Migrating S08 to Kinetis E MCU using KDS.

by: David Diaz Marin

Content

1 Introduction

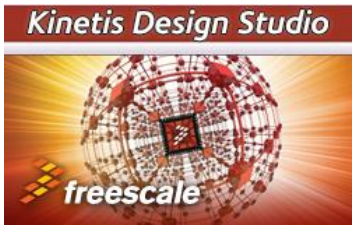
The migration between devices involves a lot of changes, one of these changes is the software.

The purpose of this document is to create a new KDS application and to mention the differences between S08 and Kinetis E devices.

Sample code and comparisons are shown along this document in order to know the main software considerations between the family devices mentioned.

2 Overview

2.1 KDS



The **Kinetis Design Studio** software development tool is a GNU/Eclipse-based development environment for Freescale Kinetis devices. It supports Cortex-M based Kinetis devices and integrates with Processor Expert and Kinetis Software Development Kit. KDS supports SEGGER J-Link/J-Trace, P&E USB Multilink Universal/USB Multilink Universal FX and CMSIS-DAP debug adapters and uses the newlib-nano C runtime library. This runtime library helps reduce the memory footprint of an embedded application.

In order to get more information about the IDE mentioned, please refer to the following link:

http://www.freescale.com/kds

3 Creating a new Project using KDS

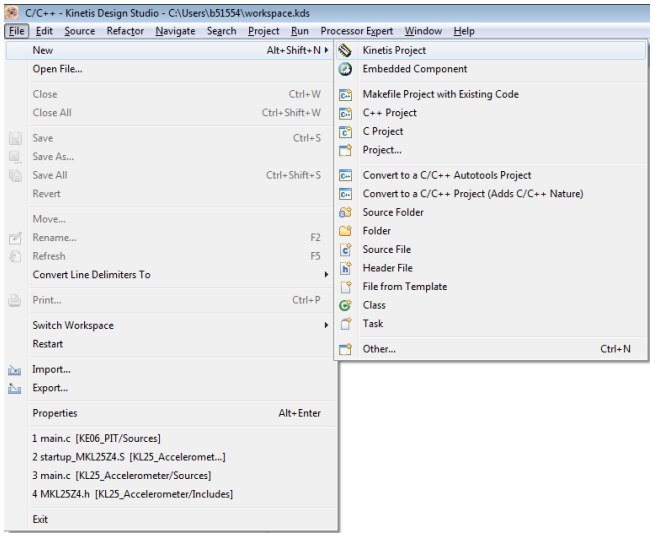
The Kinetis Project wizard help you to quickly create new projects. The wizard generates a project with placeholder files and default settings (build and launch configurations) specific targets. After the project has been created, you can easily change any default setting to suit your needs.

To create a Kinetis project using the New Kinetis Project wizard:

1. Launch the KDS.

2. Select **File** > **New** > **Kinetis Project**, from the IDE menu bar.

The Create a Kinetis Project page of the New Kinetis Project wizard appears.

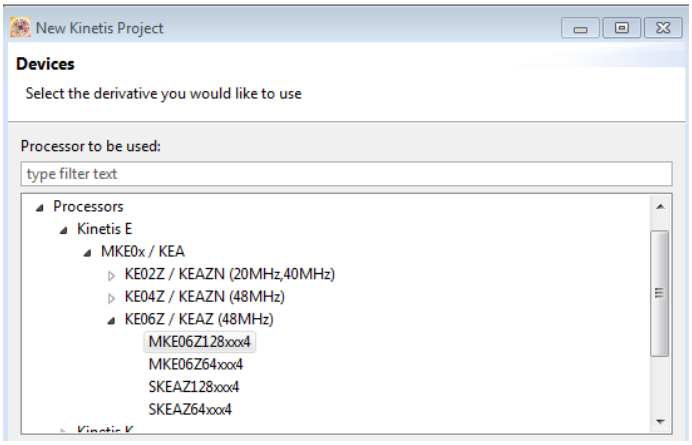


3. Specify a name for the new project. For example, enter the project name as *Project1*.

4. Click **Next**.

The Devices page appears.

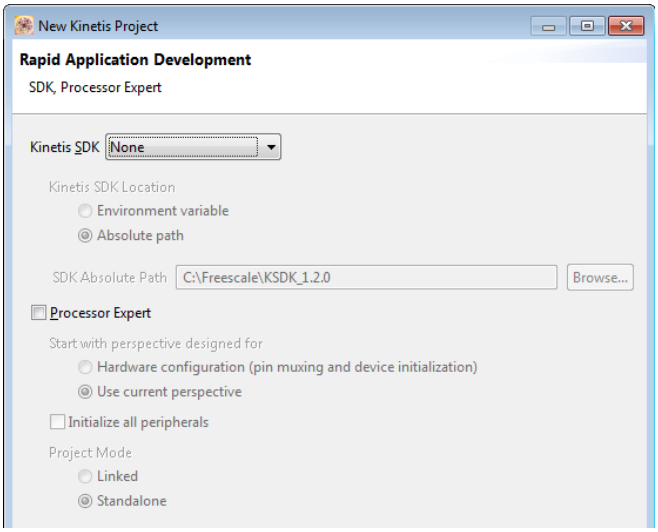
5. Expand the desired tree control and select the derivative or board you would like to use. For example, select **Processors** > **Kinetis K** > **MKE0x / KEA** > **MKE06 / KEAZ (48 MHz)** >**MKE06Z128xxx4**



6. Click **Next**.

The Rapid Application Development page appears. This page helps you to configure use Processor Expert for configuration and KSDK.

7. Select the default Kinetis SDK option to **None**.



8. Click **Finish**

4 Differences between the S08 and the Kinetis E devices.

The comparison topics will reside in the most relevant differences between the devices. Sample code is included in some topics.

The discussed topics are:

- Core (HCS08 MCU Core vs. Cortex M0+).
- Set of Instructions (Number of instructions and clock cycles).
- Memory Map.
- Interrupt Controller.
- Peripherals Differences.

3.1 HCS08 MCU Core vs Cortex M0+

The following table show the difference regarding the corresponding Cores on the devices mentioned:

	HCS08 MCU	**Cortex M0+**
CPU	• 8-bit processor core • 68HC08 instruction set • Up to 40 MHz Core Clock	• 32-bit embedded processor • Thumb 2 instruction set • Up to 48 MHz (today)
DEBUG	• Background debug controller • Non- intrusive software debugging • Allow a user to read or write MCU memory locations or access status and control registers • One hardware breakpoint	• CoreSight debug and trace • Non-intrusive – trace information • Allows the user to read or write MCU memory locations or access status and control registers • Up to 4 hardware breakpoints and 2 Watchpoints
Interrupt	• Up to 39 Interrupt vectors • Fixed vector priority. Highest priority at the bottom of the table	• 16 system interrupts + up to 32 peripheral interrupts (KE06) • Selectable 4 priority levels for peripheral interrupts

3.2 Set of instructions

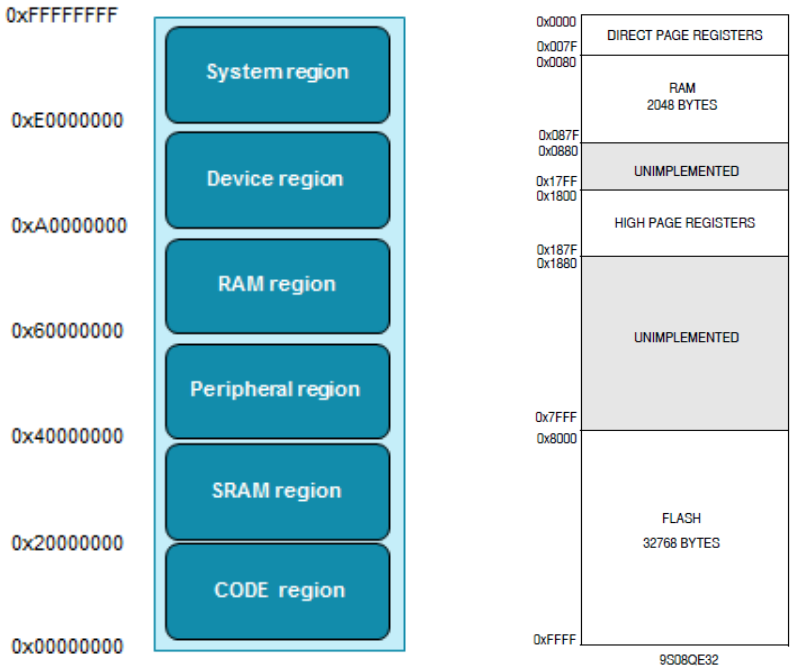
	HCS08 MCU	Cortex M0+
Instruction Set	• HCS08 instruction set	• ARMv6-M Thumb instruction set

Example of instructions and clock cycles:

Instruction	HCS08 MCU	Cortex M0+
Add with carry	• Immediate access mode • 2 cycles	• 1 cycle
Jump to Subroutine	• Dir, with offset • 5 cycles	• Any • 2 cycles

3.3 Memory Map

The main difference in the memory model is the way the addresses are accessed.



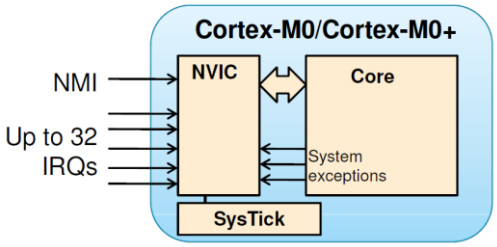
Kinetis vs S08 memory map

3.4 Interrupt Controller

	HCS08 MCU	Cortex M0+
Exceptions and Interrupts	• Up to 39 maskable interrupts and 1 non-maskable. • Fixed interrupt priority. Lowest vector number highest priority	• Nested Vector interrupt controller. • Up to 32 separate interrupt sources • Four selectable interrupt priority levels
Vector Table	• Located at fixed address in flash. • Every vector contains the address of the interrupt handler.	• Default address at 0x00000000 • Can be relocated during initialization. • Each vector contains the starting address of the corresponding interrupt handler
Interrupt handlers	• ISRs end with a RTI (return-from-interrupt) • Compiler directive to indicate the function is an ISR	• Entry and Exit sequences supported in hardware. • ISR routines written as standard C function

3.4.1 Nested Vectored Interrupt Controller (NVIC)

The NVIC is a standard module on the ARM Cortex M series. This module is closely integrated with the core and provides very low latency entering and exiting an interrupt service routine (ISR).



The NVIC provides four different interrupt priorities which can be used to control the order in which interrupts must be serviced. Priorities are 0-3, with 0 receiving the highest priority.

Configuring the NVIC for the specific module involves writing three registers:

- NVIC Set Enable Register (NVICSERx)
- NVIC Clear Pending Register (NVICCPRx)
- NVIC Interrupt Priority (NVICIPxx)

Sample Code

```
/* Set the ICPR and ISER registers accordingly */
NVIC_ClearPendingIRQ(PIT_CH1_IRQn);/* Clear pending interrupts*/
NVIC_EnableIRQ(PIT_CH1_IRQn);      /* Enable interrupts*/
```

In this case the first two functions are used in order to enable interrupts and clear pending interrupts.

The interrupt priority is left by default.

Note:

The parameter PIT_CH1_IRQn is defined at the corresponding interrupt vector at the device header.

The bellow code shows the different Interrupt Handler:

S08 ISR

```
/*****************************************************************************/
interrupt VectorNumber_Vmtim void Vmtim_isr(void)
{
     (void)MTIMSC;              /* Clear the TOF flag */
     MTIMSC_TOF = 0;
}
/*****************************************************************************/
```

Kinetis ISR

```
/*****************************************************************************/
void PIT_CH1_IRQHandler(void)
{
      // clear PIT ch1/timer1 interrupt flag
      PIT_TFLG1 = PIT_TFLG_TIF_MASK;
}
/*****************************************************************************/
```

3.5 Peripherals Differences

	HCS08 MCU	KE MCU
ICS Internal Clock source	• FLL selectable 512, 608, 1024, 1216, 1536, 1824 (QE)	• FLL multiplication factor fixed to 1024
SIM System Integration Module	• Includes Universal Unique Identifier (UUID) • Pin remapping is controlled by the SIM_PINSEL register. • There is no SIM_BUSDIV register. It cannot further divide bus clock and flash clock down. • All peripheral clocks are enabled after reset	• Includes fields for Kinetis Family ID, Sub-Family ID, Revision Number, and Device Pin ID. • Pin remapping is controlled by the SYS_SOPT1 register in S08P. The bus clock can be further divided by 2 for both bus and flash clock via the SIM_BUSDIV register. • All peripheral clocks are gated off at reset
Flash Memory Controller		• Support read-while-write. Allowing read from flash while programming/erasing the flash.
PinOut	• BKGD pin for debug 8 high-drive pins supporting 20 mA drive capability	• SWD_DIO and SWD_CLK pins for debug • 8 high-drive pins supporting 20 mA drive capability
Port Control and GPIO	• Access to it in 8 bit registers	• Access to it in 32 bit registers • Improved access to registers
ADC	• SAR ADC • ADC 12-bit result is read in more than one cycle	• SAR ADC • ADC 12-bit result is read in just one cycle
RTC Real Time Clock	• 8 bit register access width	• 32 bit register access width

TMP/FTM	• TPM	FTM • Signed up counter • Dead time insertion hardware • Fault control inputs • Enhanced triggering functionality • Initialization and polarity control
PIT Periodic Interrupt Timer		• Array of timer channels • 32-bit module counter • Each channel counts individually • Interrupt support

3.5.1 Clock gate

One of the main differences between the devices mentioned is the clock setting.

At Kinetis, each clock module must be enabled using the *SCGC* register from the System Integration Module.

The system integration module (SIM) provides system control and chip configuration registers.

12.2.6 System Clock Gating Control Register (SIM_SCGC)

Address: 4004_8000h base + 14h offset = 4004_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACMP1	ACMP0	ADC	0	IRQ	0	KBI1	KBI0	0	UART2	UART1	UART0	SPI1	SPI0	I2C1	I2C0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MSCAN	0	SWD	FLASH	0	CRC	0		FTM2	FTM1	FTM0	PWT	0		PIT	RTC
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Sample Code

```
SIM_SCGC |= SIM_SCGC_PIT_MASK; /*System Clock Gating Control Register (SIM_SCGC)*/
                              /*enable clock gate for PIT*/
```

Please note that each clock module must be enabled using the `SIM_SCGC` register.

In regard with the rest of the modules, please refer to the Kinetis E Reference Manual and the sample codes attached to this document.

5 References

KE06 Reference manual.
http://www.freescale.com/files/microcontrollers/doc/ref_manual/MKE06P80M48SF0RM.pdf

Migration Guide From S08 to Kinetis E Family.
http://www.freescale.com/files/32bit/doc/app_note/AN4757.pdf

Migration Guide From S08 to Kinetis L Series MCUs.

http://www.freescale.com/files/32bit/doc/app_note/AN4662.pdf