AN14010

Frequency Measurement Module on LPC553x/S3x Devices

Rev. 1 — 23 August 2023 Application note

Document Information

Information	Content
Keywords	AN14010, FREQME, frequency, duty cycle
Abstract	This document describes how to use and configure the frequency measurement module on LPC553x/S3x devices.

1 Introduction

The frequency measurement (FREQME) block has been introduced for LPC553x devices. This simple peripheral block can solve specific software tasks by hardware and safe computation power for other application cases. This peripheral block is intended for frequency and pulse width measurement of internal and external signals.

This application note helps the user to understand this module and examples in this application note accelerate the feature evaluation.

2 Acronyms

Table 1 lists the acronyms used in this document.

Table 1. Acronyms

Acronym	Meaning
FREQME	Frequency measurement
PWM	Pulse width measurement
SDK	Software development kit
MCU	Microcontroller unit
REF_CLK_IN	Reference clock input
TARGET_CLK_IN	Target clock input

3 FREQME module

FREQME accurately measures the frequency of an on/off-chip target clock signal using a selectable on-chip reference clock. For example, it accurately determines the frequency of a low-power oscillator that varies depending on process and temperature.

The features of the FREQME module are as follows:

- High-accuracy frequency measurement mode for on-chip and off-chip clocks
- Pulse width measurement (PWM) mode
- Reference and target clock inputs, selectable from among various chip-specific options
- Optional measurement complete interrupt
- Result of out-of-range detection with optional interrupt

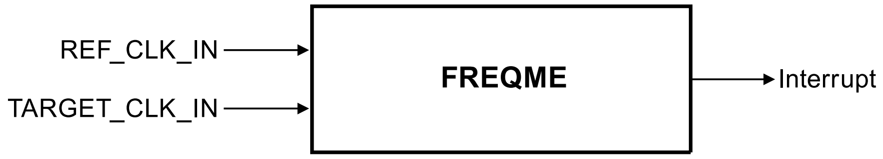

Figure 1. FREQME block

4 Configuration of register

FREQME is a basic module with five registers. The main register to configure is `CTRL_W`. Users can use the software development kit (SDK) driver or directly write to registers using predefined masks from the device header file. If there is required manual clearing of status flags without the SDK driver, then the user has to read the FREQMECTRLSTAT register first, then clear the required bits (for example, MEASURE_IN_PROGRESS_MASK, LT_MIN_STAT, GT_MAX_STAT), and write back to the CTRL_W register.

Examples for evaluation are created using the SDK driver. There is only one non-SDK function for setting up the reference scale at runtime.

4.1 Frequency measurement mode

In frequency measurement mode, the SDK configuration is set as shown in Equation 1:

$$config.operateMode = kFREQME\_FreqMeasurementMode \quad (1)$$

For this mode, FREQME counts the number of target clock cycles that occur during a specified number of cycles from a reference clock with a known frequency.

Calculation of target frequency is based on Equation 2:

$$Ftarget = (CTRL\_R[RESULT] - 2) \times Freference \div 2^{CTRLSTAT[REFSCALE]} \quad (2)$$

The Equation 2 is implemented in the SDK function FREQME_CalculateTargetClkFreq, which is used in the software example. 12 MHz oscillator is selected as reference clock and the main system clock is selected as target clock. By setting a higher reference scale number, the target frequency is measured for longer time to get higher and more precise result.

The SDK functions FREQME_SetMinExpectedValue and FREQME_SetMaxExpectedValue are used to configure the minimum and maximum registers. FREQME module trigger interrupt or set flag when the result register is out of predefined limits. For example, the flag must be checked in some periodic event or the background loop. In the example, an interrupt-based approach is used. For starting the measurement cycle, the SDK function FREQME_StartMeasurementCycle is used and it is used each time after getting the result. In the default configuration, the example must work after build.

4.2 Pulse width measurement mode

The SDK configuration sets the pulse width mode as shown in Equation 3:

$$config.operateMode = kFREQME\_PulseWidthMeasurementMode \quad (3)$$

For this mode, the reference scale parameter is ignored. FREQME module counts reference clock pulses while the target clock is in a specific state (high or low). Polarity of the measured signal can be changed in runtime using the FREQME_SetPulsePolarity function.

In the example, the eFlexPWM module is used as a reference clock. Pulse width can be set from the FreeMASTER. This signal must be externally routed to the specific GPIO pin. Main system clock is internally routed as the target clock of FREQME input.

5 Evaluation of the examples

For project evaluation, install the latest MCUXpresso IDE and FreeMASTER real-time debugger. Although some familiarity with FreeMASTER debugging is helpful, this project is so simple that even beginners must not encounter any difficulties.

In the example, define FREQ_MEAS, which switches the code between frequency or pulse width measurement.

For pulse width measurement, connect wire from PWM output pin to FREQME input pin as it is platform-dependent, see Table 2. For frequency measurement, wire connection is not required as everything is done internally.

All information provided in this document is subject to legal disclaimers.

Table 2. Hardware signals on the EVK

EVK board	Header	MCU port	MCU signal
LPCXpresso55S36	J10-15	1-20	PWM0 A0
LPCXpresso55S36	J10-5	1-4	FREQME CLKA

To import the project into the IDE, perform the steps as follows:

1. Open `main.c` source file and set `#define FREQ_MEAS` for required operation:
   - 1 for frequency measurement
   - 0 for pulse width measurement
2. Build and flash the project.
3. Start FreeMASTER and load `FREQMEASURE_evaluation.pmpx` project.
4. To run FreeMASTER communication, click the GO button.
5. Watch and set variables and observe the FreeMASTER scope.

Table 3 explains the variables used in the project. Read/write (R/W) attribute means that it is accessible from FreeMASTER.

Table 3. Variables

Variable	Description	R/W
`ui8PulsePol`	Pulse polarity `kFREQME_PulseLowPeriod` or `kFREQME_PulseHighPeriod` for pulse width measurement mode.	RW
`ui32FreqmeResultReg`	Raw measurement result	R
`ui32MeasFreq`	Calculated frequency	R
`ui32RefFreq`	Reference frequency	R
`ui32FreqMeasLoLim`	Minimum register value	RW
`ui32FreqMeasHiLim`	Maximum register value	RW
`ui32RefScale`	Scale for frequency measurement. If a higher number is set, more time is required to collect the result. In extreme cases, it takes up to minutes.	RW
`i16Duty`	PWM duty cycle source for pulse width measurement	RW
`ui32ResultOverflowCnt`	Incrementing number when overflow occurs	R
`ui32ResultUnderflowCnt`	Incrementing number when underflow occurs	R
`ui32ResultReadyCnt`	Incrementing number when the result is ready	R

5.1 Troubleshooting

The examples function in the default configuration; however, some behavior can occur, especially when editing the project. The main pointer that the application runs OK is that the numbers in FreeMASTER are live, especially `ui32ResultReadyCnt` must increment.

Table 4 lists the issues and provides the solution when the application is not working correctly.

Table 4. Troubleshooting

Issue	Solution
`ui32ResultReadyCnt` has no increment	In pulse width measurement mode, the wire is not connected on the EVK board from PWM output pin to FREQME input pin.

Table 4. Troubleshooting...*continued*

Issue	Solution
`ui32ResultReadyCnt` has no increment	Input mux to the FREQME module are not correct or source signals (clocks) are not enabled in the system.
`ui32ResultReadyCnt` has no increment	Scale set close to its limits (31) produces long waiting time until the results are ready (up to minutes). Use lower scale if you want to get the results faster.
`ui32MeasFreq` is 0	For pulse width mode, frequency calculation does not make sense so the result is 0 because `ui32RefFreq` is preset to 0. In frequency measurement mode, check why `ui32RefFreq` is not filled with reference clock value.
FreeMASTER cannot connect to the target	A debug session running in the MCUXpresso IDE is the most usual behavior. User must kill the debug session in the MCUXpresso IDE. It is not possible to run both debug sessions (MCUXpresso and FreeMASTER) because they access the same debug interface. It is recommended not to use the debug session for flashing instead use the flash button . To enable the flash button in the menu bar, perform the following steps: 1. User must select the project. ***Note:*** *User must select the project again as sometimes after the build, the project is unselected.* 2. User must kill the debug session in MCUXpresso when the conflict occurs with the debug and FreeMASTER session. 3. Unplug/plug EVK USB and restart FreeMASTER. 4. User must click the GO button to start communication.

6 References

Table 5 lists the resources that can be referred for more information.

Table 5. References

Documents/resources	Link/how to access
MCUXpresso Integrated Development Environment	MCUXpresso IDE
FreeMASTER Run-Time Debugging Tool	FreeMASTER

7 Revision history

Table 6 summarizes the revisions to this document.

Table 6. Revision history

Revision number	Release date	Description
1	23 August 2023	Initial public release

8 Legal information

8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

8.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com

Date of release: 23 August 2023
Document identifier: AN14010