

High Speed Run Mode With Kinetis K22F and KV31F MCUs

This application note, written for the latest generation of Kinetis K22F and KV31F series microcontrollers, explains the benefits and usage of the new run mode for this family called High Speed Run mode (HSRUN). HSRUN mode enables the maximum performance of these 120 MHz and 100 MHz ARM® Cortex®-M4 based MCUs with floating point units, enabling delivery of best-in-class CoreMark/MHz scores.

The general idea driving the addition of HSRUN mode is to enable system designers to configure the K and V series MCUs to achieve the maximum frequency, to execute a computational task as quickly as possible, and then return to normal or lower power run modes to conserve power.

Contents

1. Applications for High Speed Run	2
2. Using High Speed Run mode	2
3. Kinetis SDK example application	3
4. Key source code from the High Speed Run demo ...	7
5. Results and conclusions	9
6. References	11
7. Revision History	11

1 Applications for High Speed Run

The new HSRUN mode is well-suited for compute intensive applications, including sensor fusion, energy metrology, and body sensor data for healthcare devices. These applications are likely to include complex algorithms requiring functions ranging from single bit manipulation, to fixed and floating point math, as well as data buffering and movement in memory. In practice, algorithms which perform compression, signal encoding/decoding, or mathematical transforms would see greatly reduced execution times and improved efficiency in HSRUN mode.

Obviously, this increase in performance comes at the cost of increased current consumption, thus increased operating frequency will have a proportional effect on power consumption. When considering the use of HSRUN mode, we recommend system designers carefully consider the maximum allowed current consumption, maximum power consumption, and the operating temperature of the end application.

Later in this document the details of a demo application are provided, based on the Kinetis SDK platform, which executes multiple floating point fast-Fourier transforms (FFTs) as a CPU load test case. The comparative data for execution time and power consumption of this practical example is shared as well as the analysis techniques.

2 Using High Speed Run mode

In HSRUN mode, the on-chip voltage regulator remains in a run regulation state, while enabling a higher current capacity mode. In this state, the MCU is able to operate at a higher frequency compared to normal RUN mode. The Power Management chapter in the device's Reference Manual outlines the maximum allowable frequencies.

While in HSRUN mode, the following restrictions must be adhered to:

- The maximum allowable change in frequency of the system, bus, flash or core clocks is restricted to x2.
- Before exiting HSRUN mode, clock frequencies should be reduced to those acceptable in RUN mode.
- STOP mode entry is not supported from HSRUN.
- Modifications to clock gating control bits are prohibited.
- Flash programming/erasing is not allowed.

To enter HSRUN mode, the SMC_PMPROT register must first be set to allow for HSRUN mode. The protection feature is intended to prevent the unintentional entry into unsupported run modes. After the protection is configured, simply set SMC_PMCTRL[RUNM] to the HSRUN value. Before increasing clock frequencies, the SMC_PMSTAT register should be polled to determine when the system has completed entry into HSRUN mode. To re-enter normal RUN mode, clear the SMC_PMCTRL[RUNM]. Any reset will also clear RUNM and cause the system to exit to normal RUN mode after the MCU exits its reset flow. These details are included to explain what happens at the device register, however it should be noted that the setting of these fields is fully supported within the Kinetis SDK API's for SMC block which is explained in the next section of this document.

Table 1 details the most common internal clock settings for both RUN and HSRUN modes, as stated in the device's Reference Manual.

Table 1. Common internal clock settings

Clock	120 MHz Capable Devices		100 MHz Capable Devices	
	RUN Mode	HSRUN Mode	RUN Mode	HSRUN Mode
Core clock	80 MHz	120 MHz	72 MHz	100 MHz
System clock	80 MHz	120 MHz	72 MHz	100 MHz
Bus clock	40 MHz	60 MHz	36 MHz	50 MHz
FlexBus clock	20 MHz	30 MHz	NA	NA

3 Kinetis SDK example application

Kinetis SDK (KSDK) is a Software Development Kit that provides comprehensive software support for the core and peripherals of all Freescale Kinetis devices with Cortex®-M cores. The KSDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on the HAL. Example applications are provided to demonstrate driver and HAL usage to highlight the main features of targeted SoCs.

3.1 Description of High Speed Run demo application

The “highspeed_run_demo” referenced below is a simple shell like application that allows for switching between RUN and HSRUN modes safely and also includes a CPU load test case in the form of a custom complex 32-bit floating point FFT algorithm. This FFT is a decimation in time complex algorithm that is computationally more complex than the ARM CMSIS-DSP, `arm_cfft_f32()` function and thus the run times included below should not be considered for any type of benchmarking. The demo load test performs a 512-bin complex FFT on both real and imaginary data arrays containing 16 Kbytes of data each, repeated N times. The intent was to use a load test with a significantly long run time to highlight the comparative advantages of HSRUN mode versus RUN mode for the same arbitrary test case.

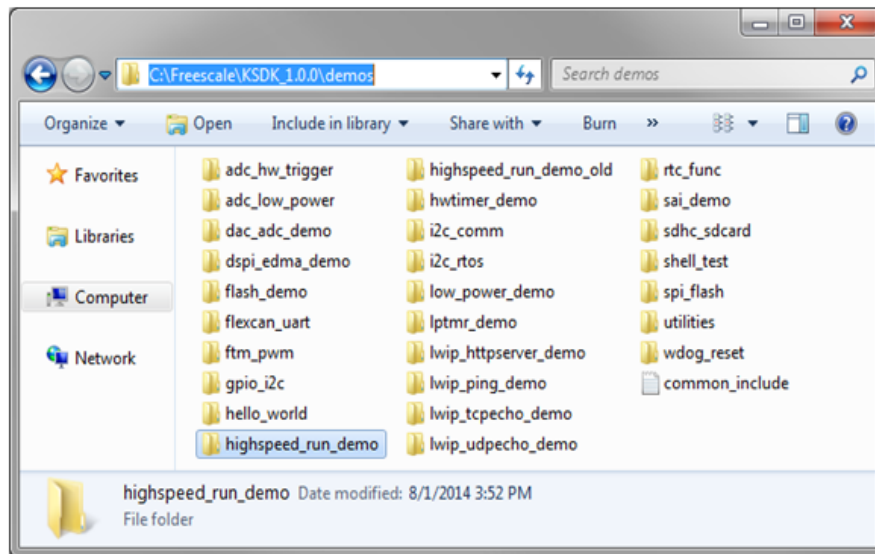
3.2 Installing the High Speed Run demo

In this section, the steps required to install and test a demo application, called the “highspeed_run_demo” within the latest KSDK V1.0 – RCS release, are provided.

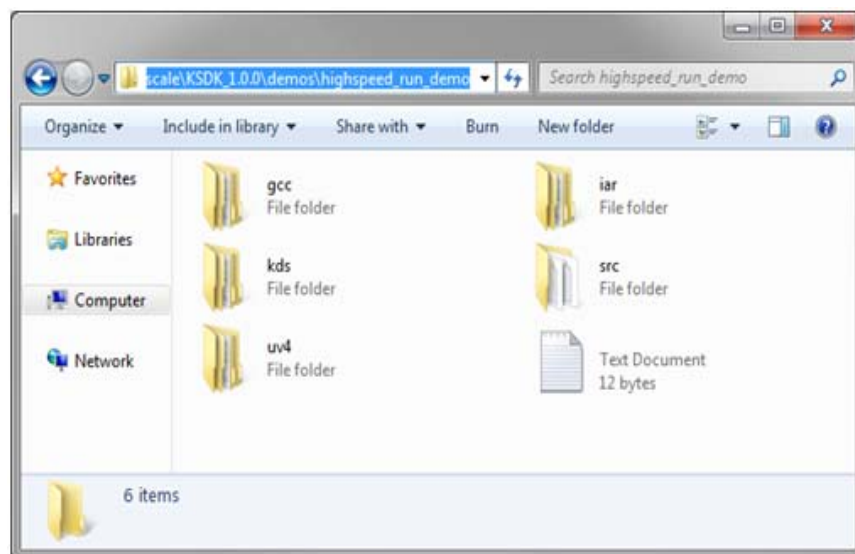
Steps to install, build and run the HSRUN demo:

1. Download the latest version of KSDK V1.0:
http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK
2. Launch the downloaded executable and click through the installation dialogues for the SDK. The SDK should install to an <Install_dir> similar too: “C:\Freescale\KSDK_1.0.0\”
3. Download the “highspeed_run_demo.zip” file at the following location:
http://cache.freescale.com/files/microcontrollers/doc/app_note/AN4985sw.zip?fpsp=1

4. Drop the extracted “highspeed_run_demo” folder from the .zip into the following location:
“C\:<Install_dir>\demos”



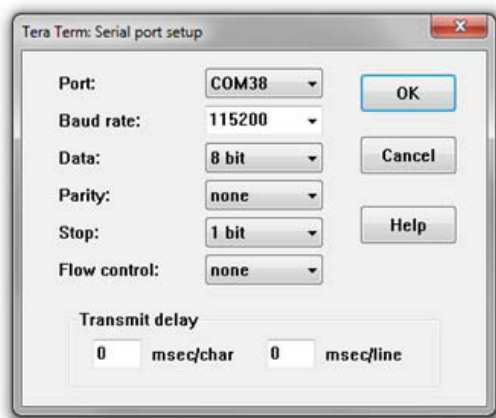
5. Within the “highspeed_run_demo” folder, the pre-built projects folders for the supported tool chains can be found (IAR, Keil, ARM Gcc, or Kinetis Design Studio).



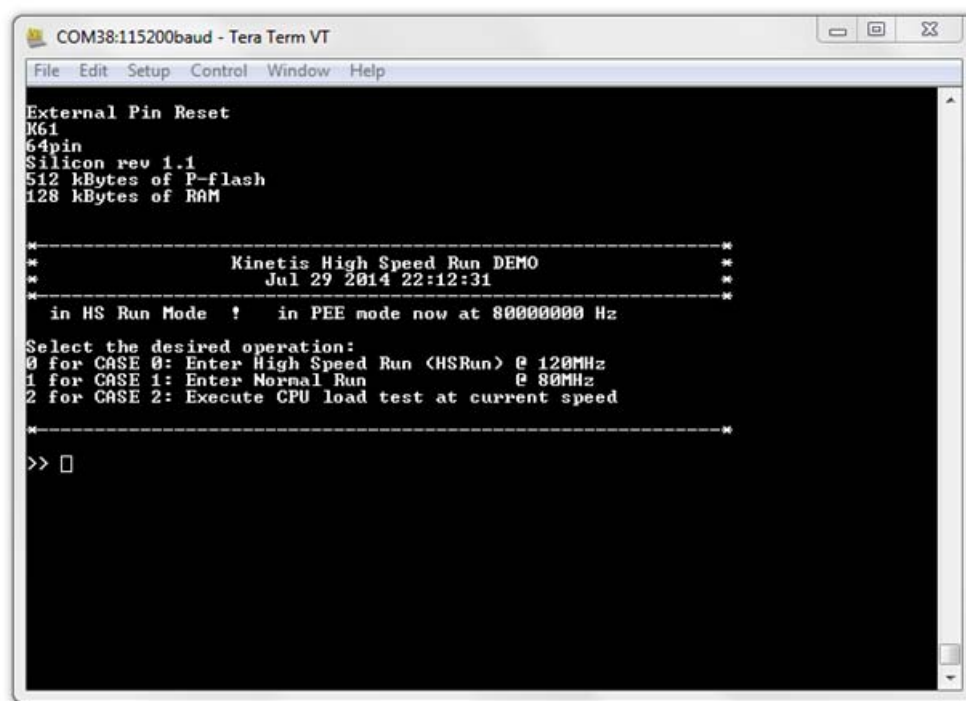
Within the folder for each tool chain, a “frdmk22f120m” and “twrk22f120m” can be found which correspond to the FRDM and TWR boards for the K22F device respectively. Open the appropriate folder for your tool chain of choice and then open the folder for the board type for your specific setup to find the IDE projects.

6. First, build the platform project for the KSDK library, then build the “highspeed_run_demo” project corresponding to the tool chain being used. For more details on the build procedures for each tool chain please review the Kinetis SDK K22 User’s Guide (KSDKK22UG).

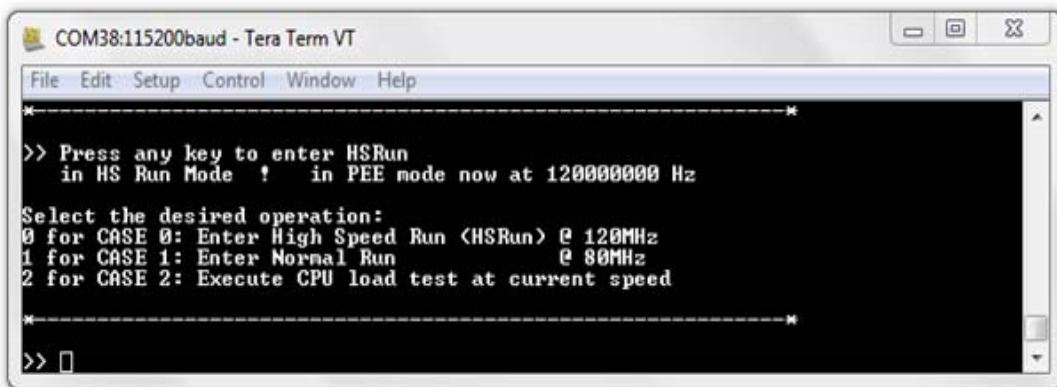
7. Open a terminal application (Tera Term shown in this example: <http://ttssh2.sourceforge.jp/index.html.en>) and configure a connection to the virtual COM port for the appropriate target board with the following settings:



8. Switch back to the IDE window, launch the debugger, program the code to your FRDM-K22F120M or TWR-K22F120M target, and run the application.
9. When the “highspeed_run_demo” executes, a terminal printout that looks like the following should appear:



10. The terminal application has three options:
 - Press ‘0’ followed by any key and the application will put the device in HSRUN mode at 120 MHz.



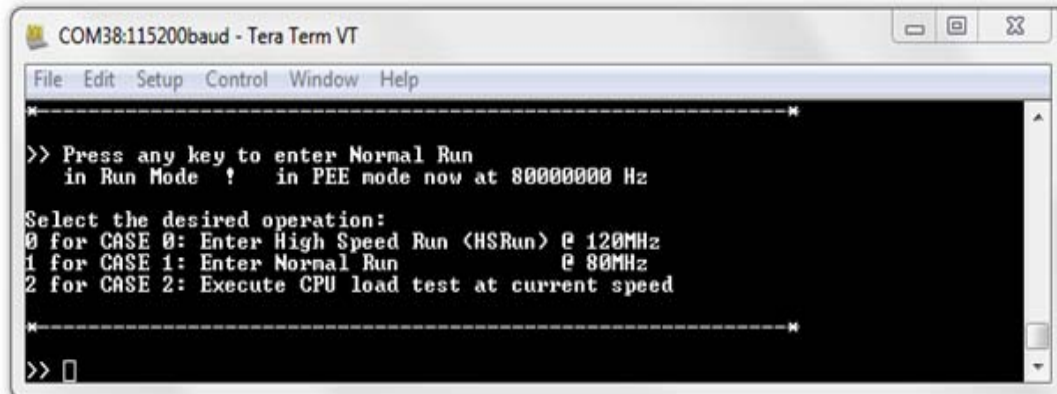
```

COM38:115200baud - Tera Term VT
File Edit Setup Control Window Help
-----*-----
>> Press any key to enter HSRun
    in HS Run Mode !    in PEE mode now at 120000000 Hz

Select the desired operation:
0 for CASE 0: Enter High Speed Run <HSRun> @ 120MHz
1 for CASE 1: Enter Normal Run           @ 80MHz
2 for CASE 2: Execute CPU load test at current speed
-----*-----
>> 

```

- Press '1' followed by any key and the application will put the device in normal RUN mode at 80 MHz .



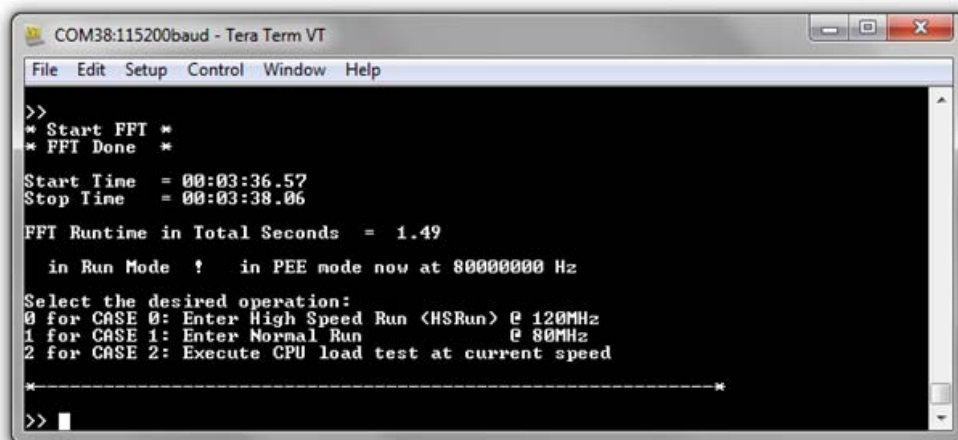
```

COM38:115200baud - Tera Term VT
File Edit Setup Control Window Help
-----*-----
>> Press any key to enter Normal Run
    in Run Mode !    in PEE mode now at 80000000 Hz

Select the desired operation:
0 for CASE 0: Enter High Speed Run <HSRun> @ 120MHz
1 for CASE 1: Enter Normal Run           @ 80MHz
2 for CASE 2: Execute CPU load test at current speed
-----*-----
>> 

```

- Pressing '2' will execute 50 iterations of a 2048 point FFT on a sine wave that is loaded in device SRAM memory. The run time of the FFT-CPU load test will be printed to the terminal window.



```

COM38:115200baud - Tera Term VT
File Edit Setup Control Window Help
-----*-----
>>
* Start FFT *
* FFT Done *

Start Time  = 00:03:36.57
Stop Time   = 00:03:38.06

FFT Runtime in Total Seconds = 1.49

    in Run Mode !    in PEE mode now at 80000000 Hz

Select the desired operation:
0 for CASE 0: Enter High Speed Run <HSRun> @ 120MHz
1 for CASE 1: Enter Normal Run           @ 80MHz
2 for CASE 2: Execute CPU load test at current speed
-----*-----
>> 

```

11. Switch between each run mode (HSRUN and RUN) and then execute the CPU load test for each to see the difference in run time. A typical result for 50 FFTs in HSRUN at 120 MHz mode will be approximately 1 second, while the same 50 FFTs in RUN mode at 80 MHz will take approximately 1.5 seconds. The test completes approximately 50 percent faster in HSRUN mode, as expected.

4 Key source code from the High Speed Run demo

Inside the root directory of the “highspeed_run_demo,” the ./src directory includes all of the demo specific source code required for this demo. The data structures and function calls required for configuring the device for HSRUN mode can be found within the “hsrun_power_modes_test()” function found within “highspeed_run_demo.c”

This function includes calls to the System Mode Controller (SMC) platform library structures and API’s found within the KSDK. A majority of the following will be defined within the “fsl_smc_hal.c/h” files located at: <Install_dir>/platform/hal/smc/

Detailed documentation on all KSDK APIs is provided within the Kinetis SDK API Reference Manual (KSDKAPIRM).

In the variable declaration section, at the top of the “hsrun_power_modes_test()” function found within “highspeed_run_demo.c”, the following two data structures are defined. These structures instantiate the run mode configuration structure as well as the power mode protection structure:

```
/* declare power mode config structure */
smc_power_mode_config_t smcConfig;
/* declare power mode protection for HSRUN */
smc_power_mode_protection_config_t pmodes = {
    .vlpProt = false,
    .llsProt = false,
    .vllsProt = false,
    .hsrunProt = true
};
```

The above structures will be used by the SMC HAL functions SMC_HAL_SetRunMode() and SMC_HAL_SetProtection() later in the code. The first step will be to enable the HSRUN mode protection. This function sets the appropriate bit value within the SMC_PMPROT register to allow for HSRUN mode.

```
/* Configure power mode protection settings for the application*/
SMC_HAL_SetProtection(SMC_BASE, &pmodes);
```

Continuing down the source the next useful SMC HAL function that's is used is the SMC_HAL_GetStat() function which is used in the following if() statement to determine and print out the devices current run mode while the main while() loop of the test function:

```
if (SMC_HAL_GetStat(SMC_BASE) == kStatRun){  
    printf("  in Run Mode  !  ");  
} else if (SMC_HAL_GetStat(SMC_BASE) == kStatHsruntime){  
    printf("  in HS Run Mode  !  ");  
}
```

The source code that executes the actual switching between RUN and HSRUN modes can be found within the “switch(testNum){}” statement. Within the “case 0” statement, you will find the statements which puts the device into HSRUN mode.

The CLOCK_HAL_SetOutDividers() function configures the settings for all clock out dividers at the same time. See the device reference manual for the supported clock dividers values ranges.

```
/* Setup Dividers for HSRun Mode */  
/* core/system=120(0), bus=60(1), FlexBus=30(3), flash=24(4) [MHz] */  
CLOCK_HAL_SetOutDividers(SIM_BASE, 0,1,3,4);
```

The SMC_HAL_SetRunMode() function sets the SMC_PMCTRL[RUNM] register to HSRUN mode.

```
/* Go to HSRun*/  
/* set power mode to HS Run mode */  
SMC_HAL_SetRunMode(SMC_BASE, kSmcHsruntime);
```

The SMC_PMSTAT register is then polled to confirm successful entry into HSRUN mode:

```
/* poll PMSTAT until Run mode has been entered */  
while(SMC_HAL_GetStat(SMC_BASE) && kStatHsruntime != kStatHsruntime)  
{}
```

The device is configured to operate in PEE mode during the initialization of the demo application, so we must traverse the MCG mode state machine back to FBE mode. Next, set the PLL reference divider (PRDIV) and divide ratio (VDIV) for the VCO output of the PLL corresponding to valid settings for HSRUN mode. Then return to PEE mode at 120 MHz in HSRUN mode.


```

/* Configure and enable clocks for HSRun mode */
pee_pbe(CLK0_FREQ_HZ);
pbe_fbe(CLK0_FREQ_HZ);
fbe_pbe(CLK0_FREQ_HZ, PLL0_PRDIV_HS, PLL0_VDIV_HS);
mcgClkHHz = pbe_pee(CLK0_FREQ_HZ);

```

The device should now be operating at 120 MHz, PEE, and HSRUN which will be confirmed on the terminal prompt. Alternately, if we examine the “case 1” statement, the steps and sequence required to exit HSRUN mode and enter normal RUN mode are shown below:

```

/* Configure and enable clocks for Run mode */
pee_pbe(CLK0_FREQ_HZ);
pbe_fbe(CLK0_FREQ_HZ);
fbe_pbe(CLK0_FREQ_HZ, PLL0_PRDIV, PLL0_VDIV);
mcgClkHHz = pbe_pee(CLK0_FREQ_HZ);

/* Setup Dividers for Run Mode */
/* core/system=80(0), bus=40(1), FlexBus=20(3), flash=26.67(2) [MHz] */
CLOCK_HAL_SetOutDividers(SIM_BASE, 0,1,3,2);

/* Currently in HS Run. Exit HS Run */
SMC_HAL_SetRunMode(SMC_BASE, kSmcRun);
while(SMC_HAL_GetStat(SMC_BASE)&& kStatRun != kStatRun)
{
}

```

The “case 2” statement contains the RTC stopwatch functions to measure run time and the code which executes N number (default code is set to 50) of complex floating-point FFTs.

5 Results and conclusions

In this application note, the HSRUN mode supported by the new Kinetis K22F and KV31F device families is introduced. A demo application called the “highspeed_run_demo” was shared and detailed. To further elaborate on the added benefit of the new HSRUN mode, the data shared within this final section was collected on a TWR-K22F120M board, but could also be repeated on a FRDM-K22F120M board as well.

The J15 jumper was removed and replace with a 10 Ω resister in order to monitor the current consumption of the HSRUN demo during execution. An oscilloscope with a differential probe was used to measure the voltage change (delta current) across the 10 Ω resister. This setup was then used to measure and record the transient current consumption of the FFT load test in both HSRUN and normal RUN modes.

Figure 1 shows a current profile comparison of a single FFT iteration in both HSRUN@120MHz and RUN@80MHz.

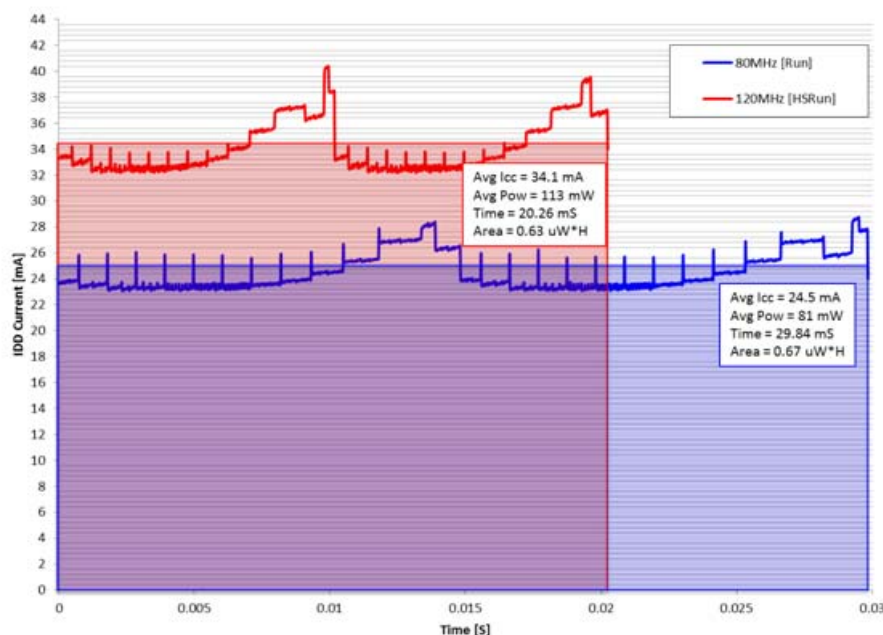


Figure 1. Single FFT iteration; current vs. time comparison

The key piece of information to note from Figure 1 is the area under the curve for each run mode. This area corresponds to the energy used to complete the same task in each run mode. The FFT executed in HSRUN mode at 120 MHz requires 0.04 uWh less energy to complete the FFT than the normal RUN mode. The demo was modified to execute 200 iterations of the complex FFT in both HSRUN and normal RUN mode, as well as being compiled to run from Flash-only and SRAM-only. The results of this experiment can be found in Table 2.

Table 2. Results of FFT executed in HSRUN mode

Code	Mode	Clock	Avg Current [mA]	Power [W]	Delta Current [mA]	200 FFTs			
						Total Runtime [s]	Performance Improvement	Energy [mWh]	Energy Savings
Flash	HSRUN	120 MHz	34.1	0.11253	9.6	4.1	47%	0.0128	6%
	RUN	80 MHz	24.5	0.08085		6.04		0.0136	
SRAM	HSRUN	120 MHz	29.6	0.09768	8	3.92	50%	0.0106	9%
	RUN	80 MHz	21.6	0.07128		5.88		0.0116	

The data shows that HSRUN mode consumes 8 to 9.6 mA more current on average while delivering a 47% to 50% performance improvement in terms of run time, depending on whether you are executing from Flash or SRAM. However, the important figure of merit is found within the energy and energy savings

columns which show the MCU running in HSRUN mode being 6% to 9% more efficient at completing the FFT load test.

6 References

Kinetis K22F Data Sheet (K22P121M120SF8)

Kinetis K22F Reference Manual (K22P121M120SF7RM)

Kinetis SDK API Reference Manual

Kinetis SDK K22 Users Guide

ARM CMSIS-DSP Reference:

http://www.keil.com/pack/doc/cmsis/dsp/html/group___complex_f_f_t.html

Tera Term Application Link: <http://ttssh2.sourceforge.jp/index.html.en>

7 Revision History

Rev. number	Date	Substantive change(s)
0	09/2014	Initial release



How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. ARM and Cortex are the registered trademarks of ARM Limited. All other product or service names are the property of their respective owners.

© 2014 Freescale Semiconductor, Inc.

