Freescale Semiconductor, Inc.
Application Note

Document Number: AN0
Rev. 0, 01/2016

Using the FCCU on the MPC5744P

Peter Vlna

1. Introduction

This document describes the configuration, restrictions, principles and correct usage of FCCU module implemented on MPC5744P device.

Contents

2. Important FCCU notes for MPC5744P device

Always have in mind as soon as you change FCCU state, the FCCU internal watchdog will start. After expiration of this watchdog the FCCU Operation run bit field in FCCU_CTRL register will change to ABORT. Therefore it is not possible to do “STEP” operations in debugger.

3. FCCU faults reading

The FCCU_NCF_Sx register contains the latched fault indication collected from the non-critical fault sources.

To read FCCU faults stored in NCF_Sx registers on MPC5744P device it is necessary to write to the FCCU control register (FCCU_CTRL).

1. Write proper OPR into the FCCU_CTRL[OPR] = 10.
2. Wait for the successful operation run (OPR) mode change
3. Read the faults from NCF_S[x] registers

Example code:

```
void FCCU_read_faults(void)
{
  /* To read faults OP10 must be set in FCCU_CFG[OPR] field */
  FCCU.CTRL.R = 0xA;  enter config state - OP10

  /* wait for the  operation run (OPR) mode change */
  while (FCCU.CTRL.B.OPS != 0x3);//operation status

  /* Read all NCFS registers to variables */
  FCCU_status[0] = FCCU.NCF_S[0].R;     //read FCCU.NCF_S0 register
  FCCU_status[1] = FCCU.NCF_S[1].R;     //read FCCU.NCF_S1 register
  FCCU_status[2] = FCCU.NCF_S[2].R;     //read FCCU.NCF_S2 register

}//FCCU_read_faults
```

4. FCCU faults clearing

To clear FCCU latched faults it is necessary to execute following procedure:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCF_Sx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

Example code:

```
void FCCU_clear_faults(void)
{
  /* 1. Write the proper key into the FCCU_NCFK register */
  //Non-critical fault key = AB34_98FEh
  FCCU.NCFK.R = 0xAB3498FE;

  /* 2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically
  /* Read all NCFS registers to clear all faults.*/
  /* For details which faults can be cleared see Table 7-36. FCCU Non-Critical Faults Mapping
in RM */
  FCCU.NCF_S[0].R = 0xFFFFFFFF;          // read FCCU.NCF_S0 register

  /* Verify if state change was successful */
  while (FCCU.CTRL.B.OPS != 0x3);//Operation status successful

  /* NCFS_1 register clear */
  FCCU.NCFK.R = 0xAB3498FE;              //Non-critical fault key = AB34_98FEh
  FCCU.NCF_S[1].R = 0xFFFFFFFF;          // clear FCCU.NCF_S1 register
  /* Verify if state change was successful */
  while (FCCU.CTRL.B.OPS != 0x3);//Operation status successful

  /* NCFS_2 register clear */
  FCCU.NCFK.R = 0xAB3498FE;              //Non-critical fault key = AB34_98FEh
  FCCU.NCF_S[2].R = 0xFFFFFFFF;          // clear FCCU.NCF_S2 register
  /* Verify if state change was sucessful */
  while (FCCU.CTRL.B.OPS != 0x3);//Operation status succesfull
}//FCCU_clear_faults
```

5. FCCU fault injection

Before attempting to inject a fault it is necessary to understand that not all faults can be injected. Some faults can be injected through peripherals like ADC or PMC, some cannot be injected at all. To distinguish between them user must refer to reference manual Table 7-37. FCCU Non-Critical Faults Mapping. The following example code represents fault injection but without any reaction set on injected fault.

Example code:

```
void FCCU_Fake_faults_inject(void)
{
  /* Inject a NCF[7] -> STCU2 fault */
  FCCU.NCFF.R = 0x7;
  /* Fault is now injected and user can read FCCU NCFSx register to see it is
     latched in NCFSx registers. But by default no reaction is set on this fault
     so it is only informative */
}//FCCU_Fake_faults_inject
```

6. CONFIG state

The default configuration can be modified only in the CONFIG state. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

Before entering configuration state of FCCU several steps need to be taken.

1. SWT must be disabled
2. FCCU pending faults latched in NCF_Sx registers must be cleared.

Entering the configuration state:

1. Unlock the configuration via providing Transition Unlocking Key value (0xBC)
2. Provide Control register key for CONFIG state (0x913756AF)
3. Enter CONFIG state
4. Verify if state transition was successful

Now FCCU is prepared for configuration defined by user.

Exiting configuration state:

1. Provide Control key for NORMAL state (0x825A132B)
2. Put FCCU into normal state (OP2)
3. Verify if state transition was successful

NOTE

FCCU implements an internal watchdog. If FCCU enters CONFIG state watchdog is started. If user is not able to configure the FCCU and return to NORMAL state in watchdog time window the CONFIG state is aborted and no changes are taken into account. As a result the OPS bit in CTRL register is set to abort value. Therefore it is not possible to debug FCCU while it is in CONFIG state.

Example code:

```
void FCCU_CONFIG (void)
{
  /* Unlock configuration */
  FCCU.TRANS_LOCK.R = 0xBC;
  /* provide Config state key */
  FCCU.CTRLK.R = 0x913756AF;            //key for OP1
  /* enter config state - OP1 */
  FCCU.CTRL.R = 0x1;                    //set OP1 - set up FCCU into the CONFIG mode
  /* wait for successful state transition */
  while (FCCU.CTRL.B.OPS != 0x3);//operation status succesful
```

```
  /*************************************/
  /* Insert here the FCCU configuration */
  /*************************************/

  /* set up the NOMAL mode of FCCU */
  FCCU.CTRLK.R = 0x825A132B;            //key for OP2
  FCCU.CTRL.R = 0x2;                    //set the OP2 - set up FCCU into the NORMAL mode
  while (FCCU.CTRL.B.OPS != 0x3);//operational status succesful

}//FCCU_CONFIG
```

7. ALARM state

FCCU state machine offers a possibility of entering into the ALARM state when fault is detected. User has a possibility to trigger an ALARM interrupt on FCCU fault event and solve the issue (fault source) in configured timeout window.

Before entering configuration state of FCCU several steps need to be taken.

3. SWT must be disabled
4. FCCU pending faults latched in NCF_Sx registers must be cleared.

Entering the configuration state:

5. Unlock the configuration via providing Transition Unlocking Key value (0xBC)
6. Provide Control register key for CONFIG state (0x913756AF)
7. Enter CONFIG state
8. Verify if state transition was successful

Now FCCU is prepared for configuration defined by user.

1. Enable ALARM timeout in NCF_TOEn register
2. Enable reaction on fault in NCF_En register
3. Enable alarm interrupt in IRQ_ALARM_ENn register

Exiting configuration state:

4. Provide Control key for NORMAL state (0x825A132B)
5. Put FCCU into normal state (OP2)
6. Verify if state transition was successful

NOTE

Make sure your INTC controller is configured properly and your ALARM interrupt has priority set.

Example code:

```
void FCCU_CONFIG (void)
{
  /* Unlock configuration */
  FCCU.TRANS_LOCK.R = 0xBC;

  /* provide Config state key */
  FCCU.CTRLK.R = 0x913756AF;            //key for OP1

  /* enter config state - OP1 */
```

```
  FCCU.CTRL.R = 0x1;                    //set OP1 - set up FCCU into the CONFIG
  /* wait for successful state transition */
  while (FCCU.CTRL.B.OPS != 0x3);       //operation status succesful

  /* FCCU moves into the ALARM state if the respective fault is enabled */
  FCCU.NCF_TOE[0].R = 0xFFFFFFFF;       //ALARM Timeout Enable

  FCCU.NCF_E[0].B.NCFE7 = 0x1;          //Enable reaction on fault NCF[7]

  FCCU.IRQ_ALARM_EN[0].R = 0x80;        //Enable ALARM interrupt on fault NCF[7]

  /* set up the NOMAL mode of FCCU */
  FCCU.CTRLK.R = 0x825A132B;            //key for OP2
  FCCU.CTRL.R = 0x2;                    //set the OP2 - set up FCCU into the NORMAL mode
  while (FCCU.CTRL.B.OPS != 0x3);//operational status succesful

}//FCCU_CONFIG
```

8. Revision History

The revision history is the last Heading 1 section of the document. A revision history section is used for all revisions. Rev. A documents (first NDA revision) and rev. 0 documents (first public revision) have placeholder revision history tables that say “Initial release.” When a document crosses from NDA alphanumeric numbering to publicly-released numeric numbering, a special document needs to be created to share the NDA rev-to-rev 0 changes with alpha customers. Contact your local documentation team for assistance.

Table 1 is an example of a revision history table.

Table 1. Sample revision history

Revision Number	Date	Substantive changes
0.1.0	1/2016	Initial Release
0.2.0	1/2016	Correction after review

Contact your technical documentation representative for more detailed instructions regarding the necessary components of the revision history table.

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.

Document Number: AN0
Rev. 0
01/2016