AN204774

F²MC-16FX Family, Clocks

This application note describes about 16FX Family MCUs feature a sophisticated clock distribution scheme with the different clock sources such as RC Clock, Main Clock, PLL Clock, Modulated PLL Clock, and Sub Clock.

Contents

1 Introduction 1
2 Clock Tree 1
2.1 Overview 1
2.2 RC Clock 2
2.3 Main Clock 3
2.4 PLL Clock 3
2.5 Sub Clock 3
2.6 Clock Modulator 3
3 Configuration 4
3.1 Using the *start.asm* file 4
3.2 Caution on Flash Timing 6
3.3 Background debugging in other clock modes than PLL clock mode 6
3.4 Manual Clock Selection 6
3.5 Clock Modulator Configuration 10
4 Clocks Example 10
4.1 Manual Clock Setting 10
4.2 Starting Clock Modulator 11
4.3 Stopping Clock Modulator 11
4.4 Transition to RC clock or sub clock mode and back to PLL clock mode 12
5 Additional Information 13
6 Document History 14

1 Introduction

The 16FX Family MCUs feature a sophisticated clock distribution scheme with the different clock sources such as RC Clock, Main Clock, PLL Clock, Modulated PLL Clock, and Sub Clock. The core and the peripherals are connected to different clock trees that can be connected to different clock sources and clock frequencies to allow for a fine-grained control over the required operation speed and power consumption.

2 Clock Tree

The 16FX Family MCUs feature a sophisticated clock distribution that allows for fine-grained control over the used clocks and frequencies. The details are given below.

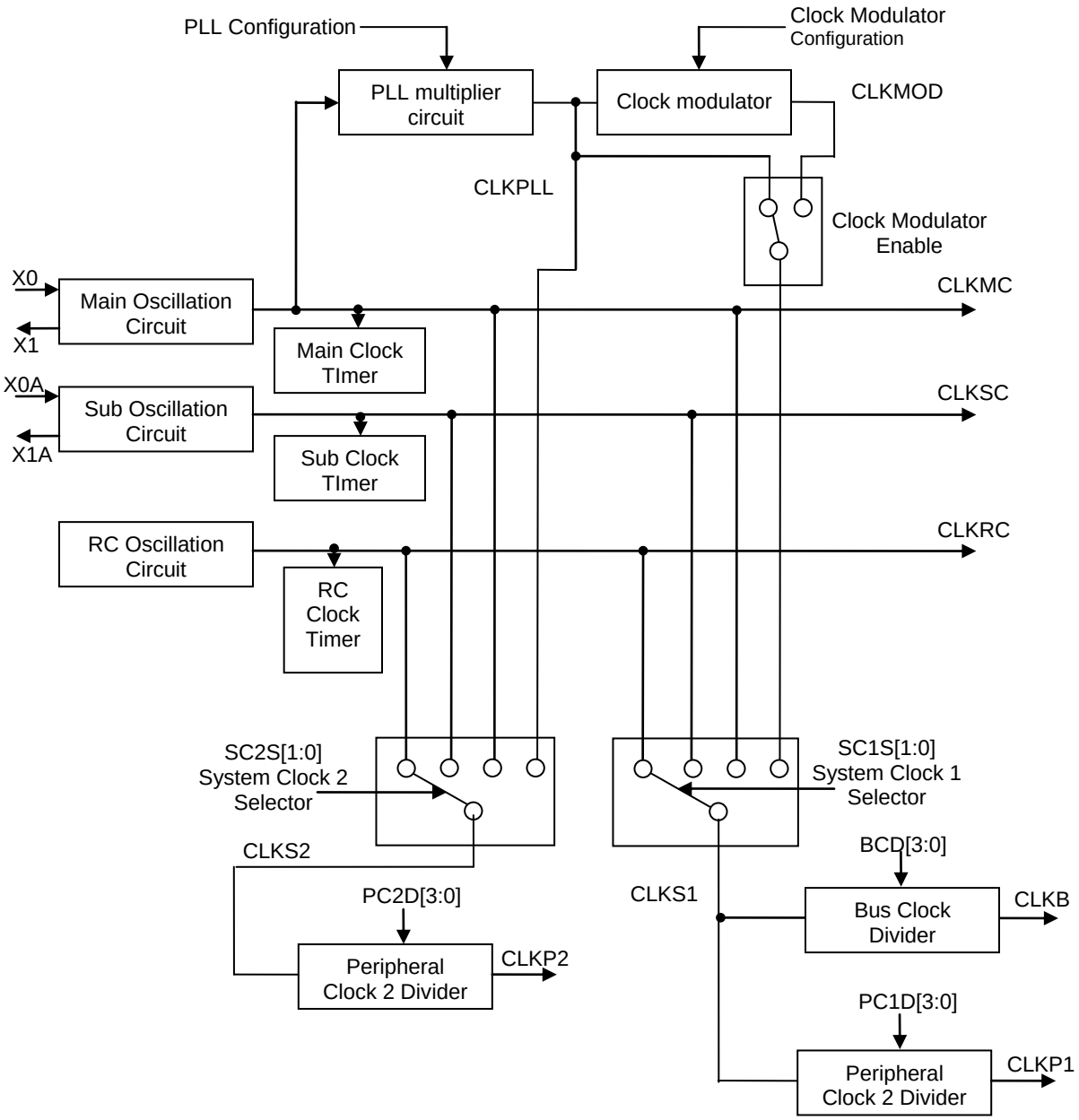
2.1 Overview

The 16FX Family can be used with 2 different external clocks: the Main Clock and the Sub Clock and 1 internal (on-chip) clock: the RC Clock. The external clocks may be connected to an oscillator, an oscillation circuit of a crystal and capacitors, or an external clock supply. The internal RC Clock can be configured at 2 MHz or 100 kHz. The availability of the Sub Clock is optional and depends on the actual device.

Internally, 3 different clocks are used: the core clock CLKB, the peripheral clock 1 CLKP1, and the peripheral clock 2 CLKP2. The CLKB provides the clock to the CPU. The CLKP2 provides the clock to the CAN controllers and the Sound Generator, if available. The CLKP1 provides the clock for all other peripherals.

Each of these three clocks features a separate prescaler. The system clocks CLKS1 and CLKS2 connect these prescalers and the selected clock source (one among RC, Sub and PLL/Modulated PLL Clock). CLKS1 connects the selected source clock with CLKB and CLKP1, CLKS2 connects the source clock with CLKP2. The following block diagram shows the clock distribution.

Figure 1. Block Diagram of Clock Circuitry



2.2 RC Clock

The RC clock is available on all 16FX devices. The RC clock CLKRC is the output clock of the internal RC oscillator. By default, it is enabled and selected as a source for CLKB, CLKP1 and CLKP2. It can be used as a clock source unless the other desired clocks are up and running.

It can be configured to generate 1 MHz to 4 MHz or 50 kHz to 200 kHz of clock frequency. The typical value of these clocks is 2 MHz and 100 kHz, respectively.

2.3 Main Clock

The Main Clock is available on all 16FX devices. It should be used as the primary clock in all applications. By default, it is enabled, but needs to be selected before use.

The allowed input frequency is limited to the range 3.5 MHz to 16 MHz when an oscillation circuit is used. The range is from 3.5 MHz to 56 MHz when an external clock signal is supplied and the Fast Clock Input is enabled. With respect to EMI considerations, a frequency of 4 MHz is recommended.

2.4 PLL Clock

The PLL Clock is available on all 16FX devices and is based on the Main Clock. By default, it is disabled. Before enabling it, it must be configured. The PLL can be set to a multiplier from 1 to 25. In any case, the maximum allowed frequency of CLKB (CPU Clock) is 56 MHz.

The PLL Clock also feeds the Clock Modulator. The Clock Modulator is discussed in subsequent sections.

2.5 Sub Clock

The Sub Clock is available on all 16FX dual-clock devices. It can be used as a supplemental clock for low-power applications. By default, it is enabled, but needs to be selected before use.

The allowed input frequency is limited to the range 32 kHz to 100 kHz when an oscillation circuit is used, while the range is extended from 0 Hz to 100 kHz when an external clock signal is supplied. Most time-dependent peripherals offer clock prescalers to allow for full-second intervals when a frequency of 32.768 kHz is provided.

2.6 Clock Modulator

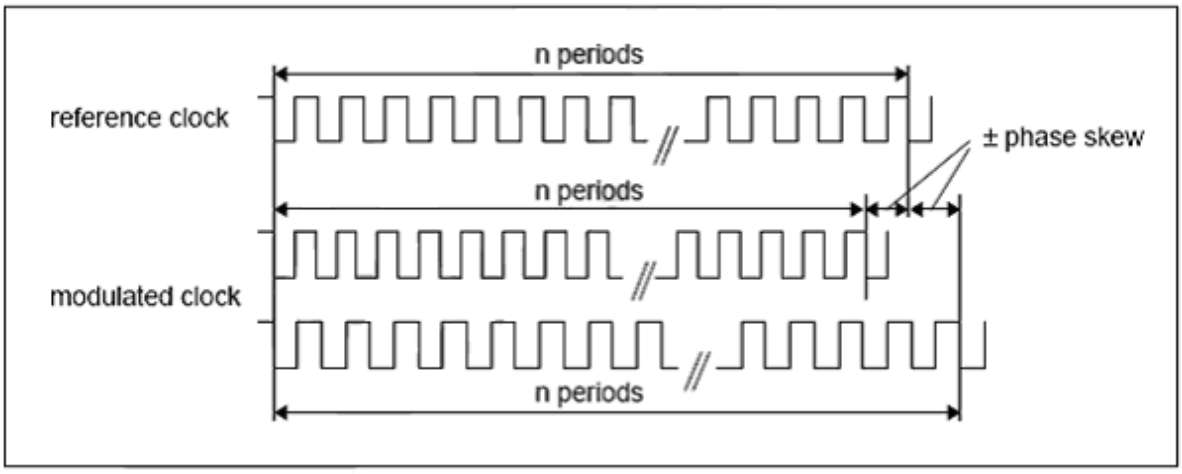
The clock modulator is primarily used to minimize the electromagnetic interference (EMI). It scatters the spectrum of the clock signal over a wide range of frequencies in order to do so.

The source of the clock for the modulator is the PLL Clock (CLKPLL). The minimum and maximum frequency for a given reference frequency of CLKPLL and the degree of modulation is dependent on the parameters like resolution of modulation range and modulation degrees. The average frequency of the resultant Modulated PLL Clock - CLKMOD is equal to the reference frequency CLKPLL.

As a result of the modulation, there results in skew in the phase of CLKMOD with reference to CLKPLL. This ‘Phase Skew’ is the maximal phase shift of the CLKMOD relative to the CLKPLL in terms of clock periods of the CLKPLL. The phase skew is dependent on configuration such resolution and modulation degrees at a given PLL frequency (CLKPLL).

The following figure explains the phase skew:

Figure 2. Phase Skew



Example:

For CLKPLL of 16 MHz (at a given CMPR configuration), if 100 periods of CLKMOD takes 6.5 microseconds then phase skew is calculated as follows:

One period of CLKPLL = (1/16 MHz) = 62.5 ns, hence the Phase Skew = (62.5 ns * 100 – 6.5 µs)/ 62.5 ns = -4 periods.

The recommended setting of the modulation parameters (for a given CLKPLL) with the resultant min/max frequency and phase skew is discussed in greater detail in the hardware manual.

If the USART is used in the asynchronous mode with the clock modulator is switched on, then it would affect the baud rate of the USART and introduce deviation in the baud rate. In some cases, there may be deviation in the baud rate which may not be acceptable.

Example:

Let's consider the case where the required Modulated PLL Clock - CLKMOD is 16 MHz with the desired baud rate for the USART in asynchronous mode is 2 MBPS.

Here the `BGRn` register needs to be configured with the value of `0x0007`. So the deviation in the baud rate as per USART chapter in the hardware manual is 0. But the CLKMOD would also have an impact on this deviation of baud rate.

Lets also consider that the character format is 8E1 that means one USART character contains 11 bits (1 Start bit + 8 Data bits + 1 Even Parity bit + 1 Stop bit).

If the data that needs to be transmitted is `0x00`, the parity bit would also be zero and the USART output should remain at low level for period of 10 bits. These 10 bits corresponds to the 80 clock periods (cycles) of CLKMOD.

Let's also consider that the clock modulator is configured for the resolution of 11 and modulation degree of 1 then the "± phase skew min/max" would be 7.875. This means that the 80 clock periods would be either shortened to 72.125 clock periods or may get lengthened to 87.875 clock period that is the error of up to 1 bit. This may cause parity error or framing error respectively at the receiver.

3 Configuration

The configuration of the clock tree is controlled by the following registers:

- Clock Selection Register (`CKSR`)
- Clock Monitor Register (`CKMR`)
- Clock Stabilization Select Register (`CKSSR`)
- Clock Frequency Control Register (`CKFCR`)
- PLL Control Register (`PLLCR`)

3.1 Using the *start.asm* file

The correct setting of these registers can be performed easily by using the Cypress provided file *start.asm*.

The *start.asm* can be configured to wait for a stabilized Main or PLL clock before calling the `main()` function. To enable this function set `CLOCKWAIT` to `ON`. If you want to start your application as soon as possible, even without the stabilized clock, you can also set `CLOCKWAIT` to `OFF`.

```
;===============================================================
; 4.8  Clock Selection
;===============================================================


#set      CLOCKWAIT      ON     ; <<< wait for stabilized clock, if
                                ;     Main Clock or PLL is used
```

However, it should be noted that the CLOCKWAIT setting is only valid if the CLOCK_SPEED is configured to any of the following values:

```
CPU_4MHZ_MAIN_CLKP2_4MHZ
CPU_4MHZ_PLL_CLKP2_4MHZ
CPU_8MHZ_CLKP2_8MHZ
CPU_48MHZ_CLKP2_16MHZ
CPU_56MHZ_CLKP2_14MHZ
```

For all the other values of CLOCK_SPEED, the setting of CLOCKWAIT is ignored and the application would be only started once the PLL gets stabilized.

Also note that the clock modulator is still under evaluation. Hence, it is not offered in the *start.asm*. As most applications use the Sub Clock only for power saving reasons but not at start-up time, it is not offered here.

Set the parameter CRYSTAL to FREQ_4MHZ, FREQ_8MHZ, or FREQ_16MHZ depending upon the actual Main (X0/X1) crystal frequency.

```
#set      FREQ_4MHZ       D'4000000L
#set      FREQ_8MHZ       D'8000000L
#set      FREQ_16MHZ      D'16000000L

#set      CRYSTAL       FREQ_4MHZ  ;<<< select external crystal frequency

#set      CPU_4MHZ_MAIN_CLKP2_4MHZ            0x00000000L
#set      CPU_4MHZ_PLL_CLKP2_4MHZ             0x04040404L
#set      CPU_8MHZ_CLKP2_8MHZ                 0x08080808L
#set      CPU_12MHZ_CLKP2_12MHZ               0x0C0C0C0CL
#set      CPU_16MHZ_CLKP2_16MHZ               0x10101010L
#set      CPU_24MHZ_CLKP2_12MHZ               0x18180C18L
#set      CPU_24MHZ_CLKP2_16MHZ               0x18181018L
#set      CPU_24MHZ_CLKP2_24MHZ               0x18181818L
#set      CPU_32MHZ_CLKP2_16MHZ               0x20201020L
#set      CPU_32MHZ_CLKP1_16MHZ_CLKP2_16MHZ   0x20101020L
#set      CPU_40MHZ_CLKP2_16MHZ               0x28281028L
#set      CPU_48MHZ_CLKP2_16MHZ               0x30301030L
#set      CPU_48MHZ_CLKP1_32MHZ_CLKP2_16MHZ   0x30201030L
#set      CPU_48MHZ_CLKP1_16MHZ_CLKP2_16MHZ   0x30101030L
#set      CPU_48MHZ_CLKP1_16MHZ_CLKP2_CLKP3_MAIN  0x30100030L
#set      CPU_56MHZ_CLKP2_8MHZ                0x38380838L
#set      CPU_56MHZ_CLKP2_14MHZ               0x38380E38L

#set      CLOCK_SPEED     CPU_56MHZ_CLKP2_14MHZ    ; <<< set clock speeds
```

Set the parameter CLOCK_SPEED to the desired value depending upon the required CLKB, CLKP1 and CLKP2 frequency.

The peripheral clock CLKP1 is set to the same frequency than the CPU (CLKB). The peripheral clock CLKP2 has its own setting. This is because it feeds only the CAN controllers and Sound Generators. These do not need high frequency clocks.

The system clock has two optimization options: Wait state optimization and low PLL clock optimization:

```
#set      WAIT_STATE_OPTIMIZATION  2
#set      LOW_PLL_OPTIMIZATION     1

#set      CLOCK_OPTIMIZATION       WAIT_STATE_OPTIMIZATION ; <<< set clock
                                                             ; optimization mode
```

The first option uses a Flash timing setting with as less as possible wait states, which can mean, that a high PLL frequency is used, which is divided then for CLKS1 and CLKS2.

The second option uses a low PLL frequency which causes less power consumption, but may use more wait states for the Flash timing.

The Main clock stabilization time can also be configured as shown below.

```
;=====================================================================
; 4.9  Clock Stabilization Time
;=====================================================================

#set       MC_2_10_CYCLES   0
#set       MC_2_12_CYCLES   1
#set       MC_2_13_CYCLES   2
#set       MC_2_14_CYCLES   3
#set       MC_2_15_CYCLES   4
#set       MC_2_16_CYCLES   5
#set       MC_2_17_CYCLES   6
#set       MC_2_18_CYCLES   7

#set       MC_STAB_TIME    MC_2_15_CYCLES  ; <<< select Main Clock Stabilization
Time
```

3.2 Caution on Flash Timing

Please be aware that the start code uses register settings for e. g. Flash timing, PLL factor, etc. *only* for the used selection.

If the programmer wants to change the clock source for CLKS1 or CLKS2 such as RC clock or sub clock in his application, he has to check, if every setting is feasible for this clock mode. Especially the Flash timing settings (MTCRA, MTCRB) has to be checked, because there are possible Flash timings, which do not allow frequencies of 32 kHz, 100 kHz, or 2 MHz for CLKS1 and/or CLKS2.

Please see example code in 4.4.

3.3 Background debugging in other clock modes than PLL clock mode

Please be aware that the start code also sets the baud rate for background debugging depending on the clock settings. The user has to consider, that background debugging is not possible when the MCU is in a different clock mode or the clock settings are changed in the application.

3.4 Manual Clock Selection

The clocks can also be configured without the help of the *start.asm* file. Then, the different registers must be set correspondingly. This is detailed in the following.

3.4.1 RC Clock Configuration

The RC Clock is enabled by setting the CKSR:RCE=1. This is the default setting. Also the system clocks CLKS1 and CLKS2, by default, are set to RC Clock (i.e. CKSR:SC1S=0 and CKSR:SC2S=0).

The RC Clock is ready to use if CKMR:RCM=1. The transition to the Main Clock for the system clocks is completed, if CKMR:SC1M=0 or CKMR:SC2M=0, respectively.

3.4.2 Main Clock Configuration

The Main Clock needs a stabilization time. This can be selected in register CKSSR by the bits MCST2-0 according to the following table:

Table 1. Main Clock Stabilization Time

MCST2	MCST1	MCST0	Main Clock Stabilization Time (The corresponding time for a Main Clock of 4 MHz is given in parentheses.)
0	0	0	2^{10}/CLKMC (approx. 256 µs)
0	0	1	2^{12}/CLKMC (approx. 1 ms)
0	1	0	2^{13}/CLKMC (approx. 2 ms)
0	1	1	2^{14}/CLKMC (approx. 4 ms)
1	0	0	2^{15}/CLKMC (approx. 8 ms)
1	0	1	2^{16}/CLKMC (approx. 16 ms)
1	1	0	2^{17}/CLKMC (approx. 32 ms)
1	1	1	2^{18}/CLKMC (approx. 65 ms)

The highest stabilization time is selected by default. It may be decreased depending on the external circuitry.

The Main Clock is enabled by setting the `CKSR:MCE`=1. This is the default setting. Then, the desired system clock must be set to Main Clock. For the system clock 1, this is done by setting `CKSR:SC1S`=1. For the system clock 2, this is done by setting `CKSR:SC2S`=1.

The Main Clock is ready to use if `CKMR:MCM`=1. The transition to the Main Clock for the system clocks is completed, if `CKMR:SC1M`=1 or `CKMR:SC2M`=1, respectively.

3.4.3 PLL Clock Configuration

The PLL is disabled by default, because it must be configured correctly before enabling it. The configuration of the PLL is done through the `PLLCR` register. This register selects the multiplier of the PLL. Please refer to the following table to select the correct setting at the CLKMC of 4MHz:

Table 2. PLL Configuration for CLKMC = 4MHz

PLL Multiplier	PLLCR setting	CLKPLL		PLL Multiplier	PLLCR setting	CLKPLL
1	0x00E0	4 MHz		14	0x000D	56 MHz
2	0x00A1	8 MHz		15	0x000E	60 MHz
3	0x0062	12 MHz		16	0x000F	64 MHz
4	0x0043	16 MHz		17	0x0010	68 MHz
5	0x0044	20 MHz		18	0x0011	72 MHz
6	0x0025	24 MHz		19	0x0012	76 MHz
7	0x0026	28 MHz		20	0x0013	80 MHz
8	0x0027	32 MHz		21	0x0014	84 MHz
9	0x0008	36 MHz		22	0x0015	88 MHz
10	0x0009	40 MHz		23	0x0016	92 MHz
11	0x000A	44 MHz		24	0x0017	96 MHz
12	0x000B	48 MHz		25	0x0018	100 MHz
13	0x000C	52 MHz				

It should be noted that irrespective of the CLKMC and desired CLKPLL, the VCO output frequency of PLL (CLKVCO) should be always between 50 to 200 MHz and this is being taken care while deriving `PLLCR` setting in above Table 2 as well as below tables Table 3 and Table 4.

The following table describes the possible `PLLCR` setting with reference to the PLL multiplier at the CLKMC of 8 MHz:

Table 3. PLL Configuration for CLKMC = 8 MHz

PLL Multiplier	PLLCR setting	CLKPLL
1	0x00E0	8 MHz
2	0x00A1	16 MHz
3	0x0062	24 MHz
4	0x0043	32 MHz
5	0x0024	40 MHz
6	0x0025	48 MHz
7	0x0006	56 MHz
8	0x0007	64 MHz
9	0x0008	72 MHz
10	0x0009	80 MHz
11	0x000A	88 MHz
12	0x000B	96 MHz

The following table describes the possible PLLCR setting with reference to the PLL multiplier at the CLKMC of 16 MHz:

Table 4. PLL Configuration for CLKMC = 16 MHz

PLL Multiplier	PLLCR setting	CLKPLL
1	0x0020	16 MHz
2	0x0021	32 MHz
3	0x0022	48 MHz
4	0x0003	64 MHz
5	0x0004	80 MHz
6	0x0005	96 MHz

The PLL stabilization time can be selected between two settings as shown below:

Table 5. PLL Clock Stabilization Time

CKSSR:PCST	PLL Clock stabilization time (The corresponding time for a Main Clock of 4 MHz is given in parentheses.)
0	2^{12}/CLKMC (approx. 1 ms)
1	2^{14}/CLKMC (approx. 4 ms)

Once the correct configuration is set, the PLL can be enabled by setting CKSR:PCE=1. When the PLL stabilization time has passed, the monitor bit CKMR:PCM is set. Then, the system clocks can be configured to use the PLL by setting CKSR:SC1S=2 or CKSR:SC2S=2, respectively. The transition to the PLL Clock for the system clocks is completed, if CKMR:SC1M=2 or CKMR:SC2M=2, respectively.

3.4.4 Sub Clock Configuration

The Sub Clock needs a stabilization time. This can be selected in register CKSSR by the bits SCST1–0 according to the following table:

Table 6. Sub Clock Stabilization Time

SCST1	SCST0	Sub Clock Stabilization Time (The corresponding time for a Sub Clock of 32.768 kHz is given in parentheses.)
0	0	2^{12}/CLKSC (approx. 125 ms)
0	1	2^{14}/CLKSC (approx. 0.5 s)
1	0	2^{15}/CLKSC (approx. 1 s)
1	1	2^{16}/CLKSC (approx. 2 s)

The highest stabilization time is selected by default. It may be decreased depending on the external circuitry.

The Sub Clock is enabled by setting the `CKSR:SCE`=1. This is the default setting. Then, the desired system clock must be set to Sub Clock. For the system clock 1, this is done by setting `CKSR:SC1S`=3. For the system clock 2, this is done by setting `CKSR:SC2S`=3.

The Sub Clock is ready to use if `CKMR:SCM`=1. The transition to the Sub Clock for the system clocks is completed, if `CKMR:SC1M`=3 or `CKMR:SC2M`=3, respectively.

3.4.5 Clock Divider Configuration

The divider for the clocks - Bus Clock CLKB, Peripheral Clock 1 CLKP1, and Peripheral Clock 2 CLKP2 can be selected individually. This is done through the `CKFCR` register, with the bits `BCD3-0`, `PC1D3-0` and `PC2D3-0` respectively. The Bus Clock CLKB and the Peripheral Clock 1 CLKP1 are derived from the System Clock 1 CLKS1. The Peripheral Clock 2 CLKP2 is derived from System Clock 2 CLKS2. Dividers from 1 to 16 are available. Please use the settings shown in the following table for the desired setting.

Table 7. Clock Divider Configuration

CKFCR:BCD / CKFCR:PC1D / CKFCR:PC2D				Clock Divider Ratio		CKFCR:BCD / CKFCR:PC1D / CKFCR:PC2D				Clock Divider Ratio
3	2	1	0			3	2	1	0	
0	0	0	0	1		1	0	0	0	9
0	0	0	1	2		1	0	0	1	10
0	0	1	0	3		1	0	1	0	11
0	0	1	1	4		1	0	1	1	12
0	1	0	0	5		1	1	0	0	13
0	1	0	1	6		1	1	0	1	14
0	1	1	0	7		1	1	1	0	15
0	1	1	1	8		1	1	1	1	16

The RC clock frequency can also be selected using the `RCFS` bit of `CKFCR` register. If this bit is set to 0 then the typical RC clock frequency is set to 100 kHz and if the same is set to 1 then the typical RC clock frequency is set to 2 MHz.

3.4.6 Constraints

- Please note that the maximum frequencies specified in the datasheet must not be exceeded. Also the Flash memory read / write timing (using `MTCA` or `MTCB` register) and operating voltage (using `VRCR:HPM` and `MCSA:RD19V` / `MCSB:RD19V`) must be configured correctly. A setting that always works (although it is slow), is `MTCA` = `0x6E3D` for Main Flash A and `MTCB` = `0x6E3D` for Main Flash B.
- Please calculate the baud rate for background debugging, if it should be used, and set the Flash marker correspondingly. Note that for some devices a different calculation has to be used.

3.5 Clock Modulator Configuration

The Clock Modulator has the start up time of 6 µs after it is powered on (using the PDX bit of Clock Modulation Control Register (CMCR) register). The following are the prerequisites to enable the clock modulator (using the MODEN bit of CMCR register):

- The PLL lock time is elapsed and it is stabilized (i.e. PCM bit of CKMR register is set)
- Clock Modulator is powered up and the start up time is elapsed.
- The Clock Modulator Parameter Register (CMPR) register is configured with the appropriate value.

The MODRUN bit of the CMCR register reflects the status of modulated clock. If it is 1 then the CLKMOD can be used as a clock resource for CLKS1.

The CMPR register contains modulation parameter which determines the degree of modulation and the maximal and minimal occurring frequencies in the modulated clock. Please refer the hardware manual for the correspondence between PLL frequency and possible modulation parameters.

In order to power down the clock modulator its needs to be disabled (using the MODEN bit of CMCR register). After that, once the MODRUN bit gets cleared to zero, the clock modulator can be powered down.

4 Clocks Example

Example for Clocks

4.1 Manual Clock Setting

The following example demonstrates how to configure the CLKB and CLKP1 to 24 MHz frequency and the CLKP2 to 12 MHz when using 4 MHz of CLKMC (without using the clock configuration provided by *start.asm* file).

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.          */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES.                                              */
/*----------------------------------------------------------------------------*/

      . . .


  PLLCR = 0x25;            // set PLL clock to 24 MHz
  CKFCR_PC2D = 2;          // CLKP2 is CLKS2 divided by 2

  while (1 != CKMR_PCM);   // wait till PLL is ready

  CKSR_SC1S = 2;     // Set the CLKS1 to CLKPLL
  while (2 != CKMR_SC1M);  // wait till CLKS1 is switched to CLKPLL

  CKSR_SC1S = 2;           // Set the CLKS2 to CLKPLL
  while (2 != CKMR_SC2M);  // wait till CLKS2 is switched to CLKPLL


             . . .
```

Here it should be noted that the Flash memory (read) timing settings should also be configured accordingly. That is out of scope of this document.

4.2 Starting Clock Modulator

The following example demonstrates how to start the clock modulator with the PLL reference frequency (CLKPLL) of 24 MHz. The modulation resolution is chosen as 3 and the modulation degrees as 2. Hence the min frequency of modulated clock (CLKMOD) is 21.33 MHz and max frequency is 27.43 MHz.

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.          */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES.                                              */
/*----------------------------------------------------------------------------*/

      . . .


  PLLCR = 0x25;            // set PLL clock to 24 MHz

  CMCR_PDX = 1;      // power up clock modulator

  while (1 != CKMR_PCM);   // wait till PLL is ready this will also make up for
       // the clock modulator's startup time of 6 µs
  CMPR = 0x046E;     // resolution - 3, modulation degree - 2
             // min freq - 21.33 MHz, max freq - 27.43 MHz
  CMCR_MODEN = 1;          // modulation enable

  while (1 != CMCR_MODRUN);       // wait till modulator calibration done


      . . .
```

4.3 Stopping Clock Modulator

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.          */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES.                                              */
/*----------------------------------------------------------------------------*/

      . . .


  CMCR_MODEN = 0;                // modulation disable

  while (0 != CMCR_MODRUN);       // wait till modulator is disabled

  CMCR_PDX = 0;            // power down clock modulator


      . . .
```

The following example demonstrates how to stop the clock modulator.

4.4 Transition to RC clock or sub clock mode and back to PLL clock mode

In the following example code a transition to the sub clock mode is performed and a transition back to PLL clock mode. The example considers the Flash timing and sets save `MTCRA` settings. Also the CLKB divider is considered for sub clock mode and is restored in PLL clock mode.

Note that this method also works for RC clock mode.

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.          */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES.                                              */
/*----------------------------------------------------------------------------*/

      . . .

  unsigned long wait;
  unsigned int  MTCRA_save;
  unsigned int  CKFCR_save;

      . . .

  MTCRA_save = MTCRA;             // redo for Flash B if available
  CKFCR_save = CKFCR;


  // Switching between Sub clock and PLL clock mode
  while(1)
  {
    // Sub Clock Mode
    // ---------------------------------------------------------------
    MTCRA = 0x6E3D;         // save setting 5 wait states

    CKSR_SCE = 1;           // Sub clock enable
    while (CKMR_SCM == 0);  // Waiting for SUB clock is ready

    CKSR=0x8B;              // Transition to sub clock mode CLKS1
       while((CKMR_SC1M0 != 1) || (CKMR_SC1M1 != 1)); // Wait for transition
       //   finished

    CKSR=0x8F;              // Sub clock for CLKS2
       while((CKMR_SC2M0 != 1) || (CKMR_SC2M1 != 1)); // Wait for sub clock ready

       MTCRA = 0x2129;            // one wait state for sub clock mode
       CKFCR &= 0xFF0F;           // Set CLKB divider to 1

    // just do something in sub clock mode here
       . . .

    // PLL clock Mode
    // ---------------------------------------------------------------
    MTCRA = 0x6E3D;         // save setting 5 wait states

    CKSR_MCE = 1;           // Main oscillator enable
    while (CKMR_MCM == 0);  // Waiting for main oscillator is ready

    CKSR_PCE = 1;           // PLL clock enable
    while (CKMR_PCM == 0);  // Waiting for SUB clock is ready

    CKFCR = CKFCR_save;            // restore original clock dividers
```

▼

```
CKSR=0xCE;              // Transition to PLL clock mode CLKS1
while((CKMR_SC1M0 != 0) || (CKMR_SC1M1 != 1));    // Wait for transition
   //   finished

CKSR=0xCA;              // PLL clock for CLKS2

while((CKMR_SC2M0 != 0) || (CKMR_SC2M1 != 1));    // Wait for PLL clock ready

MTCRA = MTCRA_save;           // restore original PLL Flash timing

// just do something in PLL clock mode here
   . . .
     }
```

5 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

6 Document History

Document Title: AN204774 - F²MC-16FX Family, Clocks

Document Number: 002-04774

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	09/01/2006	Initial Release
			07/13/2007	Reviewed the document and updated with review findings
			08/29/2007	Updated usage of start.asm, PLL Clock configuration and added constraints
			03/22/2010	Add more clock settings
			04/22/2011	Flash Timing recommendations added; sub clock transition example added
*A	5060529	MKEA	12/22/2015	Migrated Spansion Application Note from MCU-AN-300225-E-V14 to Cypress format
*B	5836303	AESATMP8	07/28/2017	Updated logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2006-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.