Problem Analysis and solutions for booting from ROM BOOTLOADER in KL series

1 Abstract

When customer use the kinetis chip KL43, KL27 and KL17 which flash size is above 128K, they have found a problem that if the code boot from the ROM instead of the flash, the application code about the LPUART and I2C will run in abnormal state, especially when use PTA1 as the LPUART receive pin, UART transmit function has no problem, but when the PTA1 receive the UART data, the code will run to the abnormal area and can't return back, the code will be crash. This problem only happens on booting from the ROM and the uart and i2c peripheral are enabled in BCA 0x3d0 address, uart peripheral enablement in BCA area will influence the application PTA1 uart receive, i2c peripheral enablement in BCA area will influence the i2c0 module in the application code. If booting from the flash or booting from ROM but the uart and I2C peripheral are disabled in the BCA 0x3d0 address, everything is working ok in the application code.

This document will take the UART problem as an example, give details of the problem reproduction, testing, analysis and the solutions. The I2C problem is the same when booting from the ROM bootloader.

2 Problem reproduction and analysis

Testing preparation:

- IDE: KDS 和 IAR
- Hardware: FRDM-KL43
- Software: KSDK1.3.0 and KSDK2.0_FRDM-KL43

We mainly reproduce the uart receive problem in two ways: new KDS PE project based on KSDK1.3.0 and official newest sample code package KSDK2.0_FRDM-KL43.

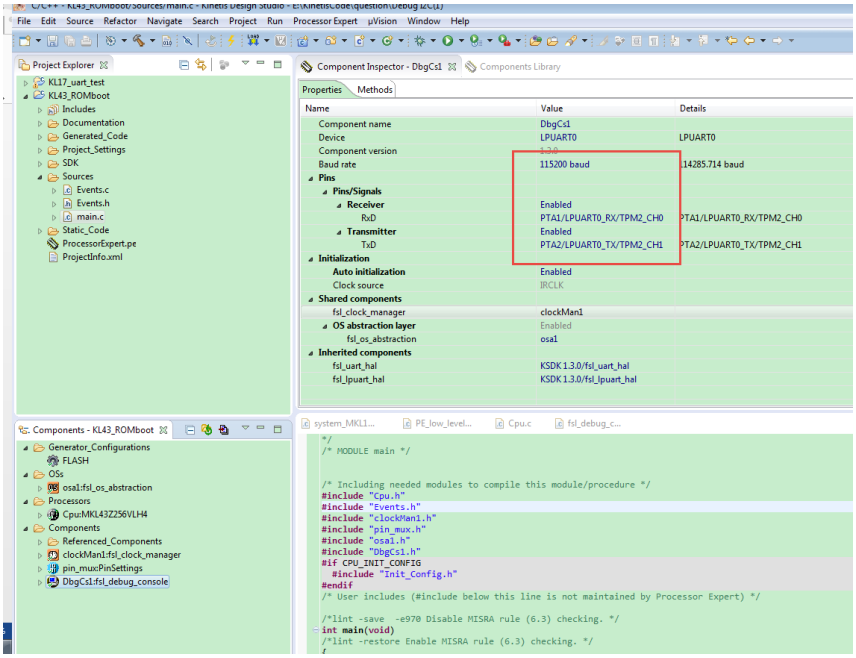
2.1 Problem reproduction in new creating kds project

Because the KSDK2.0 still doesn't support the PE function in the KDS IDE, so we use the KSDK1.3.0 as the PE KSDK to create the new KDS project.

2.1.1 Create KDS KL43 project

The new KDS PE project creating is very simple, here just describe the important points which is relate to the UART problem after booting from the ROM. At first create a new KDS PE project which is based on KSDK1.3.0, and choose the chip as MKL43Z256VLH4, select the MCG mode as HIRC, and configure core clock to 48Mhz, bus clock to 24Mhz.

Then add the uart module fls_debug_console for testing, because the FRDM_KL43 is using PTA1 and PTA2, the console module can be configured like the following picture, after the module is configured, press the code generation button to generate the project code.



Then add the simple code in file main.c main function for testing:

```
char a;
for(;;)
{
                PRINTF(" test!\n");
                a= GETCHAR();
                PUTCHAR(a);
              }
```

The code function is: printf the “test!” to the COM port in the PC, then wait the uart data, if receive the data, then printf the received data back and run this loop function again.

2.1.2 Add the BCA area

From the KL43 reference manual, we can get that, BCA start address is 0X3C0:

13.2.2 The Kinetis Bootloader Configuration Area (BCA)

The Kinetis Bootloader reads data from the Bootloader Configuration Area (BCA) to configure various features of the bootloader. The BCA resides in flash memory at offset 0x3C0, and provides all of the parameters needed to configure the Kinetis Bootloader operation. For uninitialized flash, the Kinetis Bootloader uses a predefined default configuration. A host application can use the Kinetis Bootloader to program the BCA for use during subsequent initializations of the bootloader.

Table 13-2. Configuration Fields for the Kinetis Bootloader

Offset	Size (bytes)	Configuration Field	Description
0x00 - 0x03	4	tag	Magic number to verify bootloader configuration is valid. Must be set to 'kcfg'.
0x04 - 0x07	4	-	Reserved in KLx3
0x08 - 0x0B	4	-	Reserved in KLx3
0x0C - 0x0F	4	-	Reserved in KLx3
0x10	1	enabledPeripherals	Bitfield of peripherals to enable. bit 0 LPUART bit 1 I2C bit 2 SPI bit 4 USB
0x11	1	i2cSlaveAddress	If not 0xFF, used as the 7-bit I2C slave address.
0x12 - 0x13	2	peripheralDetectionTimeout	Timeout in milliseconds for active peripheral detection

The KDS newly created project didn't contain the BCA area in the link file, so we need to add this area in the link file and add the BCA data in the start file by ourselves.

2.1.2.1 Divide the BCA flash are in .ld file

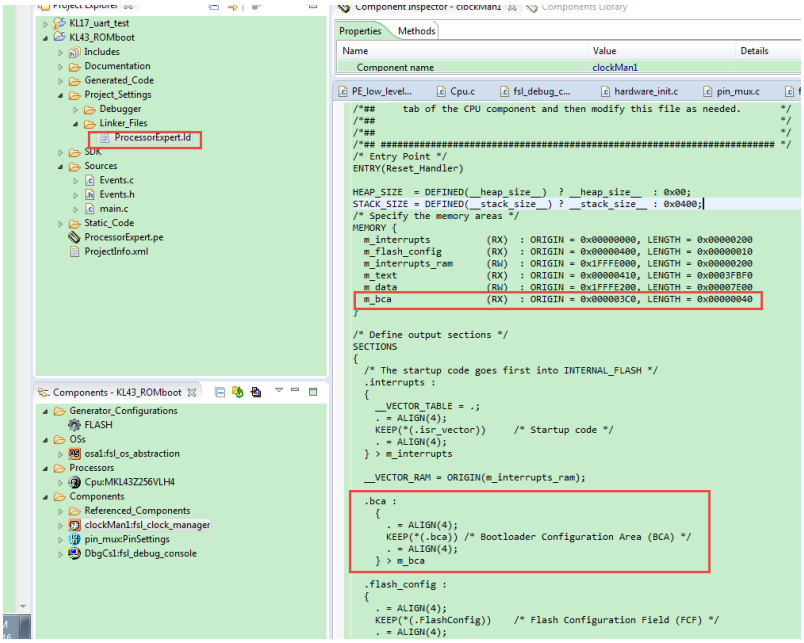
Add the following code to define the BCA start flash address and the flash size in the ProcessorExpert.ld memory area:

```
m_bca            (RX)  : ORIGIN = 0x000003C0, LENGTH = 0x00000040
```

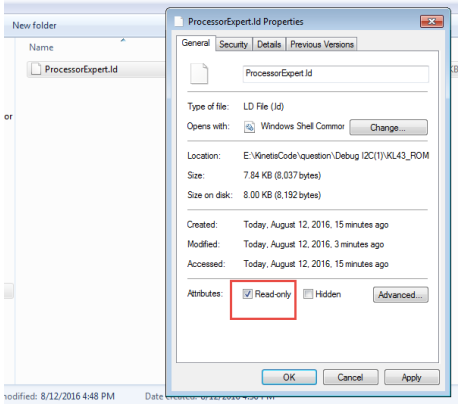
Then add this code in the SECTIONS area:

```
 .bca :
       {
        . = ALIGN(4);
        KEEP(*(.bca)) /* Bootloader Configuration Area (BCA) */
        . = ALIGN(4);
       } > m_bca
```

At last, the ld file is like this:



For the ld file protection, we can change the ld file properties to read-only, then this file won't be changed to the initial one after building.

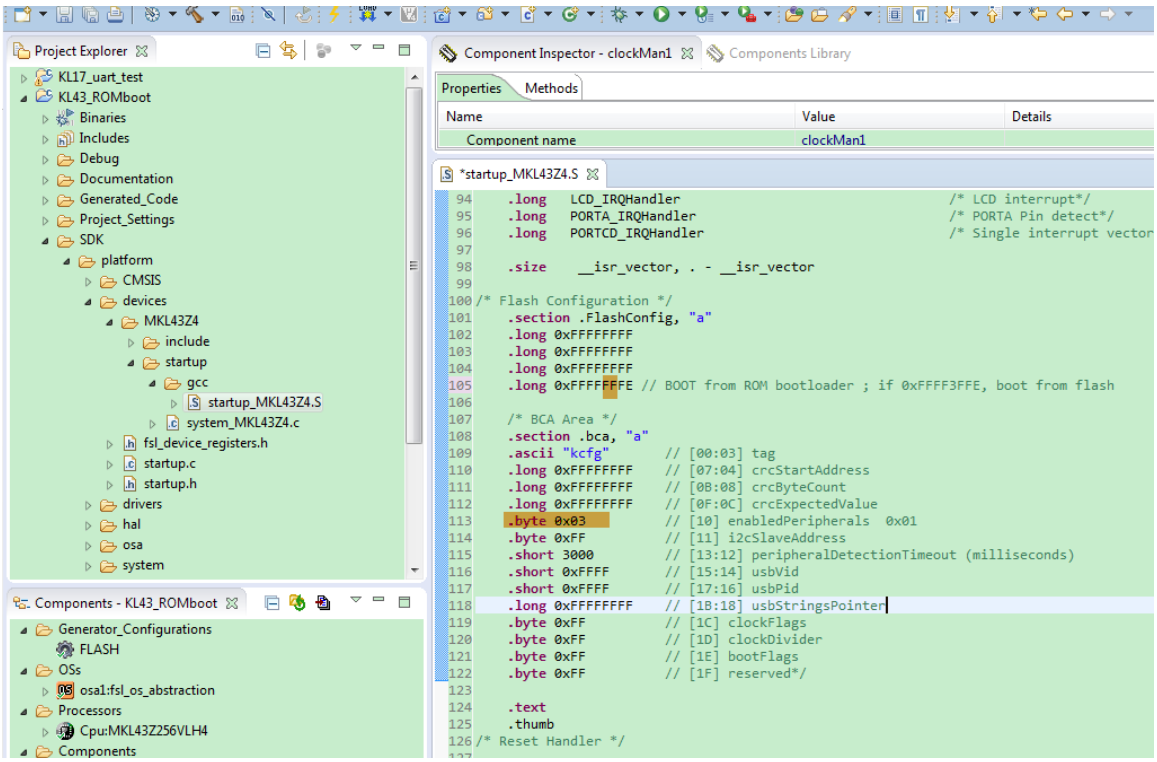


2.1.2.2 Add the BCA data in the start file

After add the BCA flash area divide code, we still need to define the BCA data in the start file:

```
/* BCA Area */
 .section .bca, "a"
        .ascii "kcfg"                 // [00:03] tag
        .long 0xFFFFFFFF  // [07:04] crcStartAddress
        .long 0xFFFFFFFF  // [0B:08] crcByteCount
        .long 0xFFFFFFFF  // [0F:0C] crcExpectedValue
        .byte 0x03                    // [10] enabledPeripherals  I2C and UART
        .byte 0xFF                    // [11] i2cSlaveAddress
        .short 3000                   // [13:12] peripheralDetectionTimeout (milliseconds)
        .short 0xFFFF                 // [15:14] usbVid
        .short 0xFFFF                 // [17:16] usbPid
        .long 0xFFFFFFFF  // [1B:18] usbStringsPointer
        .byte 0xFF                    // [1C] clockFlags
        .byte 0xFF                    // [1D] clockDivider
        .byte 0xFF                    // [1E] bootFlags
        .byte 0xFF                    // [1F] reserved*/
```

More details, please refer to this picture:



So far, we have create the FRDM-KL43 test project which contains the BCA area, and boot from the ROM that can be modified in the flash address 0X40D, bit 6-7 in 0X40D is the BOOTSRC_SEL bits, 00 boot from flash, 10 and 11 boot from ROM, more details about the FOPT, please refer to Table 6-2. Flash Option Register (FTFA_FOPT) definition in reference manual.

2.1.3 Test result and analysis

Now, list the test result after booting from ROM or flash, and boot from ROM but enable the peripherals.

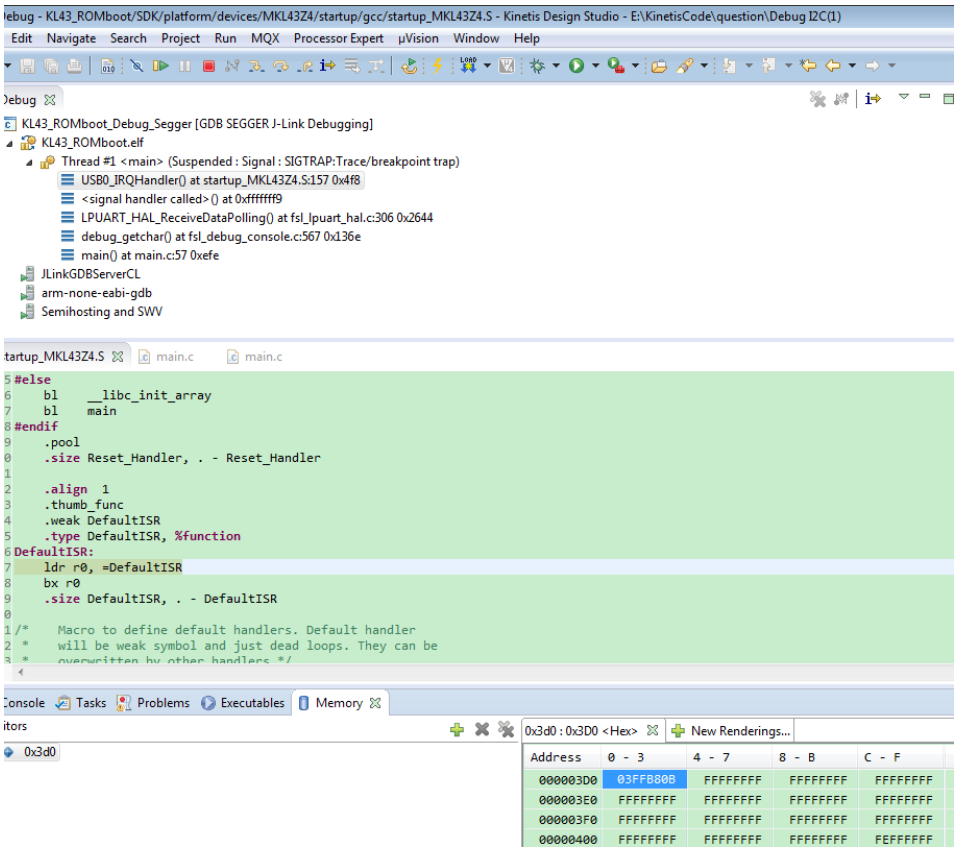
Boot from:	ROM peripheral	Test Result
Flash	XX	OK
ROM	0XFF, enable all	NO, UART can't receive
	0X08, enable USB	Yes, UART can receive
	0X04, enable SPI	Yes, UART can receive
	0X02, enable I2C	Yes, UART can receive
	0X01, enable LPUART	NO, UART can't receive

From the test result, we can reproduce the problem. The UART receive problem just happens on booting from ROM and the LPUART is enabled, when we run it with debugger, and test it step by step, we can find after the PTA1 have received the data, the code will run to the abnormal area.

Note:

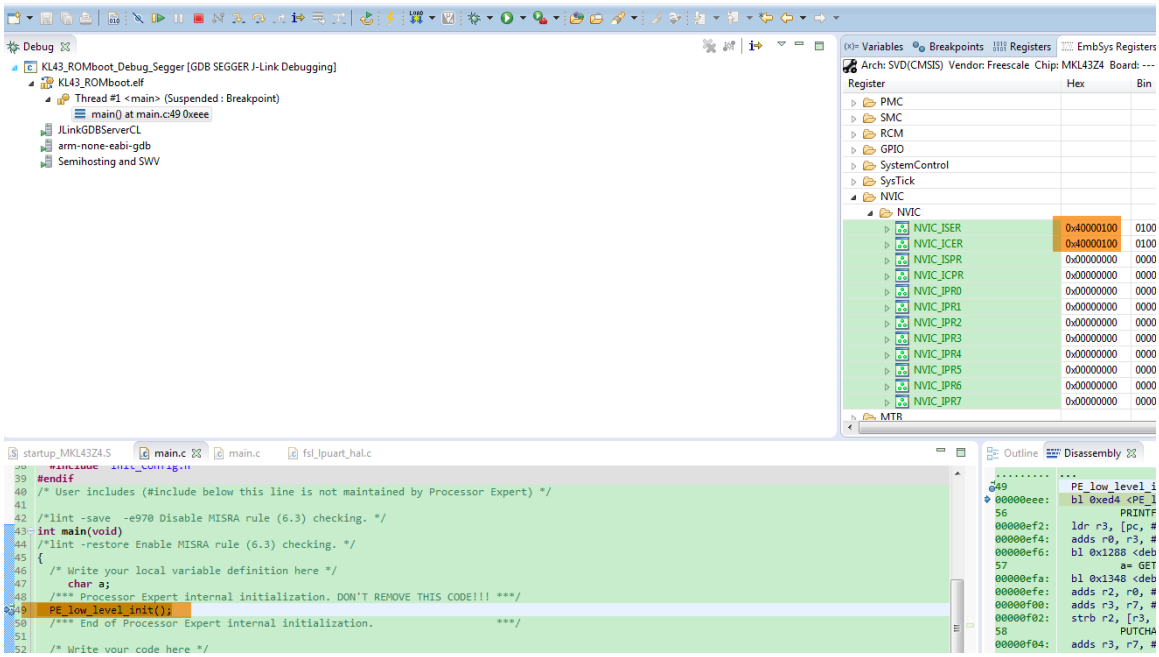
when debug this code, please choose the JLINK as the debugger, because the P&E tool will protect the FOPT area automatically in the KDS IDE when do debugging, the code will still run from flash, so if customer use the P&E tool, they will found the PTA1 still can receive the data, this is not the real result, but the JLINK won't protect FOPT area in the KDS IDE, it can reflect the real result.

After using the JLINK as the debugger, and we have found after PTA1 getting data or pulling low, the code will enter to the abnormal area like this:



We can get that the code run to the defaultISR, and display with USB_IRQHander, but this is not really the USB_IRQHander, just caused by the PC abnormal. Normally, it is caused by the missing of interrupt service function.

Now, we test the NVIC data to check which module interrupt caused this, the following picture is the result by enabling the LPUART and I2C peripheral in the ROM BCA area.



We can find, even we didn't do the cpu and peripheral initialization after booting from ROM, there still have peripheral be enabled, what the interrupt is enabled? From the definitive guide to the ARM Cortex-M0.pdf:

NVIC Register Summary

Address	Name	CMSIS Symbol	Full Name
0xE000E100	ISER	NVIC->ISER	Interrupt Set Enable Register
0xE000E180	ICER	NVIC->ICER	Interrupt Clear Enable Register
0xE000E200	ISPR	NVIC->ISPR	Interrupt Set Pending Register
0xE000E280	ICPR	NVIC->ICPR	Interrupt Clear Pending Register
0xE000E400	IPR0-7	NVIC->IPR[0] to NVIC->IPR[7]	Interrupt Priority Register

Interrupt Set Enable Register (NVIC -> ISER)

To enable an interrupt with a CMSIS-compliant device driver library, please use the `NVIC_EnableIRQ` function:

Address	Name	Type	Reset Value	Descriptions
0xE000E100	SETENA	R/W	0x00000000	Set enable for Interrupts 0 to 31; write 1 to set bit to 1, write 0 has no effect Bit[0] for Interrupt #0 (exception #16) Bit[1] for Interrupt #1 (exception #17) … Bit[31] for Interrupt #31 (exception #47) Read value indicates the current enable status

NVIC_ISER = 0x40000100, Vector46=IRQ30 and vector24=IRQ8 is enabled, it should be not disabled after booting from the ROM. Now check the KL43 reference manual, **Table 3-2. Interrupt vector assignments,** we can get that the I2C0 and PORTA interrupt is enabled.

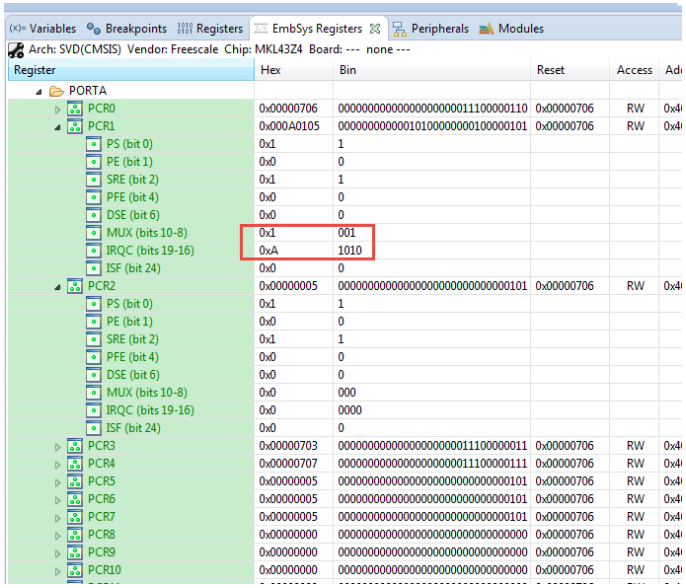
0x0000_005C	23	7	1	LLWU	Low Leakage Wakeup
0x0000_0060	24	8	2	I²C0	Status and Timeout and wakeup flags
0x0000_0064	25	9	2	I²C1	Status and Timeout and wakeup flags
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_006C	27	11	2	SPI1	Single interrupt vector for all sources
0x0000_0070	28	12	3	LPUART0	Status and error
0x0000_0074	29	13	3	LPUART1	Status and error
0x0000_0078	30	14	3	UART2 or FlexIO	Status and error—

Table continues on the next page...

Table 3-2. Interrupt vector assignments (continued)

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_007C	31	15	3	ADC0	Conversion complete
0x0000_0080	32	16	4	CMP0	Rising or falling edge of comparator output
0x0000_0084	33	17	4	TPM0	Overflow or channel interrupt
0x0000_0088	34	18	4	TPM1	Overflow or channel interrupt
0x0000_008C	35	19	4	TPM2	Overflow or channel interrupt
0x0000_0090	36	20	5	RTC	Alarm interrupt
0x0000_0094	37	21	5	RTC	Seconds interrupt
0x0000_0098	38	22	5	PIT	Single interrupt vector for all channels
0x0000_009C	39	23	5	I²S0	Single interrupt vector for all sources
0x0000_00A0	40	24	6	USB	—
0x0000_00A4	41	25	6	DAC0	—
0x0000_00A8	42	26	6	—	—
0x0000_00AC	43	27	6	—	—
0x0000_00B0	44	28	7	LPTMR0	LP Timer compare match
0x0000_00B4	45	29	7	SLCD	—
0x0000_00B8	46	30	7	Port control module	Pin detect (Port A)
0x0000_00BC	47	31	7	Port control module	Pin detect (Single interrupt vector for Port C and Port D)

Checking the PORTA register before do the cpu and peripheral initialization, PTA1 is enabled the port interrupt, and choose Flag and Interrupt on falling-edge.



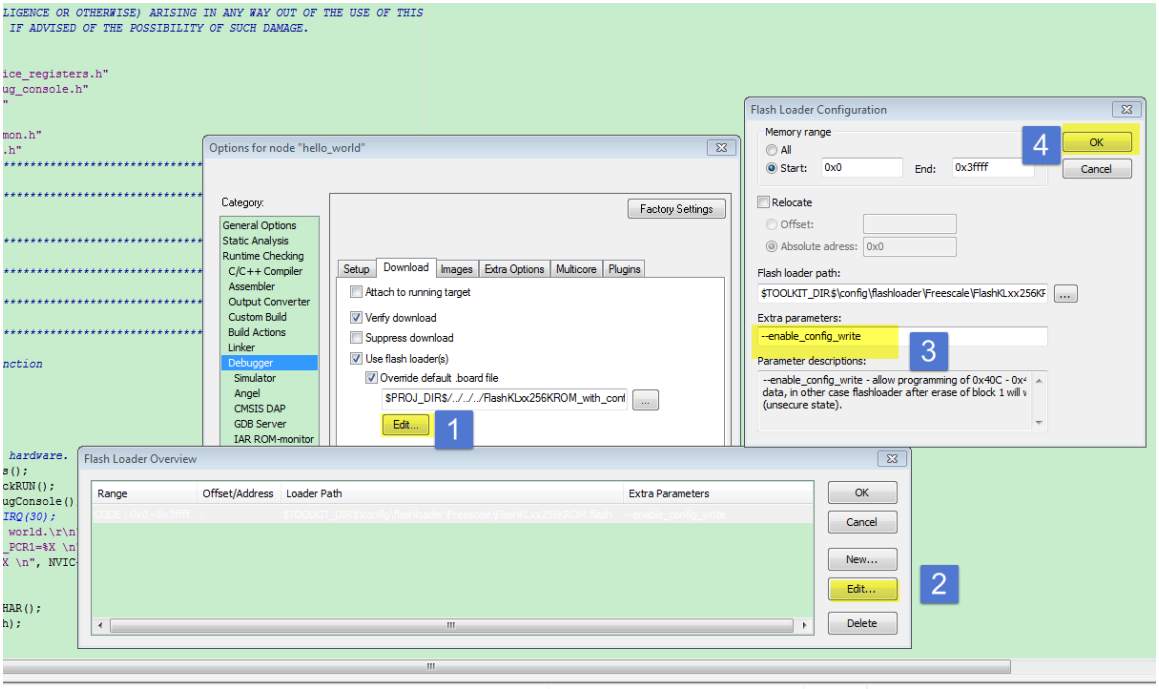
This can tell us why the PTA1 pin have the problem of uart receive data or give a falling edge in PTA1 will run abnormal, because in default, even we configure the PTA1 as the uart receive function, but the code didn't clear IRQ and NVIC register, when the signal happens on PTA1 pin, it will caused the PORTA interrupt, but we didn't add the PORTA interrupt ISR function, it is also not useful to us, then PC don't know where to go, so it will run abnormal, enter the defaultISR, and can't recover. If you have interest, you can add the PORTA_IRQHandler function, you will find the code will run to this function.

2.2 Problem reproduction in KSDK2.0 IAR project

Test project: SDK_2.0_FRDM-KL43Z\boards\frdmkl43z\demo_apps\hello_world

Test the official project just to make sure, it is really the chip hardware function, not only the problem from new generated code in KDS.

Because the IAR IDE will protect the 0X400 area, then if we want to modify the FOPT, we need to modify the .board, add –enable_config_write at first.



Then modify the FOPT in startup_MKL43Z.s:

```
__FlashConfig
      DCD 0xFFFFFFFF
      DCD 0xFFFFFFFF
      DCD 0xFFFFFFFF
      DCD 0xFFFFFFFE   ; 0xFFFF3FFE
__FlashConfig_End
```

Because the BCA peripheral area is in default as 0XFF, it enables all the peripheral, we don't need to define the BCA area independently.

For getting the real test result, we add the NVIC and PORTA_PCR1 register printf code in the main function,

```
PRINTF("PORTA_PCR1=%X \n", PORTA->PCR[1]);
PRINTF("NVIC=%X \n", NVIC->ICER[0U]);
```

And download the modified KSDK sample code to the chip, after testing, we get this result:

```
hello world.
PORTA_PCR1=A0205
NVIC=40000100
```

It is the same result as the new created project after booting from the ROM, PORTA interrupt and I2C interrupt is enabled, and it caused the PTA1 receive data problem.

3 Solutions and test result

3.1 Solutions

From the Chapter 2 testing and analysis, we can get that UART receive problem is caused by the PORT interrupt and NVIC is enabled after booting from the ROM, this should be caused by exiting the ROM, the ROM forget to disable it. We also can find some descriptions from the KL43 reference manual page 211:

13.6 Bootloader errata

The bootloader has the following errata:

1. PORT clock gate, pin mux and peripheral registers are not reset to default values on ROM exit

Description

- Affected PORT clock gates: PORTA, PORTB, PORTC, PORTD and PORTE (SIM_SCGC5_PORTA, SIM_SCGC5_PORTB, SIM_SCGC5_PORTC, SIM_SCGC5_PORTD and SIM_SCGC5_PORTE are enabled)
- Affected pin mux: LPUART0(PTA1, PTA2), I2C0(PTB0,PTB1), SPI0(PTC4,PTC5,PTC6,PTC7)
- Affected peripheral registers:
  - UART and UART clock source (SIM_SOPT2_UART0SRC = 3)
  - SPI
  - I2C

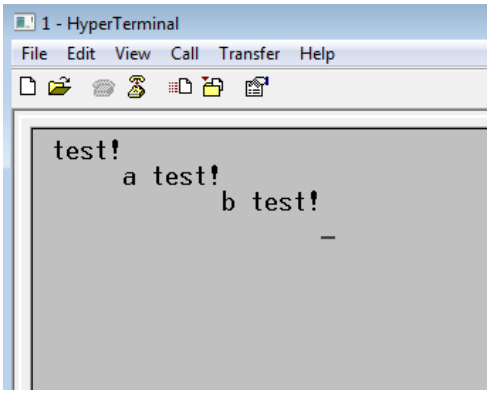
Workaround

- User must re-configure corresponding register to expected value instead of relying on the default value.

So, if customer want to solve this problem, to avoid the application enter to the abnormal area, we can disable the NVIC in the application code like this, the I2C NVIC is the same:

```
NVIC_DisableIRQ(8);//disable I2C0 interrupt
NVIC_DisableIRQ(30); //disable PTA interrupt
```

3.2 Test result



From the test result after adding the NVIC I2C and PORTA disable code, we can get the uart can works ok, if you have interest to test, the I2C will also work ok.

4 Conclusion

When customer use the kinetis chip KL43, KL27 and KL17 which flash size is above 128K, and want to boot from the ROM and enable the LPUART and I2C in BCA area, please add the NVIC I2C(IRQ8) and PORTA(IRQ30) disable code in the application code:

NVIC_DisableIRQ(8);//disable I2C0 interrupt

NVIC_DisableIRQ(30); //disable PTA interrupt

So far, I just find KL43, KL27 and KL17 which flash size is above 128K have this problem, other kinetis chip which have ROM bootloader don't have this problem.