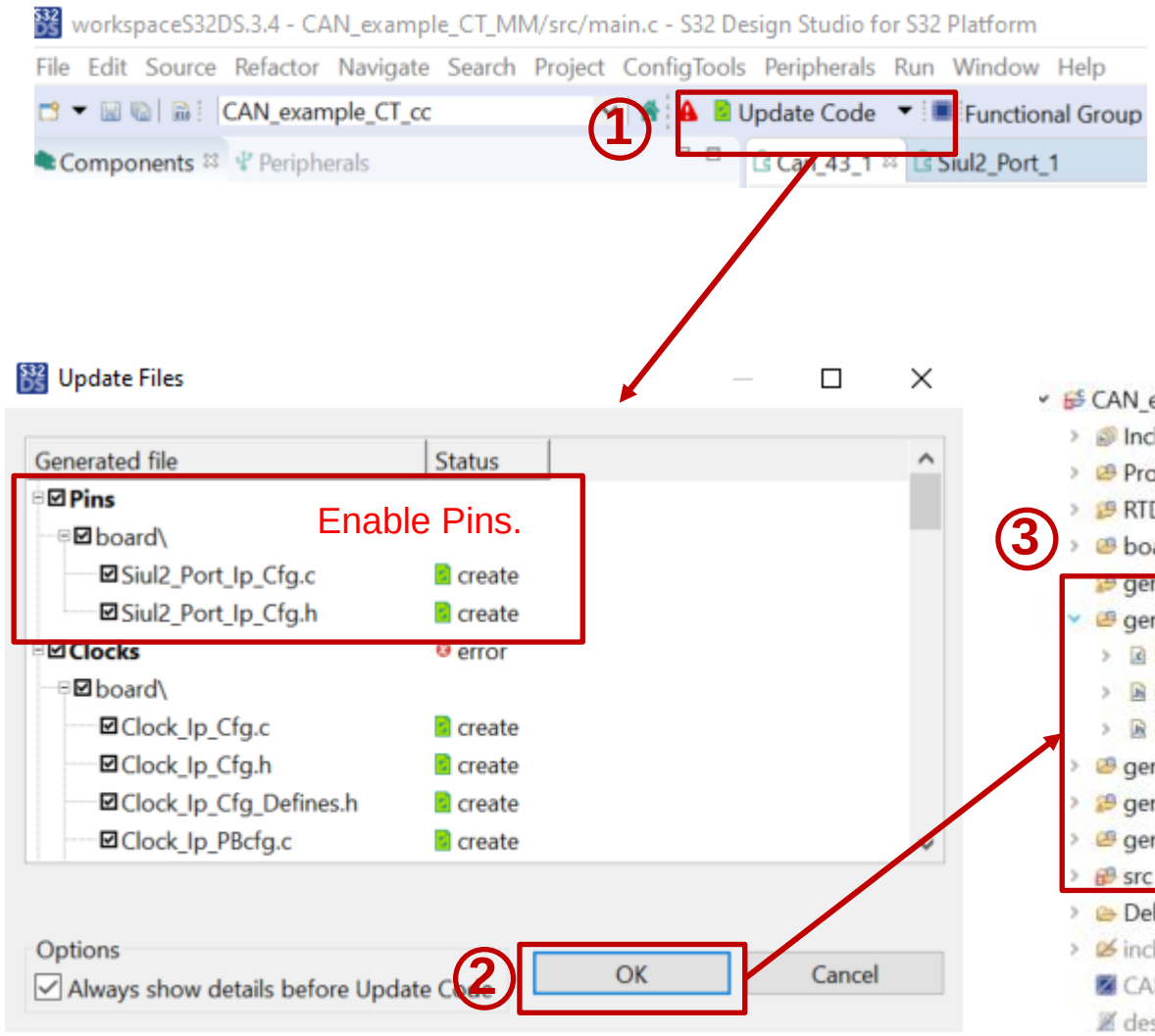
Generate code method:

1.Click on any configuration tool, like Pins

Then click **Update Code** (ensure desired project is selected!)

2.The Update Files widow pops up. It shows the detail update information. Click **ok** button.

3.The configuration .c and .h file will be generated at “generate” folder.



HANDS ON CAN: APPLICATION CODE

```
/*==================================================
*                                          INCLUDE FILES
* 1) system and project includes
* 2) needed interfaces from external units
* 3) internal and external interfaces from this unit
==================================================*/
#include "Mcu.h"
#include "Platform.h"
#include "Can.h"
#include "SchM_Can.h"
#include "check_example.h"

#include "Siul2_Port_Ip.h"
```

```
int main(void)
{
    uint8 u8TimeOut = 100U;
    CanIf_bTxFlag = FALSE;
    CanIf_bRxFlag = FALSE;
    /* Initialize the Mcu driver */
    Mcu_Init(NULL_PTR);

    /* Initialize the clock tree and apply PLL as system clock */
    Mcu_InitClock(McuClockSettingConfig_0);

    while ( MCU_PLL_LOCKED != Mcu_GetPllStatus() )
    {
        /* Busy wait until the System PLL is locked */
    }

    Mcu_DistributePllClock();
    Mcu_SetMode(McuModeSettingConf_0);
    /* Initialize Platform driver */
    Platform_Init(NULL_PTR);
    static Can_PduType Can_PduInfo;

    Siul2_Port_Ip_Init(NUM_OF_CONFIGURED_PINS0, g_pin_mux_InitConfigArr0);
    /* Can_CreatePduInfo(id, swPduHandle,length, sdu) */
```

Add the Port configuration and initiation function

Add the Can_SetControllerMode for CanController_1

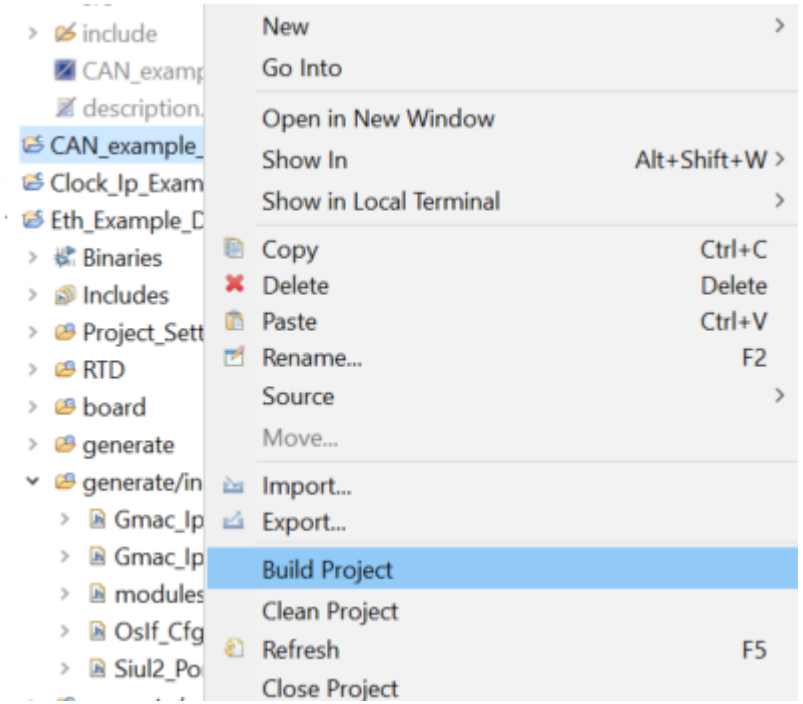
```
/* Can_CreatePduInfo(id, swPduHandle,length, sdu) */
Can_PduInfo = Can_CreatePduInfo(0U, 0U, 8U, Can_au8Sdu8bytes);
/* Initilize Can driver */
Can_Init(&Can_Config_VS_0);
Can_SetControllerMode(CanController_0, CAN_CS_STARTED);
Can_SetControllerMode(CanController_1, CAN_CS_STARTED);
if((Can_Write(CanHardwareObject_1, &Can_PduInfo) == E_OK))
while((!CanIf_bTxFlag) && (u8TimeOut != 0U))
{
    Can_MainFunction_Write();
    Can_DummyDelay(100U);
    u8TimeOut--;
}

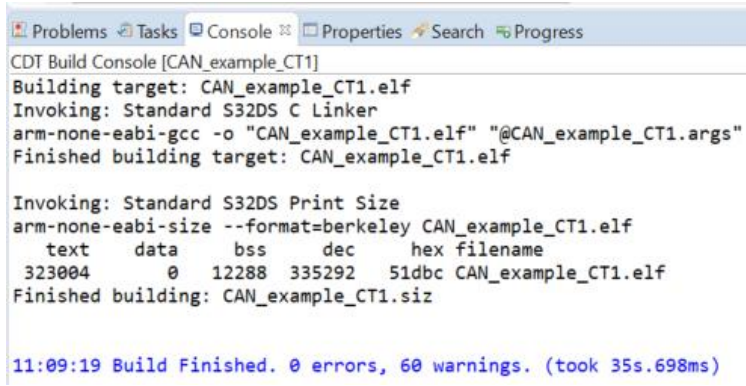
u8TimeOut = 100U;
while((!CanIf_bRxFlag) && (u8TimeOut != 0U))
{
    Can_MainFunction_Read();
    Can_DummyDelay(100U);
    u8TimeOut--;
}
Can_SetControllerMode(CanController_0, CAN_CS_STOPPED);
Can_SetControllerMode(CanController_1, CAN_CS_STOPPED);
Can_DeInit();
```

HANDS ON CAN: BUILD AND DEBUG 1

Build target Project:

- Right click on Project
- Build Project
- The console print build information
- CAN_example_CT1.elf is created



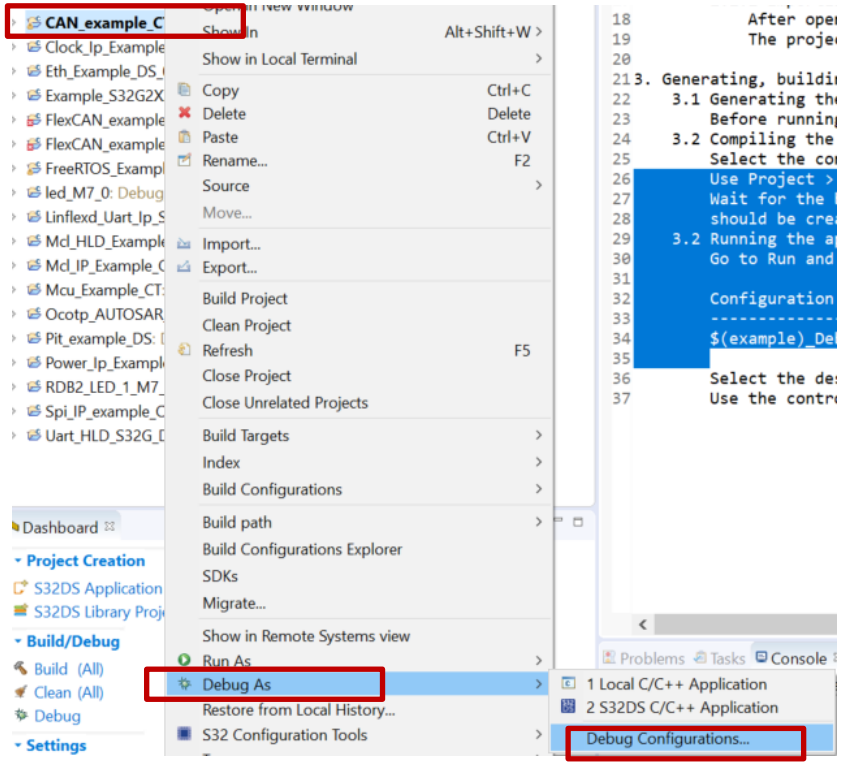




HANDS ON CAN: BUILD AND DEBUG 2

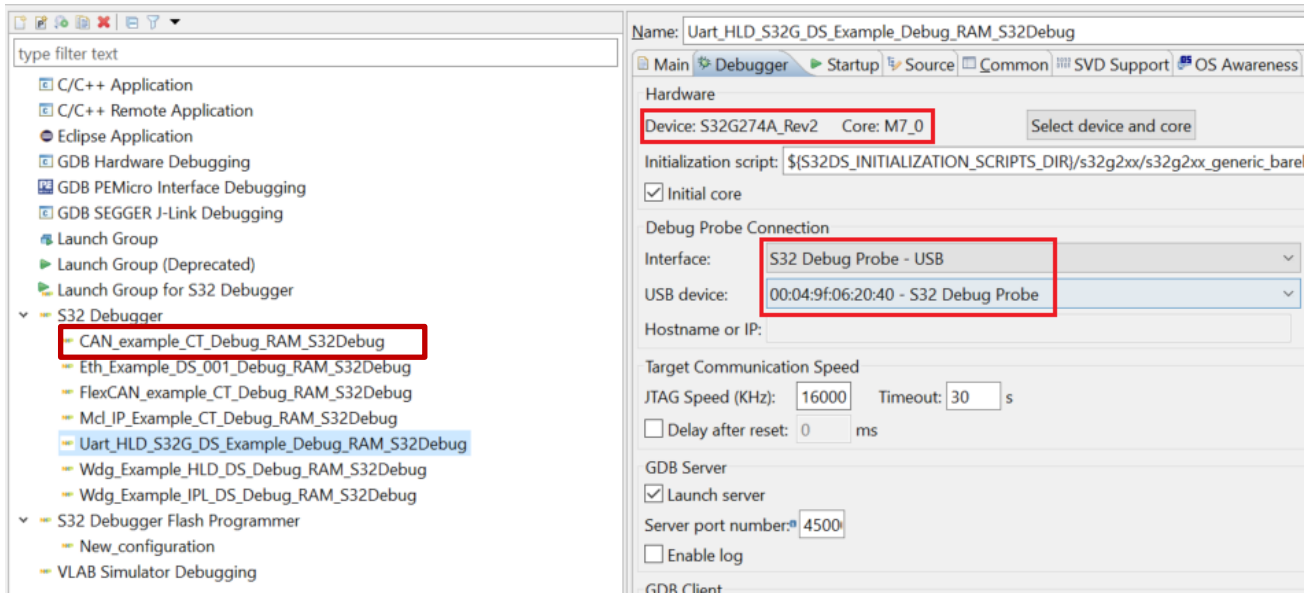
Go to debug configuration:

- Right click on Project,
- Select the Debug As
- Click Configurations



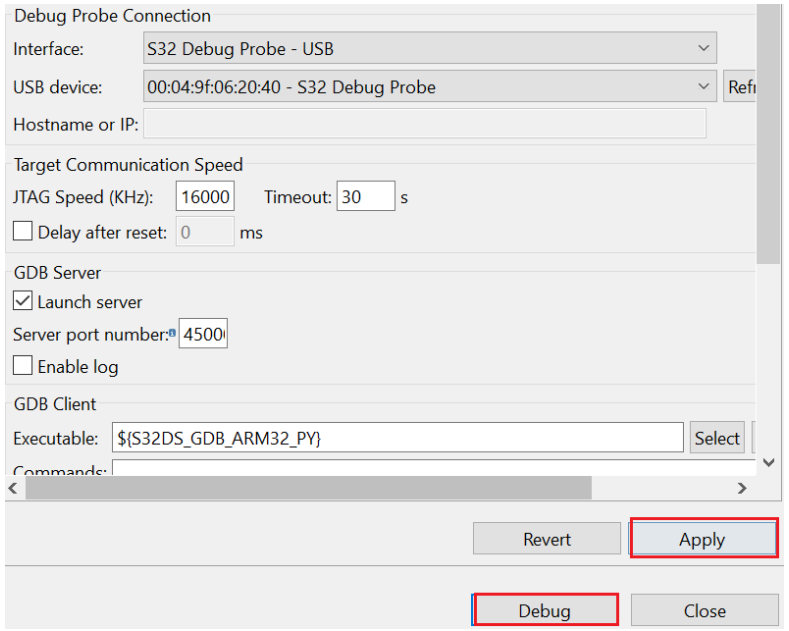
Debug configuration set:

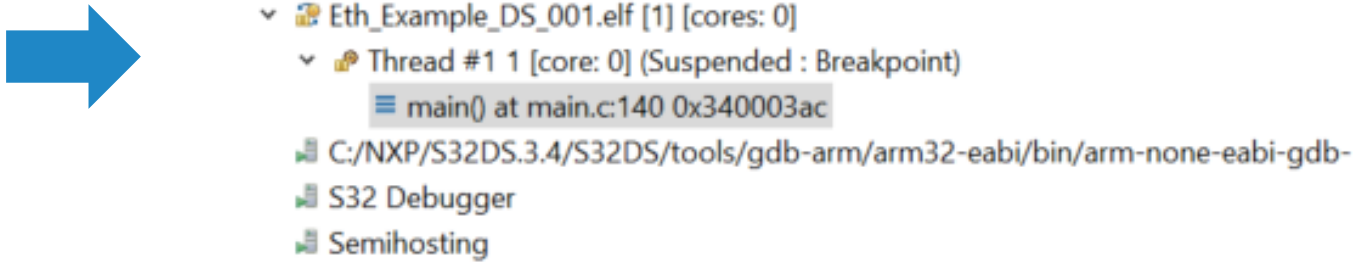
- Click target project ,
- Select the target board
- Select target debugger



HANDS ON CAN: DEBUG AND RUN

Click on “Apply”, then click on “Debug”. the perspective will jump to the Debug Perspective, and you can use the controls to control the program flow.



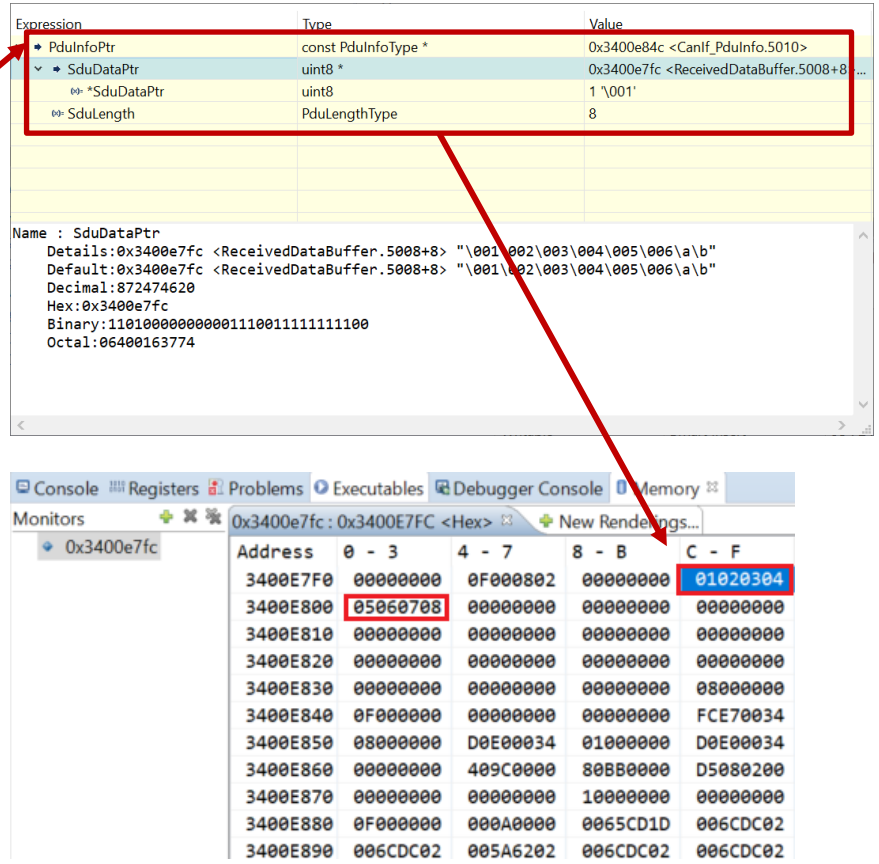


HANDS ON CAN: TEST RESULT

Through the modification, the CAN frame transmits from CAN0 to CAN1. the callback function **`CanIf_RxIndication`** capture the received CAN frame.

```
void CanIf_TxConfirmation(PduIdType CanTxPduId)
{
    CanIf_u8TxConfirmCnt++;
    CanIf_bTxFlag = TRUE;
}

void CanIf_RxIndication(const Can_HwType* Mailbox,
const PduInfoType* PduInfoPtr )
{
    CanIf_u8RxIndicationCnt++;
    CanIf_bRxFlag = TRUE;
}
```





SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V. ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.