How to use printf() to print string to Console and UART in KDS

By : Alice Yang

This document summarizes how to use printf() to print string to Console and UART in KDS .

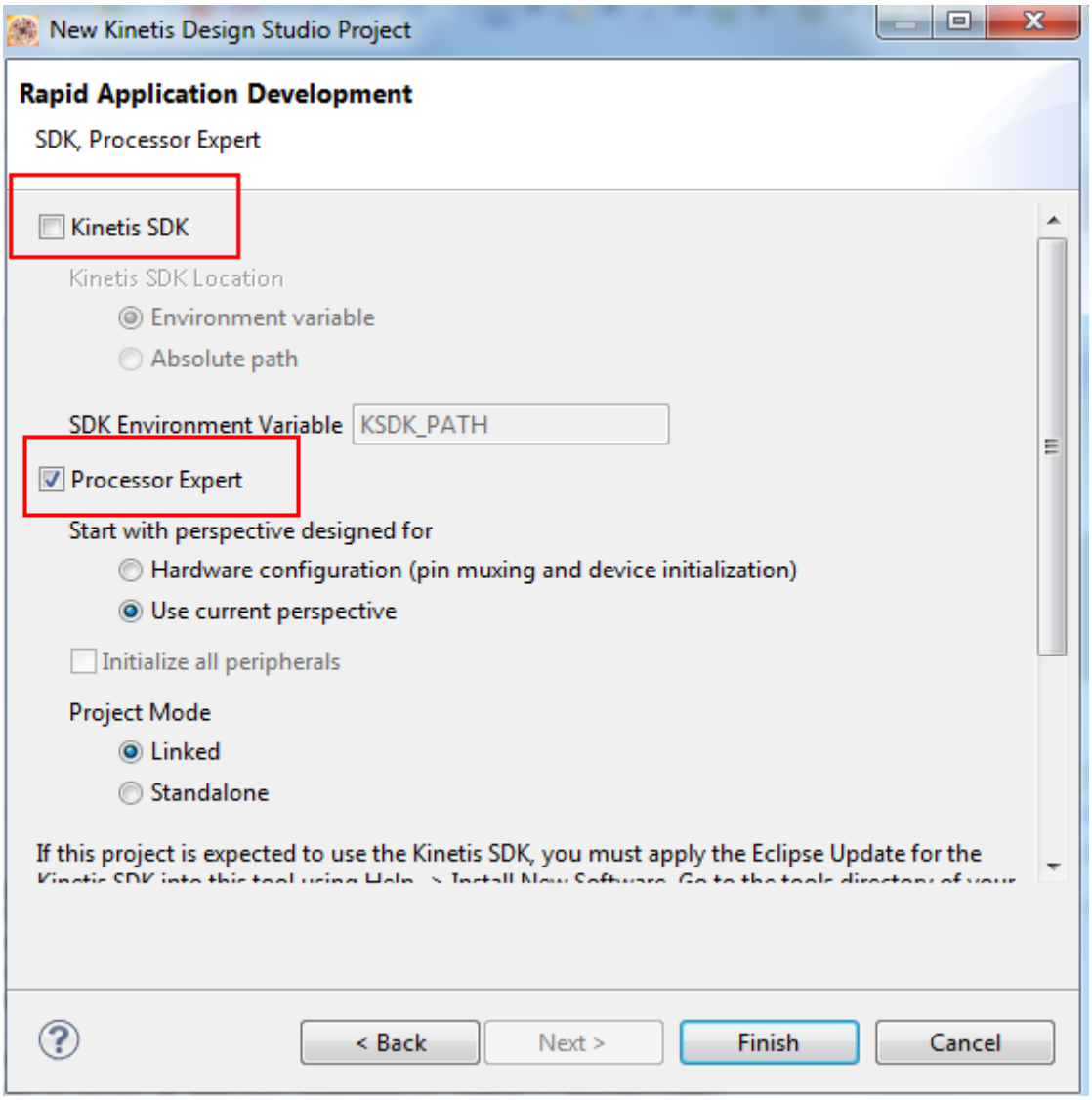
Printing string to Console and UART both includes three conditions: Processor Expert project, KSDK project and bare board project.

1. Use printf() to print string to Console

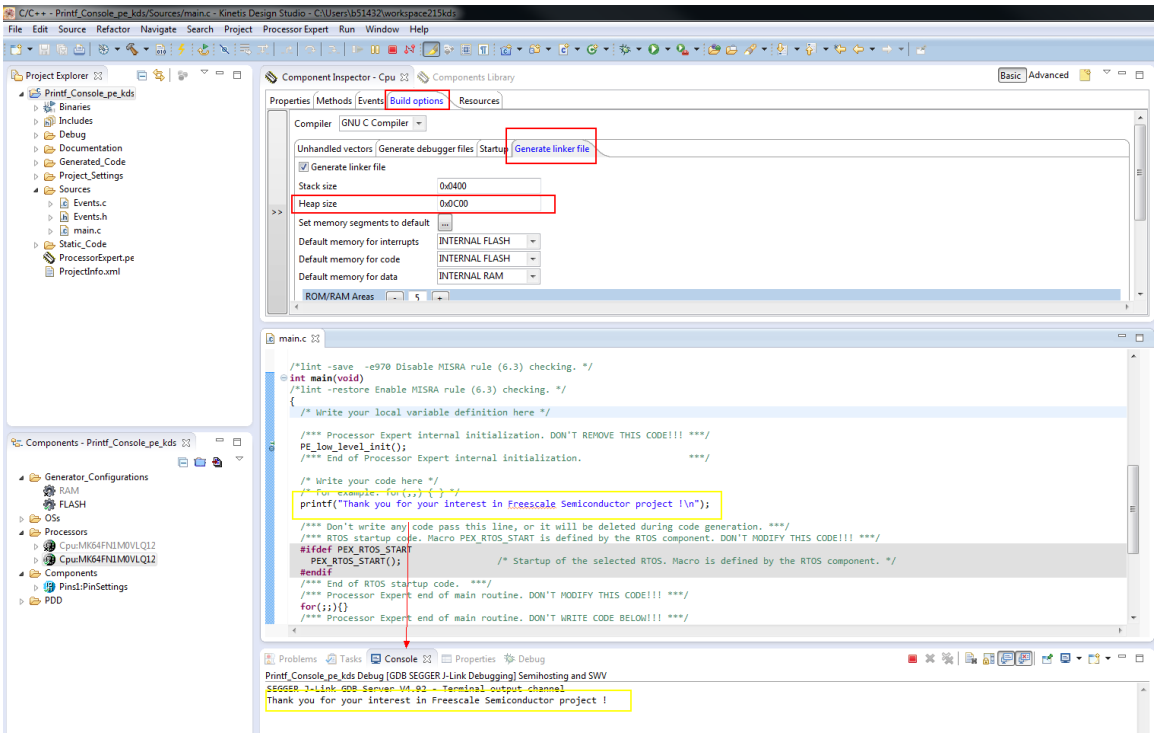
It includes three conditions: Processor Expert project, KSDK project and bare board project.

1.1 Use printf() in Processor Expert Project

- Create a Processor Expert project: check the “Processor Expert” selection, uncheck “Kinetis SDK” selection, and keep other configurations as default.



- printf() is memory consuming, so in project, we need make sure have enough stack and heap allocated. In the Processor Expert project, we need increase the stack and heap size in the CPU component “Build options” tab, here I set the “Heap size” to 0x0C00, keep the “Stack size” default 0x400.
- Build, launch the debugger, the result will be printed on the “Console” window.



1.2 Use printf() in KSDK Project

- Create a KSDK project: check “Kinetis SDK” and “Processor Expert” selection, and keep other configurations as default.
- The default size of stack and heap is 0x400.Generally, it can meet our requirement. If it can’t, the method of increasing it is the same as PE project.

1.3 Use printf() in Bare Board Project

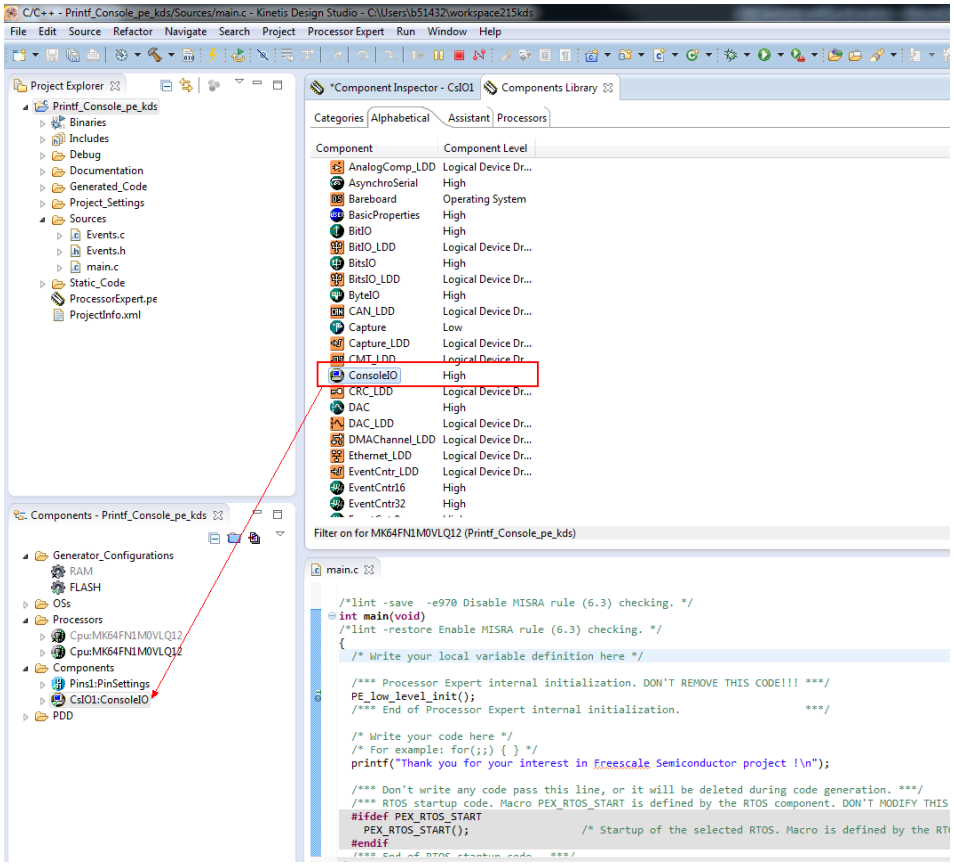
Create a bare board project: uncheck the “Kinetis SDK” and “Processor Expert” selection, keep other configurations as default. From the linker file we can see that the default size of stack and heap is 0x400. Generally, it can meet our requirement. If it can’t, we can increase it in the linker file.

2. Use printf() to print string to UART

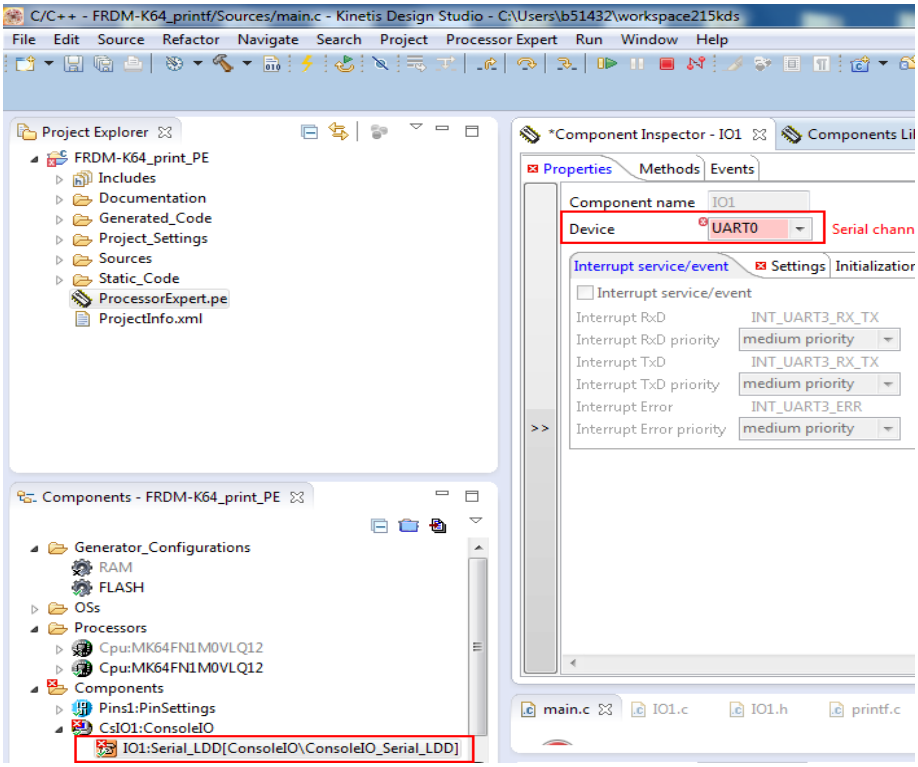
It also includes three conditions: Processor Expert project, KSDK project and bare board project.

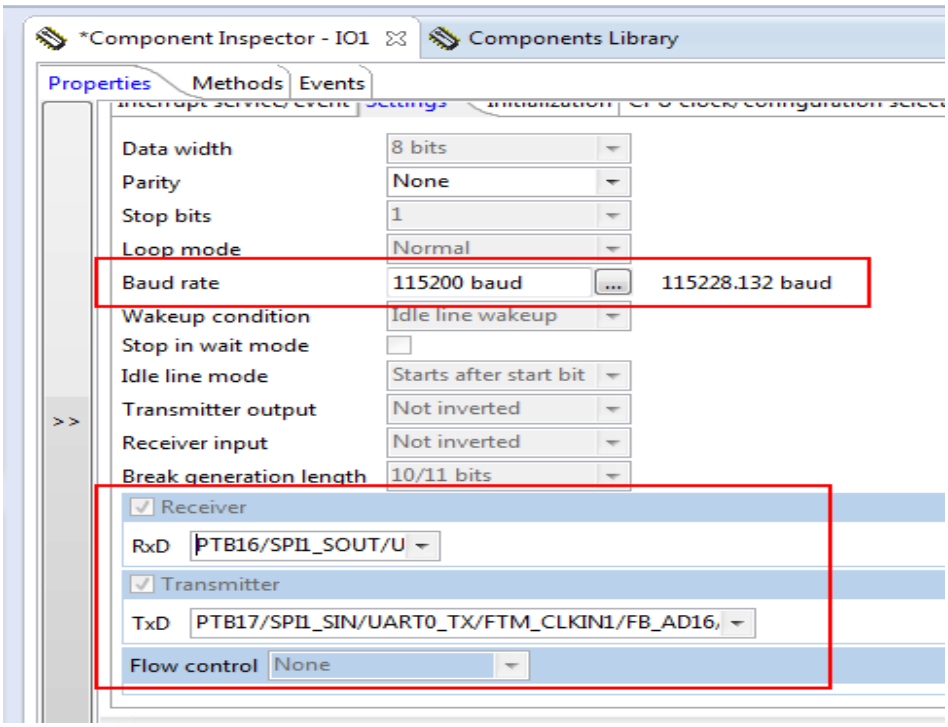
2.1 Use printf() in Processer Expert Project

- Create a Processer Expert project like 1.1.
- Add “ConsoleIO” component.

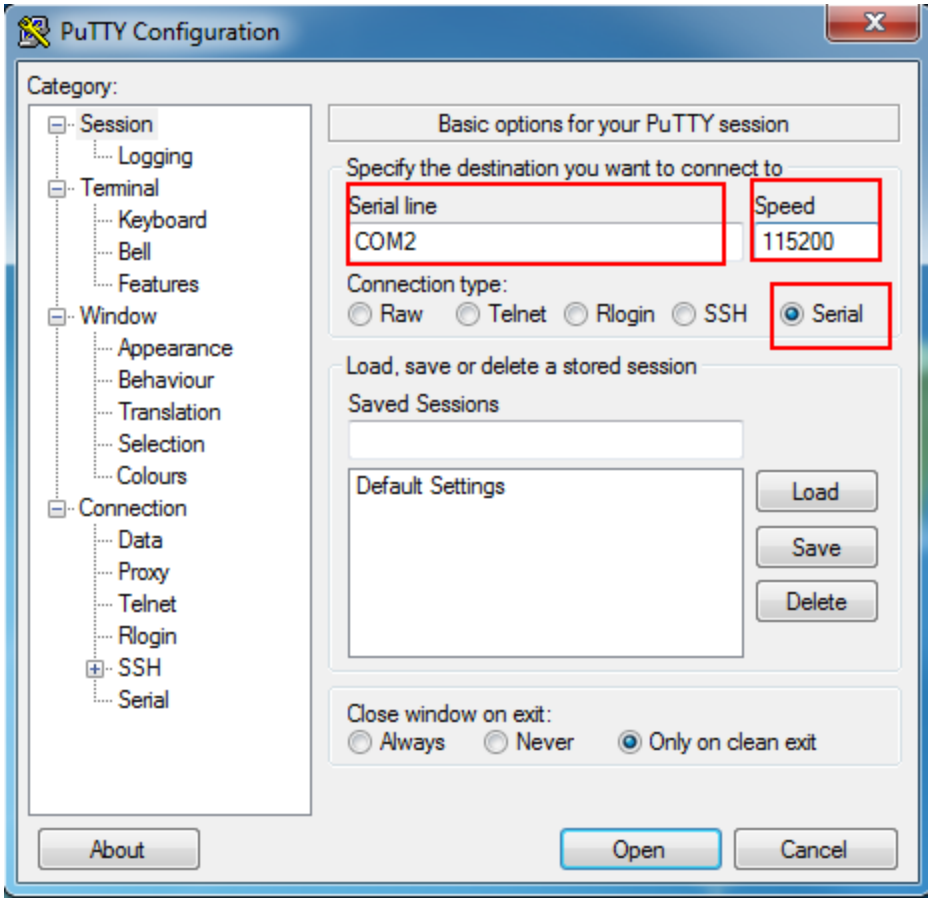


- Configure the UART: Device, Baud rate and pins. (For example FRDM-K64 board)

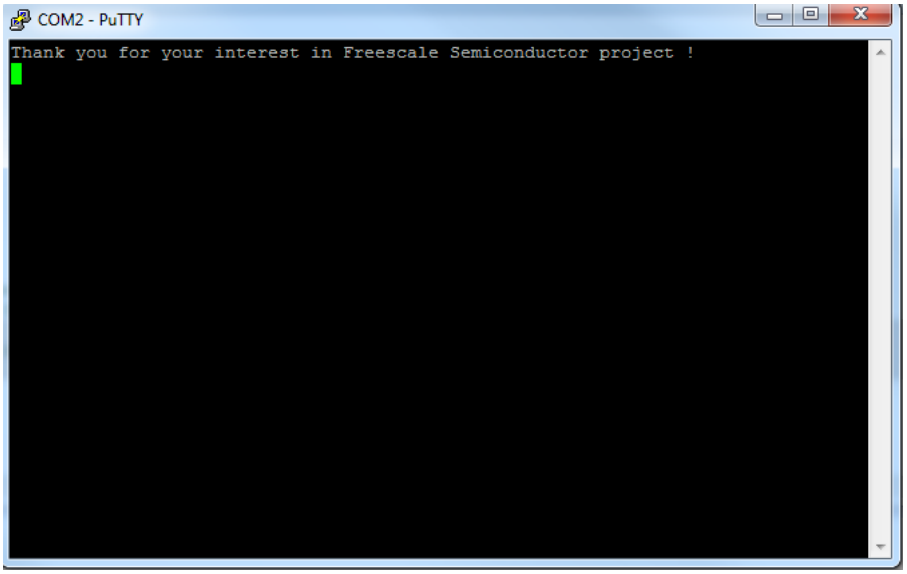




- Configure the PuTTY: 1152000 baud, No parity, 8 data bits, 1 stop bit.

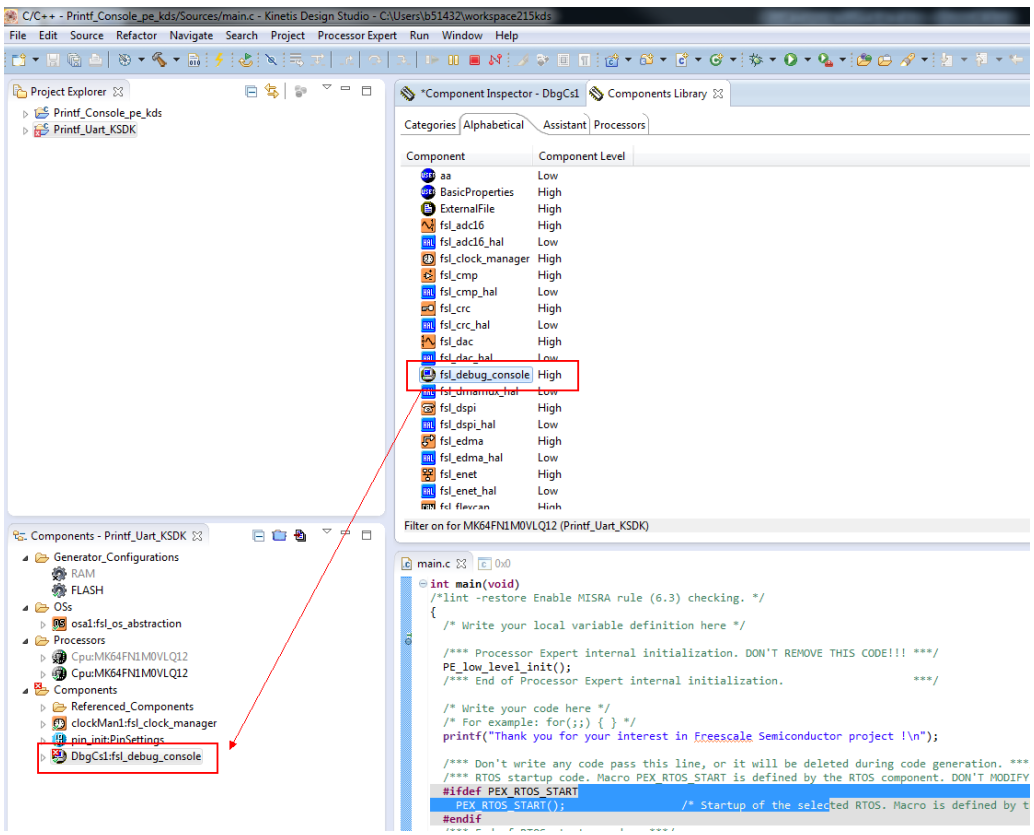


- Run the project, we will see the result on the PuTTY.

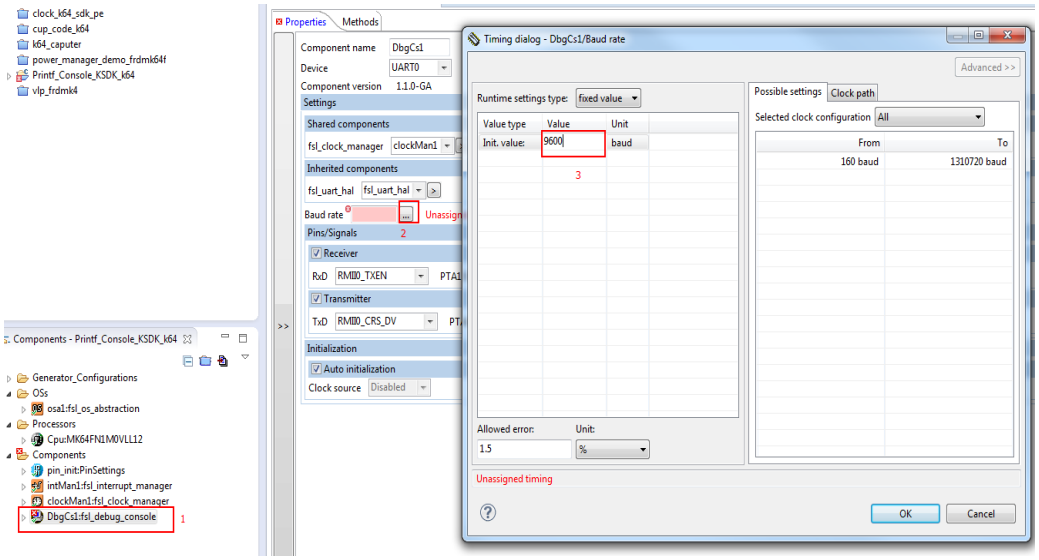


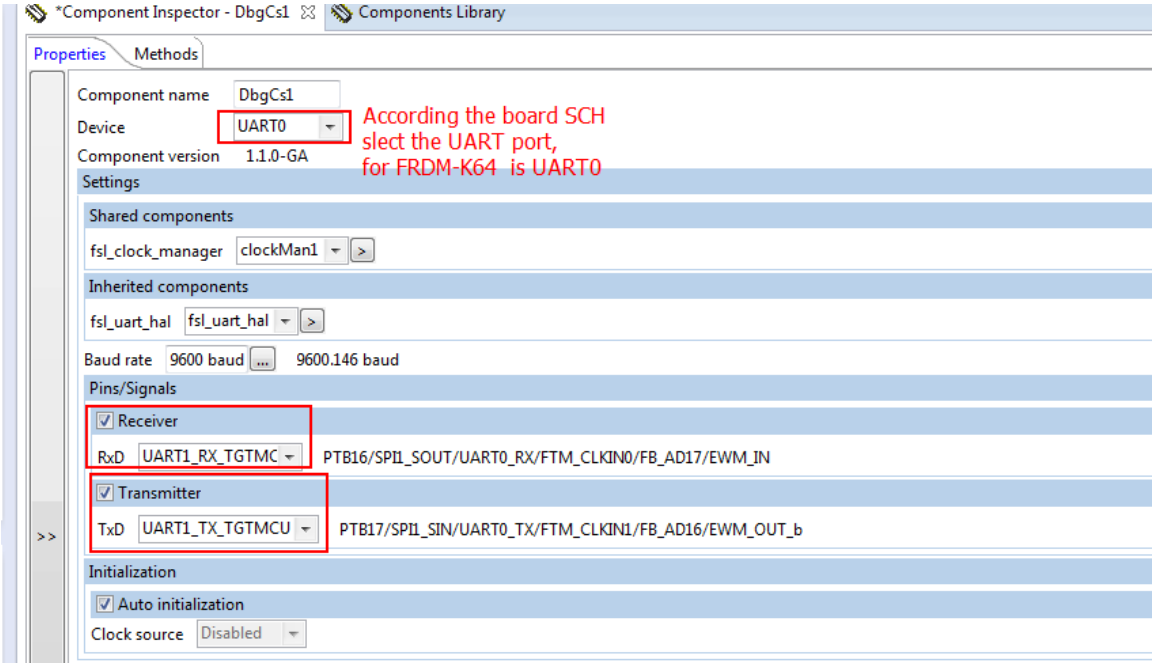
2.2 Use printf() in KSDK Project

- Create a KSDK project as 1.2.
- Add “fsl_debug_console ”component.



- Configure the UART: Device, Baud rate and pins. (for example FRDM-K64 board)





- The other steps are the same as PE project.

2.3 Use printf() in Bare Board Project

- Create a Bare Board project as 1.3.
- Initialize UART: enable UART and related PORT clock, configure UART pin and baud rate. For FRDM-K64 board, use UART0 , PTB16 and PTB17 pin:

```
/* SIM_SCGC4: UART0=1, enable UART0 clock */
SIM_SCGC4 |= SIM_SCGC4_UART0_MASK;
/* SIM_SCGC5: PORTB=1, enable PORTB clock */
SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;

  /*configure UART TX and RX pin */
PORTB_PCR16 = (uint32_t)(PORTB_PCR16 | (uint32_t)(
            PORT_PCR_MUX(0x03) ));
PORTB_PCR17 = (uint32_t)(PORTB_PCR16 | (uint32_t)(
            PORT_PCR_MUX(0x03) ));

/* Set baud rate fine adjust */
UART_PDD_SetBaudRateFineAdjust(UART0_BASE_PTR, 12u);
/* Set the baud rate register. */
UART_PDD_SetBaudRate(UART0_BASE_PTR, 11U);
```

-Write "_write()" function. "_write()" function is used to transmit character to terminal, in FRDM-k64 board , UART0 is used to transmit. The core code looks like:

```
#define UART_PDD_PutChar8(PeripheralBase, Char) ( \
UART_D_REG(PeripheralBase) = \
(uint8_t)(Char) \
)
/* Save a character into the transmit buffer of the UART0 device */
UART_PDD_PutChar8(UART0_BASE_PTR, (unsignedchar)*(uint8_t*)buf);
```

(About use printf() to print string to UART about bare board project , please refer to the demo "FRDM-K64_printf_UART_bareboad")

Reference:

(1) http://mcuoneclipse.com/2014/06/06/semihostring-with-kinetis-design-studio/
(2) KL26 Sub-Family Reference Manual
(3) FRDM-KL26Z_SCH_REV_A.pdf
(4) Kinetis SDK v.1.1 API Reference Manual