

# Programming Reserved Register Bits for MPC55xx, MPC56xx, and MPC57xx Families

## Contents

1 Introduction.....	1
2 Programming reserved register bits...	1

by: NXP Semiconductors

## 1 Introduction

This engineering bulletin provides guidance on programming reserved register bits for the MPC55xx, MPC56xx, and MPC57xx families of devices.

## 2 Programming reserved register bits

Certain device register bits are marked as reserved and should not be modified. Since registers are always written completely and it is not possible to write code that only modifies specific bits, when writing to registers which contain one or more reserved bits, you must do one of the following:

1. Write back the reset value of a reserved bit as documented in the reference manual.

OR

2. Write back the value which has just been read from that bit of the register.

For example, consider the following register:



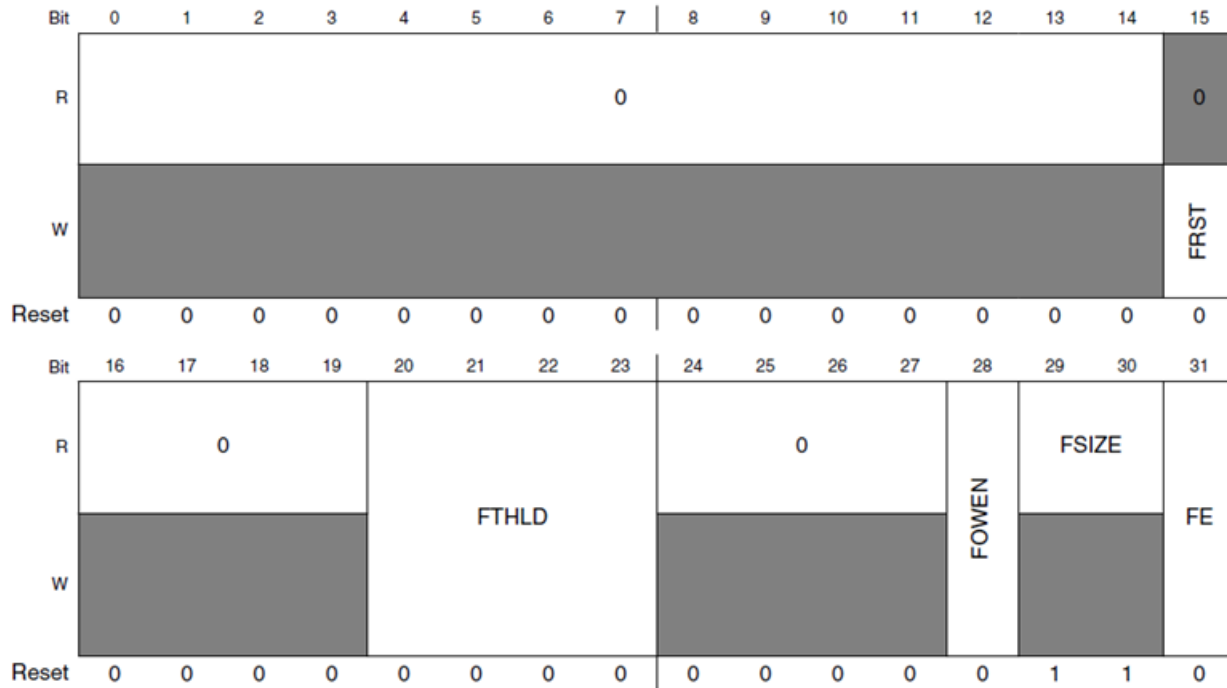


Figure 1. Register with reserved bits

To set FRST to 0x0, FTHLD to 0x4, FOWEN to 0x0, and FE 0x1, you can:

- Use option 1 and fill in the reserved bits with their reset value, thus writing 0x0000\_0407 to the register.  
C code representation: register.R = 0x00000407; // write reset value to reserved bits.
- Use option 2 and fill in the reserved bits with the value that has just been read. First, you read the value of the register, suppose it reads as 0x0000\_0006. Then, OR it with the bits you want to set, thus writing 0x0000\_0407 to the register.  
C code representation: register.R |= 0x00000401; // OR bits to be set with current value

While either of these two options work in most cases, you must always do as specified in the reference manual. The reference manual notes any register bits that require special handling.

Write 1 to clear (w1c) bits also require special attention. Some registers such as interrupt status registers might contain multiple w1c bits along with reserved bits. Writing back the value you just read from a set w1c bit will clear it. Therefore, you must write 0 to any w1c bits you don't intend to clear.

For example, consider the following register with four w1c status bits:

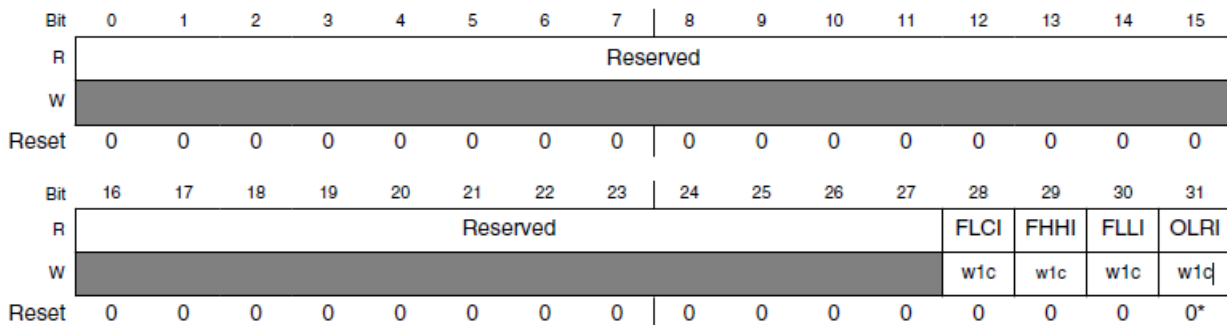


Figure 2. Register with four w1c status bits

Suppose the register currently holds the value 0x0000\_000F and you want to clear bit 30 (FLLI). To do so, you must write 0x0000\_0002. Writing this value sets the reserved bits to their reset value and preserve the flags you do not intend to clear.

## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP 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. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and  $\mu$ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2017 NXP B.V.

EB00855  
Rev. 0  
May 2017

