PIC32MX Oscillator Library Documentation

Revision 1.0

Luka Gacnik

November 2022

Table of contents

Table of figures

Acronyms

PLL Phase-Locked Loop

CPU Central Processing Unit

SFR Special Function Register

RC Resistor Capacitor network

MCU Microcontroller Unit

PIO Programmable Input Output

1 Library description

The library consists of basic functions that control the oscillator module of the PIC32MX series Microchip microcontrollers.

1.1 About the oscillator

An oscillator module generates clock signal which is required for a device to execute instructions and for the peripherals to function. Such a module may consist of a set of registers that control various clocking settings such as oscillator source selection, clock frequency division factors, PLL multiplier, clock source switching, fine frequency tuning, and more.

The PIC32MX oscillator module also provides separate control over system clock which is used for CPU clocking and control over peripheral frequency that is distributed to all peripheral modules (this frequency cannot be controlled for each peripheral module separately since individual modules don't have corresponding oscillator related registers). Additionally, low power low speed oscillator always provides a separate clock signal for peripherals as the Watchdog timer and Fail-safe clock monitor to always function in the background (if enabled). This oscillator is also suitable for low power consumption sleep modes of peripheral modules.

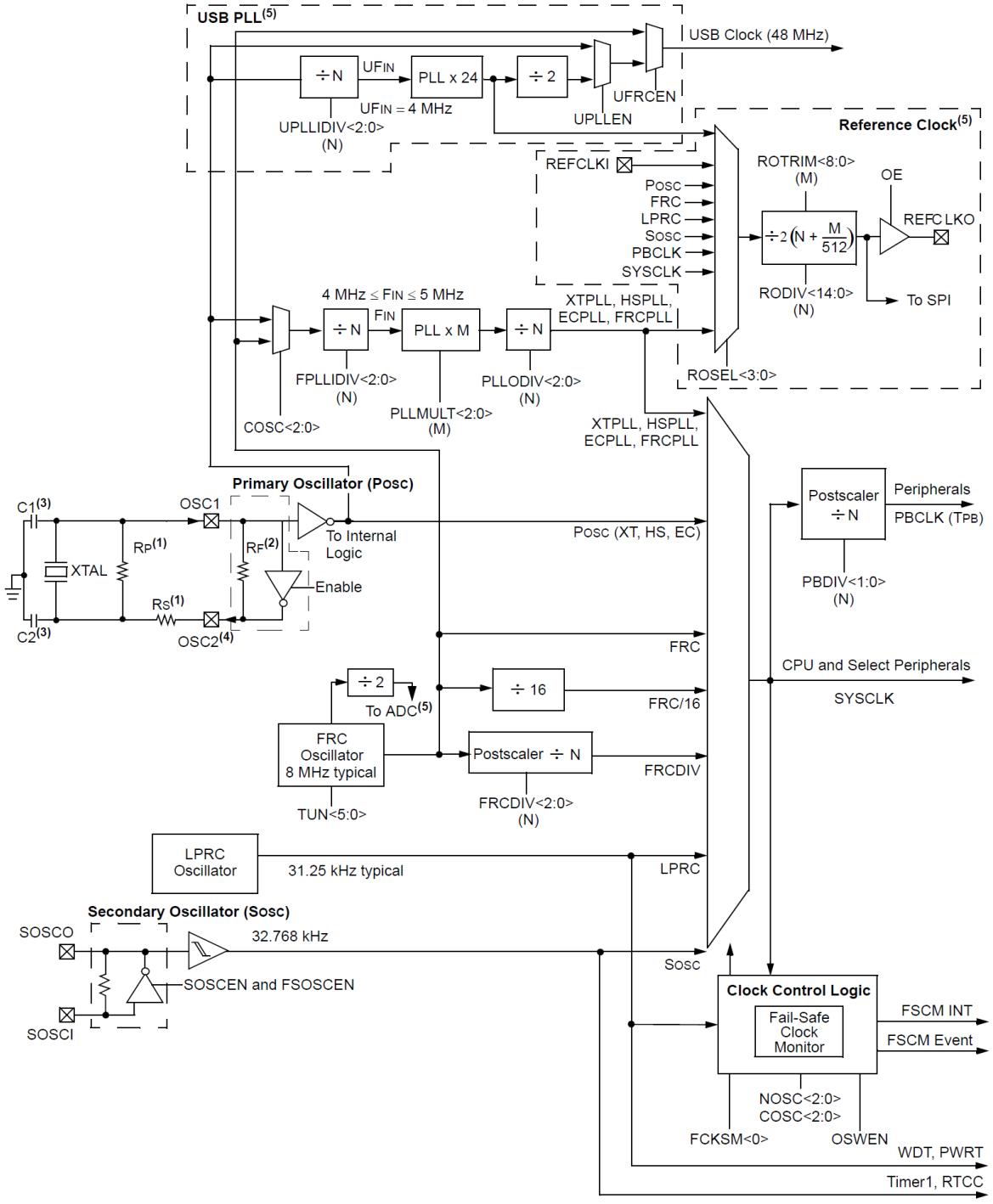

Figure 1.1: PIC32MX oscillator module block diagram

1.2 Library features

Currently, the Oscillator driver is capable of performing the following operations:

- Configuring system (and peripheral) clock frequency with source selection option
- Reading system (or peripheral) clock frequency currently configured

2 Dependencies

The library consists of the following source and header files:

- `Osc.c`
- `Osc.h`
- `Osc_sfr.h`

All the external user-accessible function prototypes are provided in the `Osc.h` file. Their definitions and other local functions are contained in the `Osc.c` file. The file `Osc_sfr.h` contains all the macro definitions for register bit-fields and SFR memory layout type.

3 Notes and warnings

This library depends on the library `Cfg.h` which provides the functions for (un)locking the Device Configuration register set of the PIC32MX microcontroller.

a) **User-defined oscillator macros**

In the `Osc.h` file there are few macro definitions which need to be carefully inspected prior to using the oscillator module of the PIC32MX device. The macro `OSC_XTAL_FREQ` defines the frequency of external oscillator (crystal oscillator, RC oscillator, external clock) and needs to be set prior to using this library if external source is used for primary oscillator. Additionally, the following macros must be changes if required:

- `OSC_FRC_FREQ` - the frequency of internal fast RC oscillator of a PIC32MX device
- `OSC_LPRC_FREQ` - the frequency of internal low power RC oscillator of a PIC32MX device
- `OSC_SOSC_FREQ` - the frequency of external secondary oscillator (`SOSC`) of a PIC32MX device (the SOSC frequency must be the same as specified by the device)
- `OSC_SYSCLK_MAX` - the maximum frequency of any clock source whose clock signal may be still used

The constants defined by these values should only be changed if any other PIC32 series is used (however other library functionalities may not be compatible with it).

b) **Critical configuration bits**

During the device programming the configuration bits of a specific device (PIC32MX in this case) must be programmed (and cannot be modified during user-defined program execution). There are few important such bit-fields which need to be set to required values in order for the Oscillator library to properly function. These important bit-fields:

- `FPLLIDIV` (PLL Input Divider bits from DEVCFG2 register) - must be set to the value `0x1` if the internal `FRC` clock source is used in conjunction with the PLL (`FRCPLL` source). If a PLL is to be used with an external source (`POSCPLL` source) then the `FPLLIDIV` should be manually determined by the equation (applies for the PIC32MX devices):

$$4MHz \leq \frac{F_{source}}{FPLLIDIV} \leq 5MHz \tag{3.1}$$

  or F_{source} may need to be changed to meet the PLL requirements
- `FCKSM` (Clock Switching and Monitor Selection Configuration bits from DEVCFG1 register) - must be set so that the clock switching is enabled if clock switching will be performed
- `FNOSC` (Oscillator Selection bits from DEVCFG1 register) - must be set to proper value so that the desired oscillator source is selected

Note that the device configuration bits may be modified via `#pragma` directive for the Microchip XC32 compiler for the PIC32MX microcontroller.

4 Future ideas

None.

5 Custom types

The library comes with a set of custom made structure or enumeration types. For the sake of convenience these types are explained throughout this chapter. Note that individual enumeration types are not covered here because they (and their set of values) follow self-explanatory naming conventions.

5.1 OscConfig_t

This structure-based type contains the properties for configuration of the oscillator module. The type consists of the following properties:

- **oscSource** (*OscClkSource_t*) - an oscillator source to be selected for generating the clock signal
- **sysFreq** (*uint32_t*) - the system frequency of a PIC32MX device
- **pbFreq** (*uint32_t*) - the peripheral frequency of a PIC32MX device

6 Function description

The library includes the following functions:

- `OSC_GetSysFreq()`
- `OSC_GetPbFreq()`
- `OSC_GetClkSource()`
- `OSC_ConfigOsc()`

The following sub-sections describe the use and operationality of these functions.

6.1 OSC_ConfigOsc()

PROTOTYPE

`OSC_ConfigOsc(oscConfig)`

DESCRIPTION

Configures the oscillator registers to generate a system and peripheral frequency from the selected oscillator source.

PARAMETERS

- **`oscConfig`** (*`OscConfig_t`*) - contains the settings which determine the basic configuration of the oscillator module

RETURNS

Returns `false` is system clock frequency is set to zero (or not configured), otherwise it returns `true`.

OPERATION

Initially checks whether the user-selected frequency is more than the `OSC_SYSCLK_MAX` frequency. Then the calculation of the divisor and multiplier factors (if any) is carried out by calling either of the following private functions: `GetPllMultDiv()`, `GetFrcDiv()`, `GetPbDiv()`, or none if the fixed frequency of the source matches the desired frequency. To configure oscillator control register it needs to be unlocked by the function `CFG_UnlockSystemAccess()` (and later locked).

If clock switch is required (new source is different than the old one) the source is first switched to `FRC`, then registers are modified, and lastly clock is switched to the new source. If it is not required (current source is one of the PLL sources and remains unchanged) then only PLL registers are modified.

Lastly, the peripheral clock division bits are written to. And secondary oscillator (`SOSC` source) is enabled if required.

NOTES

- System (and peripheral) clock frequency will be configured to the closest possible value in case the exact user-selected frequency cannot be configured
- If the peripheral clock frequency equals zero (or not configured) it defaults to the value of system clock frequency
- It is possible to set a secondary oscillator as a (new) source however an external component is physically not connected to the device. In the case of enabling the `SOSC` source this function waits for the oscillator source to become stable. If the external component is not connected to the device then this function will cause a deadlock in an infinite loop
- Flexible yet simple configuration of oscillator module allows the user to configure PLL registers without manually setting the multiplier and divisor factors. The downside of this benefit is that additional private functions are executed which perform moderate amount of calculations to find valid settings to be written to PLL registers

LIMITATIONS

- The maximum system clock frequency depends on the specific device and is defined by the `OSC_SYSCLK_MAX` macro
- The maximum peripheral clock frequency is limited by the system clock frequency and the minimum is the system clock frequency divided by 8

6.2 OSC_GetSysFreq()

PROTOTYPE

`OSC_GetSysFreq()`

DESCRIPTION

Reads the value of the system clock frequency.

PARAMETERS

None.

RETURNS

An unsigned number of the type *uint32_t*.

OPERATION

First, a set of SFR reads is performed to obtain information about the related oscillator module configurations. This is then followed by a number of conditional statements to determine parameters to the system clock frequency calculation based on the current oscillator module configuration.

NOTES

It's advisable to use this function and get the actual system clock frequency which was previously configured by the `OSC_ConfigOsc()` function. The user-selected frequency value and the value resulting from this function may differ.

LIMITATIONS

None.

6.3 OSC_GetPbFreq()

PROTOTYPE

`OSC_GetPbFreq()`

DESCRIPTION

Reads the value of the peripheral clock frequency.

PARAMETERS

None.

RETURNS

An unsigned number of the type *`uint32_t`*.

OPERATION

A value of the system clock frequency, obtained with the `OSC_GetSysFreq()` function, is divided by the peripheral clock frequency that was read from the oscillator module SFR.

NOTES

It's advisable to use this function and get the actual peripheral clock frequency which was previously configured by the `OSC_ConfigOsc()` function. The user-selected frequency value and the value resulting from this function may differ.

LIMITATIONS

None.

6.4 OSC_GetClkSource()

PROTOTYPE

`OSC_GetClkSource()`

DESCRIPTION

Reads the value of the oscillator module source selection.

PARAMETERS

None.

RETURNS

An enumeration value of the type *`OscClkSource_t`*.

OPERATION

Direct read of a SFR.

NOTES

Function is always inlined due to short length of it.

LIMITATIONS

None.

7 Examples

A set of examples is provided to demonstrate a simple operation of a Timer module. Examples provide only the main operation related code. Other processes such as the start-up of an MCU, programming of the device configuration register, setting up other peripheral modules, and similar is not provided here.

7.1 Oscillator clock switch

This example shows a simple example how Oscillator library functions may be used. The PIO library is optional for pin toggling. A measurement figure is provided in the appendix (A) to indicate the effects of this sample code.

Initially, primary oscillator settings are configured using the function `OSC_ConfigOsc()` and frequency read with the function `OSC_GetSysFreq()`. Toggling of `RPB4` pin takes place to indicate the change in system clock speed for the following changes in its frequency. Then the first re-configuration of the oscillator module is carried out with system clock frequency reduced and the pin is toggled again. Looking at the figure (A.1) the reduced toggling speed is evident. Oscillator module is again re-configured but this time switching to the low power RC oscillator where system clock frequency is reduced to 31,25 kHz. Not just toggling speed is reduced but also an additional delay from executing the function `OSC_ConfigOsc()` can be seen.

```
/** Custom libs **/
#include "Osc.h"
#include "Pio.h"

int main(int argc, char** argv)
{
   /* Oscillator initial configuration parameters */
   OscConfig_t oscConfig = {
      .oscSource = OSC_COSC_FRCPLL,
      .sysFreq = 20000000,
      .pbFreq = 20000000
   };

   uint32_t cntr, sysFreq1, sysFreq2, sysFreq3;

   /* Configure indicating pin */
   PIO_ClearPin(GPIO_RPB4);
   PIO_ConfigGpioPin(GPIO_RPB4, PIO_TYPE_DIGITAL, PIO_DIR_OUTPUT);

   /* Initial oscillator configuration */
   OSC_ConfigOsc(oscConfig);
   sysFreq1 = OSC_GetSysFreq();

   /* Toggle pin */
   cntr = 1000;
   while( cntr-- )
   {
      PIO_TogglePin(GPIO_RPB4);
   }

   PIO_ClearPin(GPIO_RPB4);

   /* Re-configure oscillator first time */
   oscConfig.sysFreq = 1000000;
   OSC_ConfigOsc(oscConfig);
   sysFreq2 = OSC_GetSysFreq();

   /* Toggle pin */
   cntr = 100;
   while( cntr-- )
   {
```

```
42          PIO_TogglePin(GPIO_RPB4);
43      }
44
45      /* Re-configure oscillator second time */
46      oscConfig.oscSource = OSC_COSC_LPRC;
47      OSC_ConfigOsc(oscConfig);
48      sysFreq3 = OSC_GetSysFreq();
49
50      /* Toggle pin */
51      cntr = 10;
52      while( cntr-- )
53      {
54          PIO_TogglePin(GPIO_RPB4);
55      }
56
57
58      while(1){}
59
60      return 0;
61  }
```

A Oscillator clock switch measurement

Here is the figure which results in executing code provided by the (7.1) sample code. At the right side panel of the time diagram a measurement of a single pin toggle at different system clock frequencies is shown. From the ratio of pin toggle periods is the same as the ratio of three different system clock frequencies. This indicates the system clock frequency was configured to the desired value for all three oscillator configurations.

Figure A.1: Oscillator clock switch time diagram

B Revision history

Revision 1.0 (November 2022)

This is the initial released version of this document.