NXP Semiconductors
Application Note

Document Number: AN5427
Rev. 1.0.1, 10/2016

Using the Built-in Self-Test (BIST) on the MPC5746R

By: Peter Vlna and Petr Stancik

1. Introduction

The MPC5746R device targets industrial and automotive engine/transmission control applications that require advanced performance, timing systems and functional safety capabilities and ASIL-D Compliance. The ISO26262 standard defines functional safety for automotive equipment. A requirement of the standard is to detect the accumulation of latent defects. To meet this requirement the MPC5746R has the ability to execute Built-In Self-Test (BIST) procedures. The BIST is intended to identify latent system faults within the MCU.

The BIST can be performed on the device's embedded memories and logic.

Additionally, there is "SafeAssure" Functional Safety program to reduce the development effort required by customers to meet ISO26262. As part of this program NXP provides an MPC5746R safety manual to advice users on how to configure the MPC5746R to obtain ISO26262 ASIL D compliance.

Contents

2. Objective

This application note provides an introduction to BIST on the MPC5746R and explains how to configure and use the Off-line and On-line BIST features of the MPC5746R. After reading this applications note the user should:

- Understand the BIST features that are available on the MPC5746R
- Understand the difference between MBIST and LBIST
- Understand the difference between On-line and Off-line BIST
- Be able to develop application strategies for deploying On-line BIST testing
- Be able to develop application strategies for deploying Off-line BIST testing
- Understand what is included in each BIST partition on the MPC5746R
- Understand when Off-line and On-line BIST takes place
- Be able to develop a specified configuration for Off-line and On-line BIST testing and understand how to use DCF clients to achieve this
- Be able to invoke the desired Off-line and On-line BIST sequence using the STCU2

The application note also provides example for Off-line and On-line Self-Test configurations that can be directly implemented by the user. The example is provided in the software package that accompanies this document and is also explained in detail.

To aid in understanding of this document and software package the reader should obtain the MPC5746R Reference and Safety manuals from the NXP website.

3. Overview of Built-In Self-Test

The term Built-In Self-Test (BIST) is used to describe the on-chip hardware mechanisms that can be used to detect latent faults within the MCU. The BIST allows the MCU to conduct periodic self-tests to identify faults. The results of these self-tests can then be used by the MCU to handle the faults and ensure that the device remains in a safe state.

3.1. LBIST and MBIST

There are two different types of BIST implemented on MPC5746R devices:

- MBIST (Memory Build-in-self-test) – for memory testing purposes
- LBIST (Logic Build-in-self-test) – for logical testing purposes

MBIST is implemented for each of the SRAM and peripheral memories on the MCU. Like SRAM memory contained in the peripheral modules such as FlexRay or eDMA. For MBIST testing purposes each of the memories is segmented into individual MBIST partitions. The segmentation of the memories is discussed in MBIST partitions.

Each memory is broken down into multiple partitions providing flexibility to test selected address ranges only.

LBIST tests operate on the digital logic of the device and use scan test techniques to provide high coverage defect detection. The logic is divided up into multiple partitions, with each partition containing user recognizable logic modules (CPU, XBAR, eDMA, etc.). LBIST can be configured to test partitions sequentially or in parallel to allow a good combination of power consumption vs. execution time.

3.2. Off-Line and On-line BIST overview

It is anticipated that the MPC5746R automotive user application would require two standard configurations for BIST testing.

- Vehicle start-up – test as much as possible within the vehicle start-up time constraints – known as offline.
- Vehicle Shutdown or diagnostic testing – maximum test coverage, no time constraints when vehicle powered down– known as On-line

For off-line testing the BIST tests can be configured to execute every time the MCU boots or gets a destructive reset. This procedure is performed while the MCU is powered and held in reset. In this mode, user configurable Device Configuration Format (DCF) Records that are stored in the one-time programmable UTEST memory are loaded at start-up by the System Status and Configuration Module (SSCM) module into the STCU2 to configure the self-test procedure. When the BIST executes successfully in offline mode the device exits reset and the application software is executed.

In addition to being able to run at start-up under control of the STCU2, the MCU allows software to write to the STCU2 during runtime to configure and trigger the execution of MBIST or LBIST. This is known as On-line testing.

The intended usage of on-line BIST is to execute a test of memory and modules that are critical to the start-up of the application. This helps to minimize the startup time of the MCU. Executing a full BIST at start-up will exceed the needs of many users. On-line testing is mainly intended for a full BIST of the MCU, typically performed prior to shutdown of the ECU, when execution time is not as critical. The On-line mode can also be used for failure diagnostics and quality control within a manufacturing environment.

4. Self-Test Control Unit

The STCU2 is a programmable hardware module that controls the self-test sequence applied both during the offline and/or On-line conditions. It is able to manage by hardware the device's LBIST and MBIST blocks. To control offline BIST testing the STCU2 operates in conjunction with the System Status and Configuration Module (SSCM) module which has the ability load the Self-test parameters from flash memory automatically during the boot phase. The SSCM interface is able only to write the configuration parameters and start the Self-Test execution once after the STCU2 global reset has been applied.

To configure On-line BIST testing via software an IPS interface allows access by the device CPU(s) to the STCU2 registers. Using this interface software can configure the STCU2 registers for execution of the On-line tests, or check the results of the offline tests.

For off-line testing the BIST tests can be configured to execute every time the MCU boots or gets a destructive reset. This procedure is performed while the MCU is powered and held in reset. In this mode, user configurable Device Configuration Format (DCF) Records that are stored in the one-time programmable UTEST memory are loaded at start-up by the System Status and Configuration Module (SSCM) module into the STCU2 to configure the self-test procedure. When the BIST executes successfully in offline mode the device exits reset and the application software is executed.

5. MBIST and LBIST testing and partitions

This section details the partitioning of the memory and logic on the MPC5746R.

5.1. LBIST testing

Logic Built-In Self-Test (LBIST) is implemented by 4 LBIST controllers which operate independently on each LBIST partition. This independence is needed to meet safety requirements regarding the independence and diversity of replicated IP and also helps to avoid exceeding power limits. Each of the LBIST controllers is connected independently to the STCU2. The STCU2 can be configured by user to run LBIST in parallel, sequentially or combination of parallel and sequential.

5.2. LBIST partitions

The self-test of each logic partition is deemed to be successful by checking the resulting Multiple Input Signature Register (MISR) value of the LBIST against a 64-bit expected MISR value. The Multiple – Input Signature Register is a type of linear feedback signature register. Each state of the MISR relies on the previous states rather than just the current state, so the MISR will always generate the same correct output sequence from the same input sequence unless there is a fault in the tested logic. The STCU2 provides registers for the result and expected MISR. It is important to note that the expected MISR values and other LBIST configuration of the device may change with any modification to the internal logic design of the device. The expected values provided with this application note apply only to the 1N83M mask set of the MPC5746R device. In case of different mask please refer to reference manual.

Table 1. LBIST Partitions

Lake	Partition	IP Module
0	L0	CAN 1
0	L0	CAN 3
0	L0	DMA_MUX 3
0	L0	DSPI_1
0	L0	DSPI_3
0	L0	DSPI_M1
0	L0	SENT_1
0	L0	CMU_0
0	L0	CMU_1
0	L0	CMU_2
0	L0	CMU_3
0	L0	CRC 1
0	L0	LINFlexD_1
0	L0	LINFlexD_M1
0	L0	LINFlexD_3
0	L0	eMIOS 1
0	L0	AIPS1
0	L0	INTC
0	L0	Checker_DMA
0	L0	Checker_Core_0

1	L1	DSPI_0
1	L1	DSPI_M0
1	L1	DSPI_2
1	L1	DSPI_4
1	L1	PLLDIG
1	L1	CMU_9
1	L1	eMIOS 0
1	L1	AXBS
1	L1	PRAMC
1	L1	Main Core_0
1	L1	Main Core_1
1	L1	DMA
1	W0	CAN 0
1	W0	CAN 2
1	W0	eTPU
1	W0	BAM
1	W0	LFAST
1	W0	DMAMUX_0
1	W0	DMAMUX_1
1	W0	DMAMUX_2
1	W0	DTS
1	W0	JDC
1	W0	MC_ME
1	W0	MC_PCU
1	W0	SENT_0
1	W0	TDM
1	W0	WKPU
1	W0	PIT0
1	W0	PIT1
1	W0	CMU_6
1	W0	CMU_4
1	W0	CMU_5
1	W0	CMU_7
1	W0	CMU_8
1	W0	CRC 0
1	W0	JTAGM
1	W0	LINFlexD_0
1	W0	LINFlexD_m0
1	W0	SDADC
1	W0	SIPI
1	W0	SIUL2
1	W0	DECFILTER 0
1	W0	DECFILTER 1
1	W0	REACM2
1	W0	IAHBG
1	W0	IGF
1	W0	BCTU
1	W0	SEMA42
1	W0	STM0
1	W0	STM1
1	W0	SWT0
1	W0	SWT1
1	W0	AIPS0

1	W0	FEC
1	W0	PCM
1	W0	SWT3
1	W1	SPU
1	W1	MEMU
1	W1	NAR
1	W1	CMU_12
1	W1	FCCU
1	W1	Flash
1	W1	SMPU
1	W1	Flash Memory Controller

5.3. LBIST configuration

The following table shows the configurations the user should set to run LBIST on a specific partition as well as the expected MISR and coverage values at the end of the run.

Table 2. LBIST Configuration by Partition and Expected MISR

Partition	Clock Config	PRPG seed	MISR seed	Number of Patterns	Number of Shifts	Expected MISR	Coverage
L0	1	3fffffffffff	ffffffffffffffff	2,150	63	64'h18F2A1F5FDC31B19	90%
L1	1	ffffffffffff		2,000	48	64'hAC001775D8E3627D	90%
W0	1	ffffffffffffffff		2,100	55	64'h765D7C3D4A32B225	90%
W1	1	3fffffffffff		2,750	62	64'hB489FD55EED3E3B7	90%

For LBIST On-line self-test, the user must program the STCU_LBRMSW[LBRMSWn] field corresponding to the LBIST partition to "1" to generate a global functional reset at the end of the test:

- LBRMSW0: corresponds to LBIST L0
- LBRMSW1: corresponds to LBIST L1
- LBRMSW2: corresponds to LBIST W0
- LBRMSW3: corresponds to LBIST W1

5.4. MBIST testing

The MBIST is executed by a single MBIST controller that is programmed via the STCU2. The MBIST engine controls the self-test of multiple partitions. The MPC5746R memory is split into 47 different MBIST partitions which are numbered 0 – 46.

The MBIST controller has three types of march test it can apply when running MBIST that allow for different coverage levels.

- The Full Test Mode tests memory using all algorithms including the open PMOS algorithm.
- The Reduced Test Mode tests memory using all algorithms except the open PMOS algorithm.
- The Auto Test Mode uses a smaller set of algorithms which has lower coverage.

The Auto Test mode is designed to quickly test the RAM with good fault coverage, so this mode is recommended for the off-line BIST as it has an optimum balance of test time versus fault coverage. The Full Test Mode is comparable to the tests used in the NXP factory to test the RAM. Since NXP has already tested the parts prior to delivery, the primary concern for the user should be to detect latent defects. The Full Test is recommended for the on-line BIST since time constraints are typically not as critical in the user's on-line use case.

Table 3. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_ CFG[PMOSEN]	STCU2_ CFG[MBU]	MBIST Algoritm	InternalNodes/ Circuits Additionally Covered by Test	Usecase
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by NXP production test. Recommended to use in the field for MBIST On-line self-test.
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm	Address Decoder and Bitcell	Recommended to use in the field for MBIST On-line self-test if the Full Test takes too long.
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self- test as an optimum balance of test time vs. fault coverage.

5.5. MBIST partitions

The Table 4 describes the mapping between the MBIST number as the STCU2 captures it, the corresponding memory, and the corresponding MBIST control register.

Table 4. MBIST mapping

Partition Number	Memory	Corresponding MBIST Control Register
0	FlexCAN_3 MB/Mask (2 of 2)	STCU_MB_CTRL_0
1	FlexCAN_3 MB/Mask (1 of 2)	STCU_MB_CTRL_1
2	FlexCAN_1 MB/Mask (2 of 2)	STCU_MB_CTRL_2
3	FlexCAN_1 MB/Mask (1 of 2)	STCU_MB_CTRL_3
4	BAM ROM	STCU_MB_CTRL_4
5	Core0 I-MEM	STCU_MB_CTRL_5
6	Core0 D-MEM (2 of 2)	STCU_MB_CTRL_6
7	Core0 D-MEM (1 of 2)	STCU_MB_CTRL_7
8	Core0 I-Cache Data (4 of 4)	STCU_MB_CTRL_8
9	Core0 I-Cache Data (3 of 4)	STCU_MB_CTRL_9
10	Core0 I-Cache Data (2 of 4)	STCU_MB_CTRL_10
11	Core0 I-Cache Data (1 of 4)	STCU_MB_CTRL_11
12	Core0 I-Cache Tag	STCU_MB_CTRL_12

13	Core1 I-MEM	STCU_MB_CTRL_13
14	Core1 D-MEM (2 of 2)	STCU_MB_CTRL_14
15	Core1 D-MEM (1 of 2)	STCU_MB_CTRL_15
16	Core1 I-Cache Data (4 of 4)	STCU_MB_CTRL_16
17	Core1 I-Cache Data (3 of 4)	STCU_MB_CTRL_17
18	Core1 I-Cache Data (2 of 4)	STCU_MB_CTRL_18
19	Core1 I-Cache Data (1 of 4)	STCU_MB_CTRL_19
20	Core1 I-Cache Tag	STCU_MB_CTRL_20
21	PRAM0 32kB (Standby RAM)	STCU_MB_CTRL_21
22	PRAM1 32kB	STCU_MB_CTRL_22
23	PRAM3 64kB	STCU_MB_CTRL_23
24	PRAM2 64kB	STCU_MB_CTRL_24
25	PRAM4 64kB	STCU_MB_CTRL_25
26	eDMA TCD	STCU_MB_CTRL_26
27	FlexCAN_2 MB/Mask (2 of 2)	STCU_MB_CTRL_27
28	FlexCAN_2 MB/Mask (1 of 2)	STCU_MB_CTRL_28
29	FlexCAN_0 MB/Mask (2 of 2)	STCU_MB_CTRL_29
30	FlexCAN_0 MB/Mask (1 of 2)	STCU_MB_CTRL_30
31	FEC FIFO	STCU_MB_CTRL_31
32	FEC MIB Counter	STCU_MB_CTRL_32
33	eTPU Shared Data Memory	STCU_MB_CTRL_33
34	eTPU Shared Code Memory	STCU_MB_CTRL_34
35	eTPU Nexus Trace (2 of 2)	STCU_MB_CTRL_35
36	eTPU Nexus Trace (1 of 2)	STCU_MB_CTRL_36
37	NAR1 (2 of 2)	STCU_MB_CTRL_37
38	NAR1 (1 of 2)	STCU_MB_CTRL_38
39	NAR0 (2 of 2)	STCU_MB_CTRL_39
40	NAR0 (1 of 2)	STCU_MB_CTRL_40
41	NAR3 (2 of 2)	STCU_MB_CTRL_41
42	NAR3 (1 of 2)	STCU_MB_CTRL_42
43	NAR2 (2 of 2)	STCU_MB_CTRL_43
44	NAR2 (1 of 2)	STCU_MB_CTRL_44
45	Internal Overlay RAM (2 of 2)	STCU_MB_CTRL_45
46	Internal Overlay RAM (1 of 2)	STCU_MB_CTRL_46

5.6. MBIST test cycles

The Table 5 describes the test cycles for different MBIST partitions on this chip according to test type.

Table 5. MBIST Test Cycles

Partition Number	Memory	Size	Full Test		Reduced Test		Auto Test	
			TCKs	MEM Clks	TCKs	MEM Clks	TCKs	MEM Clks
0	FlexCAN_3 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
1	FlexCAN_3 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
2	FlexCAN_1 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
3	FlexCAN_1 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
4	BAM ROM	2048 x 32	2244	15422	179	10282	N/A	N/A
5	Core0 I-MEM	2048 x 72	2244	269167	2112	252429	528	36970
6	Core0 D-MEM (2 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
7	Core0 D-MEM (1 of 2)	4096 x 40	2244	525247	2112	504333	528	73834

8	Core0 I-Cache Data (4 of 4)	256 x 72	2244	33903	2112	32013	528	4714
9	Core0 I-Cache Data (3 of 4)	256 x 72	2244	33903	2112	32013	528	4714
10	Core0 I-Cache Data (2 of 4)	256 x 72	2244	33903	2112	32013	528	4714
11	Core0 I-Cache Data (1 of 4)	256 x 72	2244	33903	2112	32013	528	4714
12	Core0 I-Cache Tag	128 x 65	2244	17135	2112	16269	528	2410
13	Core1 I-MEM	2048 x 72	2244	269167	2112	252429	528	36970
14	Core1 D-MEM (2 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
15	Core1 D-MEM (1 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
16	Core1 I-Cache Data (4 of 4)	256 x 72	2244	33903	2112	32013	528	4714
17	Core1 I-Cache Data (3 of 4)	256 x 72	2244	33903	2112	32013	528	4714
18	Core1 I-Cache Data (2 of 4)	256 x 72	2244	33903	2112	32013	528	4714
19	Core1 I-Cache Data (1 of 4)	256 x 72	2244	33903	2112	32013	528	4714
20	Core1 I-Cache Tag	128 x 65	2244	17135	2112	16269	528	2410
21	PRAM0 32kB (Standby RAM)	4096 x 72	2244	527087	2112	504333	528	73834
22	PRAM1 32kB	4096 x 72	2244	527087	2112	504333	528	73834
23	PRAM3 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
24	PRAM2 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
25	PRAM4 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
26	eDMA TCD	256 x 72	2244	33903	2112	32013	528	4714
27	FlexCAN_2 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
28	FlexCAN_2 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
29	FlexCAN_0 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
30	FlexCAN_0 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
31	FEC FIFO	128 x 44	2244	17135	2112	16269	528	2410
32	FEC MIB Counter	64 x 40	2244	8815	2112	8397	528	1258
33	eTPU Shared Data Memory	1536 x 52	2244	206527	2112	192525	528	30826
34	eTPU Shared Code Memory	6144 x 39	2244	800111	2112	768525	528	122986
35	eTPU Nexus Trace (2 of 2)	32 x 72	2244	4687	2112	4461	528	682
36	eTPU Nexus Trace (1 of 2)	32 x 72	2244	4687	2112	4461	528	682
37	NAR1 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
38	NAR1 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
39	NAR0 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
40	NAR0 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
41	NAR3 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
42	NAR3 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
43	NAR2 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
44	NAR2 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
45	Internal Overlay RAM (2 of 2)	1024 x 72	2244	135791	2112	126477	528	18538
46	Internal Overlay RAM (1 of 2)	1024 x 72	2244	135791	2112	126477	528	18538

6. DCF records, clients and configuration

DCF records are used to configure device register values during reset phase. When reset is released the desired registers contain values defined by DCF content. Figure 1 represents when SSCM use stored DCF records to configure device (also BIST).

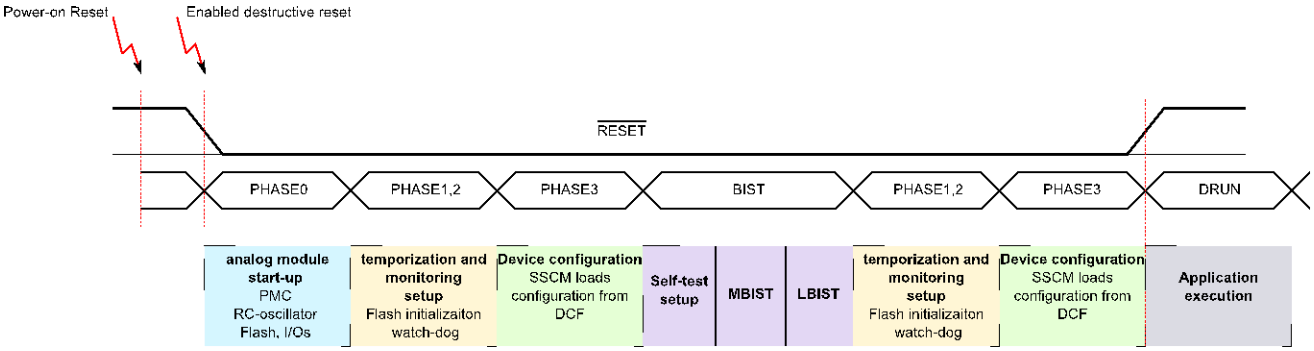

Figure 1. Reset sequence

The offline BIST tests are configured using Device Configuration Format (DCF) records that automatically configure the STCU2 to enable and run BIST at start-up or following a destructive reset.

The DCF is a mechanism to automatically configure specific registers during system boot and to set up an initial configuration for the device after reset or start up. The term DCF client is used to describe a module whose registers can be written by DCF record, e.g. the STCU2 is a DCF client. DCF records are stored in both TEST and UTEST flash. During the boot sequence of the device the SSCM automatically loads the DCF records to the DCF clients.

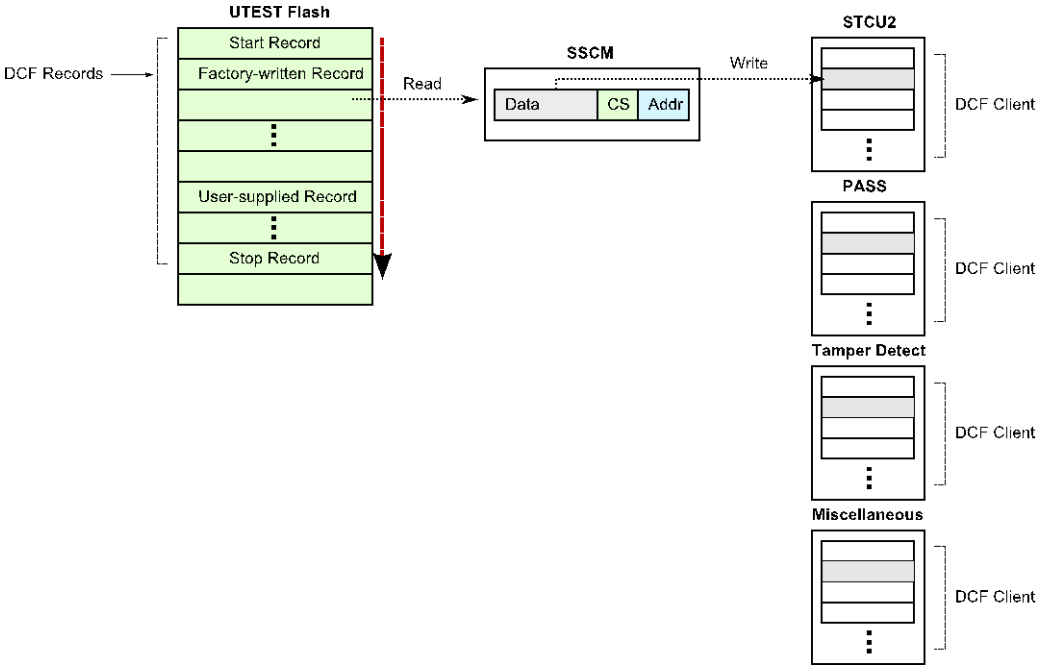

Figure 2. DCF record mechanism

7. On-line BIST configuration

To successfully perform On-line BIST correct configuration must be done before BIST execution. This chapter describes how to correctly prepare, setup and execute On-line BIST.

On-line BIST configuration consist of these steps:

1. DCF IPS clients configuration
2. Unlock the STCU2
3. Program the Reset management
4. Overwrite/Program the STCU2_MB_CTRL registers
5. Overwrite/Program the STCU2_WDG register
6. Overwrite/Program the STCU2_CFG register
7. Overwrite/Program the STCU2_LB_CTRL registers
8. Start Online BIST

7.1. DCF IPS clients configuration

For the successful On-line BIST execution is necessary to program few DCF records.

- DCF_SELFTEST_CONFIG_IPS_1 DCF client

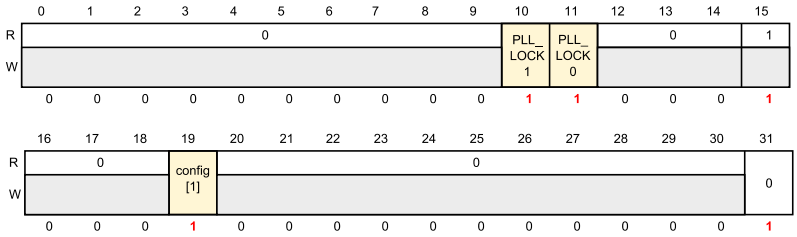

Figure 3. DCF_SELFTEST_CONFIG_IPS_1 DCF client

- DCF_SELFTEST_CONFIG_IPS_4 DCF client

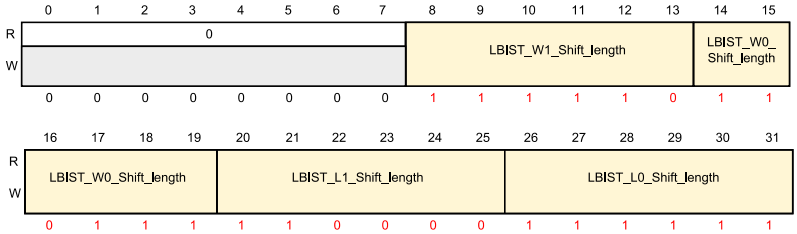

Figure 4. DCF_SELFTEST_CONFIG_IPS_4 DCF client

7.1.1. DCF_SELFTEST_CONFIG_IPS_1 DCF client

The DCF_SELFTEST_CONFIG_IPS_1 register controls the fault Model for LBIST and selects which PLL would be used for LBIST testing if any. User must perform following steps in order to successfully pass the BIST tests.

- Configure PLL_LOCK bits (determine which clock is used for self-tests):
- Program config[1] bit to “1”.
- Set the zero bit to 1 when running LBIST to get correct MISR values. This is required in order to bypass the memories scan chains (implementation for Test purposes only).

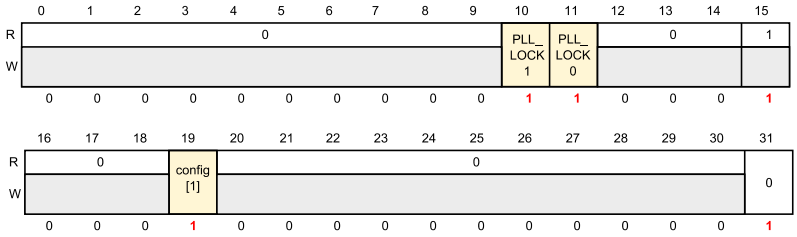

Figure 5. User configuration of DCF_SELFTEST_CONFIG_IPS_1 DCF client

DCF_SELFTEST_CONFIG_IPS_1 DCF client construction:

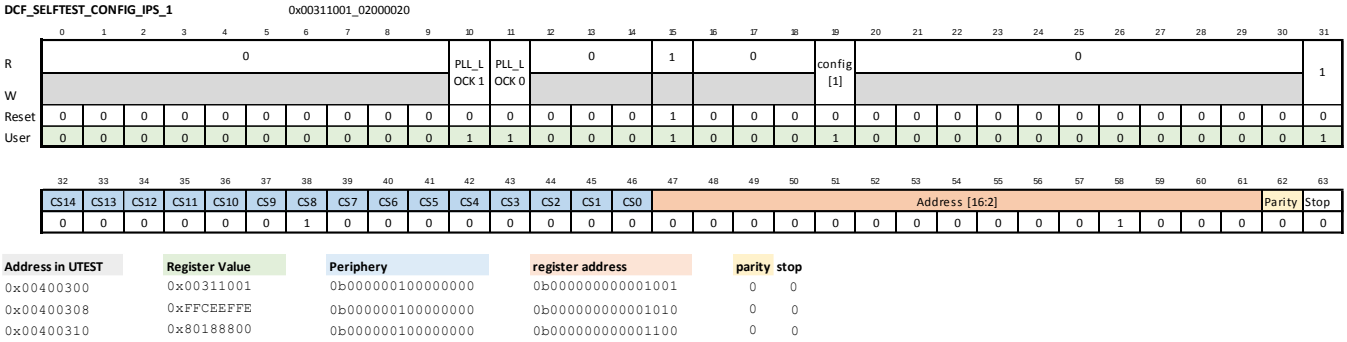

Figure 6. IPS_1 DCF record construction

When the DCF client is constructed the special strategy must be used in order to correctly write to DCF STCU2 client. For IPS_1 client spread address and triple voting special DCF record strategy are used. This means 3 DCF records need to be written in order to correctly configure IPS_1 DCF record as shown at Figure 7. For more details refer to reference manual of MPC5746R.

DCF records

```
0x00400300        0x0031100102000024
0x00400308        0xFFCEEFFE02000028
0x00400310        0x8018880002000030
```

Figure 7. IPS_1 DCF record format

Lauterbach script example:

```
data.set 0x00400300 %QUAD 0x0031100102000024 ;DCF_SELFTEST_CONFIG_IPS_1
data.set 0x00400308 %QUAD 0xFFCEEFFE02000028
data.set 0x00400310 %QUAD 0x8018880002000030
```

Figure 8. IPS_1 Lauterbach script example

7.1.2. DCF_SELFTEST_CONFIG_IPS_4 DCF client

The DCF_SELFTEST_CONFIG_IPS_4 register controls the LBIST shift length for each partition.

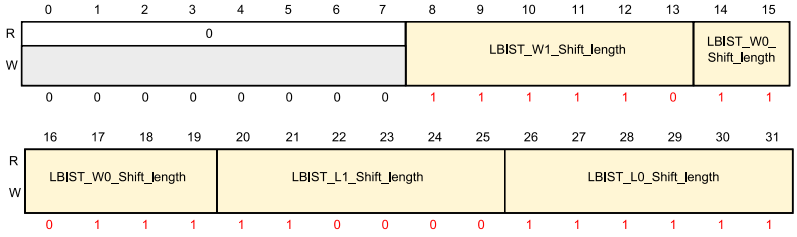

Figure 9. User configuration of DCF_SELFTEST_CONFIG_IPS_4 DCF client

The correct configuration of LBIST shift registers must be done in order to successfully execute LBIST. The LBIST shift length for IPS_4 register is described in reference manual and can be also found at LBIST Configuration by Partition and Expected MISR. Follow the recommended shift lengths as this chapter describes in order to get correct expected MISR.

DCF_SELFTEST_CONFIG_IPS_4 DCF client construction:

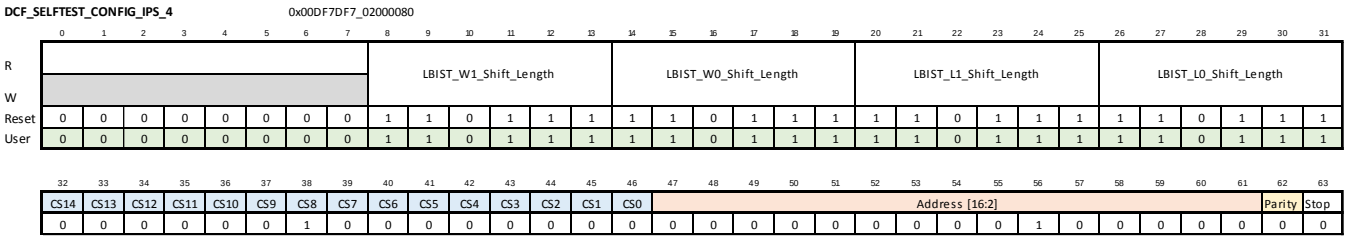

Figure 10. IPS_4 DCF record construction

When the DCF client is constructed the special strategy must be used in order to correctly write to DCF STCU2 client. For IPS_4 client spread address and triple voting special DCF record strategy are used. This means 3 DCF records need to be written in order to correctly configure IPS_4 DCF record as shown at Figure 11IPS_1 DCF record format. For more details refer to reference manual of MPC5746R.

DCF records

```
0x00400318          0x00FB7C3F02000084
0x00400320          0xFF0483C002000088
0x00400328          0x807DBE1F02000090
```

Figure 11. IPS_4 DCF record format

Lauterbach script example:

```
data.set 0x00400318 %QUAD 0x00FB7C3F02000084 ;DCF_SELFTEST_CONFIG_IPS_4
data.set 0x00400320 %QUAD 0xFF0483C002000088
data.set 0x00400328 %QUAD 0x807DBE1F02000090
```

Figure 12. IPS_4 Lauterbach script example

7.1.3. Example of UTEST user DCF area

Figure 13 represents how DCF user area looks like after configuration of IPS DCF clients. Six 64-bit writes are done in order to correctly configure IPS_1 and IPS_4 registers.

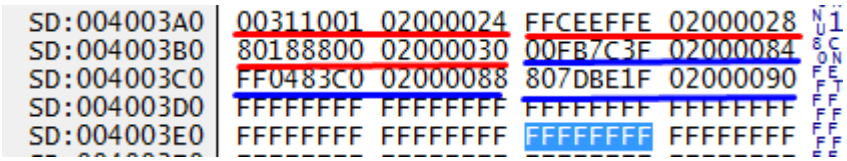

Figure 13. Flash UTEST area after DCF IPS client programming

7.2. Unlock the STCU2

The STCU2_SKC register implements the security key code mechanism needed to access the write mode for write protected STCU2 registers.

In order to unlock the STCU2 access after:

• The STCU2 asynchronous reset

• The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

• write the key1 into the STCU2_SKC register

• write the key2 into the STCU2_SKC register

Unlock the On-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register.

```
STCU2.SKC.R = 0x753F924E;
STCU2.SKC.R = 0x8AC06DB1;
```

NOTE

In case it is required to extend the STCU2 register access cycles before the hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter.

7.3. Program the Reset management

Program the Reset management during on-Line LBIST execution setting the STCU2_LBRMSW register.

```
STCU2.LBRMSW.R = 0xF;                    Dedicated functional reset is pulsed at the end
of LBIST run
```

7.4. Overwrite/Program the STCU2_MB_CTRL registers

The STCU2_MB_CTRL register defines the control setting of MBIST controller. Overwrite/Program the STCU2_MB_CTRL registers of each NMCUT to be executed.

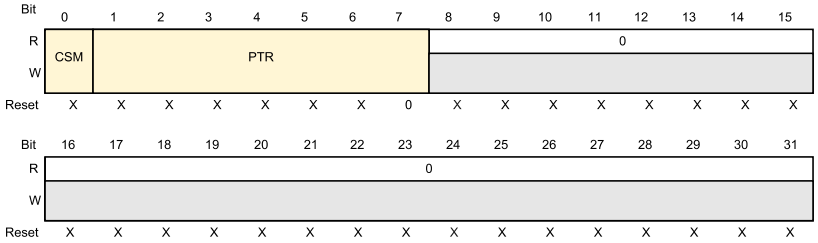

Figure 14. STCU2_MB_CTRL registers

CSM - Concurrent/sequential mode

PTR - PTR defines the logical pointer to the next LBIST or MBIST to be scheduled.

The example of MBIST configuration is shown in STCU2_MB_CTRL configuration example appendix.

NOTE

Last MBIST control register must be configured to sequential mode. Otherwise the STCU will report pointer configuration error via error status register bit invalid linked pointer list.

7.5. Overwrite/Program the STCU2_WDG register

Overwrite/Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution.

```
STCU2.WDG.R = 0xFFFFFFFF;                /* Watchdog time out to Max value */
```

7.6. Overwrite/Program the STCU2_CFG register

Overwrite/Program the STCU2_CFG register in order to define: the core and L/MBIST TCK clock presaging factor setting the CLK_CFG bits, and finally set the pointer to the first LBIST/NMCUT to be executed.

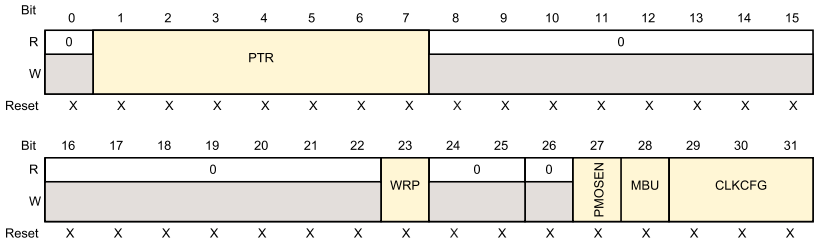

Figure 15. STCU2_CFG register

```
STCU2.CFG.B.CLK_CFG = 0;          // Logic, Memory BIST and STCU2 core CLK Clock configuration
STCU2.CFG.B.MBU = 0;                  // MBIST MBU Test Enabled Full or AUTO
STCU2.CFG.B.PMOSEN = 1;          // MBIST PMOS Test Enable (Full test)
STCU2.CFG.B.WRP = 0;                  // Write Protection
STCU2.CFG.B.PTR = 0x10;          // Start with MBIST_0 pointer (0x10)
```

7.7. Overwrite/Program the STCU2_LB_CTRL registers

The STCU2_LB_CTRL register defines the control setting of each LBIST controller.

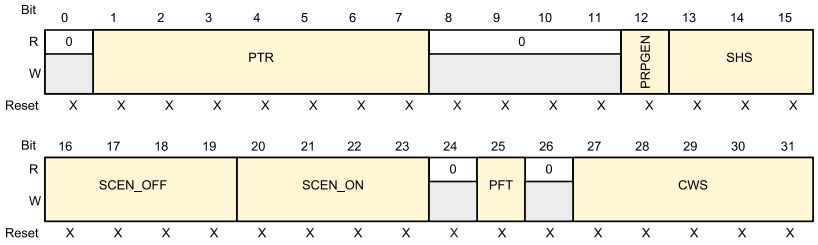

Figure 16. STCU2_LB_CTRL

The example of LBIST configuration is shown in STCU2_LB_CTRL configuration example.

7.8. Start online BIST

Program the RUNSW bit into the STCU2_RUNSW register to enable the On-Line Self-Test execution the L/MBSWPLLEN bits to enable the PLL lock signal monitor during L/MBIST execution and the L/MBIE to enable the related Interrupt requests.

```
/* Start the BIST sequence */
STCU2.RUNSW.R = 0x00000301;
```

Verify if the BIST finished correctly via checking RUNSW bit.

```
while (STCU2.RUNSW.B.RUNSW == 1);
```

8. Off-line BIST

To successfully perform Off-line BIST, correct configuration must be done before BIST execution. This chapter describes how to correctly prepare, setup and execute Off-line BIST.

Off-line BIST is configured through DCF records and it uses special DCF record strategies such as triple voting or spread address. For more details on DCF records refer to reference manual of MPC5746R.

Offline BIST consist of following steps:

1. Unlocking STCU module for Off-line BIST configuration
2. Overwrite/Program the STCU2_CFG register
3. Overwrite/Program the STCU2_PLL_CFG register
4. Overwrite/Program the STCU2_WDG register
5. Overwrite/Program the STCU2_MB_CTRL registers
6. Overwrite/Program the STCU2_LB_CTRL registers
7. DCF IPS clients configuration
8. Start Off-line BIST

Attachment of this document contain DCF calculator. It is useful excel tool prepared to ease working with DCF records.

Calculator is capable of:

- DCF record construction (parity, special strategy, etc.)
- Correct memory address calculation
- Partial Lauterbach debugger script generation

8.1. Unlocking STCU module for Off-line BIST configuration

As STCU module is safety critical part of microcontroller it is protected to unintended writes.

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

NOTE

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

STCU2 security key code for off-line Test

Key1 to unlock the write access the STCU2 = **0xD3FEA98B**

Key2 to unlock the write access the STCU2 = **0x2C015674**

Lauterbach script example:

Unlocks writes to the STCU:

```
data.set 0x00400310 %QUAD 0xD3FEA98B00080008 ;STCU SKU Key 1
data.set 0x00400318 %QUAD 0x2C01567400080008 ;STCU SKU Key 2
```

8.2. Overwrite/Program the STCU2_CFG register

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

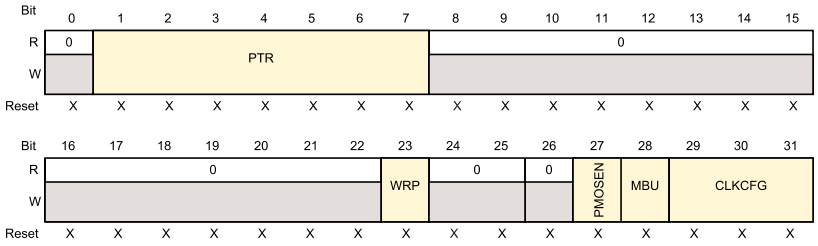

Figure 17. STCU2_CFG register

8.3. Overwrite/Program the STCU2_PLL_CFG register

The STCU2_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU2_RUN[MBPLLEN] = 1 or STCU2_RUN[LBPLLEN] = 1. In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue. For the offline self-test IRC is the only reference clock for the STCU_PLL.

NOTE

User should program PLL to maximum 100MHz for Offline BIST. In addition, configure the system clock dividers in

DCF_SELFTEST_CONFIG_IPS_2 DCF client to divide by 1, 1 and 3 (FXBAR 100MHz; SXBAR 100MHz; PBRIDGE 33.3MHz)

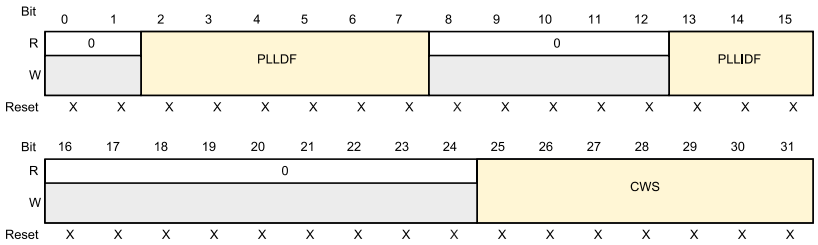

Figure 18. STCU2_PLL_CFG register

8.4. Overwrite/Program the STCU2_WDG register

The STCU2_WDG register defines the time budget of LBIST and MBIST execution providing a protection mechanism in case of dead-lock or end-less conditions during the Self-Test procedure.

Lauterbach script example:

data.set 0x00400330 %QUAD 0x02DC6C0000080014 ;STCU WDG Watchdog time out to Max value

8.5. Overwrite/Program the STCU2_MB_CTRL registers

Configure desired memory locations to be tested. User can choose from concurrent and sequential mode. However last tested partition must be set to sequential execution. Otherwise STCU2 will report pointer error in STCU2_ERR_STAT register. The example of MBIST configuration for Off-line BIST is in Off-line BIST Lauterbach script example.

8.6. Overwrite/Program the STCU2_LB_CTRL registers

The STCU2_LB_CTRL register defines the control setting of each LBIST controller. If the LBIST is running after MBIST then set last LBISTn pointer to value 0x7F. This setting is pointer to NILL which means end of LBIST execution. Incorrect pointer will result into error which will be latched in STCU2_ERR register.

8.7. DCF IPS clients configuration

It is necessary to correctly configure DCF_IPS clients before start of the BIST. For Off-line BIST there are used 4 IPS clients.

8.7.1. DCF_SELFTEST_CONFIG_IPS_1 DCF client

The DCF_SELFTEST_CONFIG_IPS_1 client controls the fault Model for LBIST and selects which PLL would be used for LBIST testing if any.

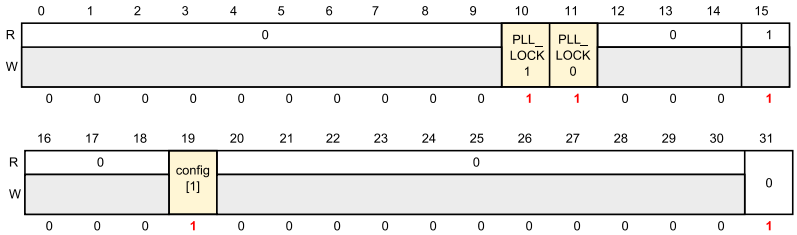

Figure 19. DCF_SELFTEST_CONFIG_IPS_1 DCF client

When the DCF client is constructed the special strategy must be used in order to correctly write to DCF STCU2 client. For IPS_1 client spread address and triple voting special DCF record strategy are used. This means 3 DCF records need to be written in order to correctly configure IPS_1 DCF record as shown at Figure 20IPS_1 DCF record format. For more details refer to reference manual of MPC5746R.

Configuration example for DCF record calculation:

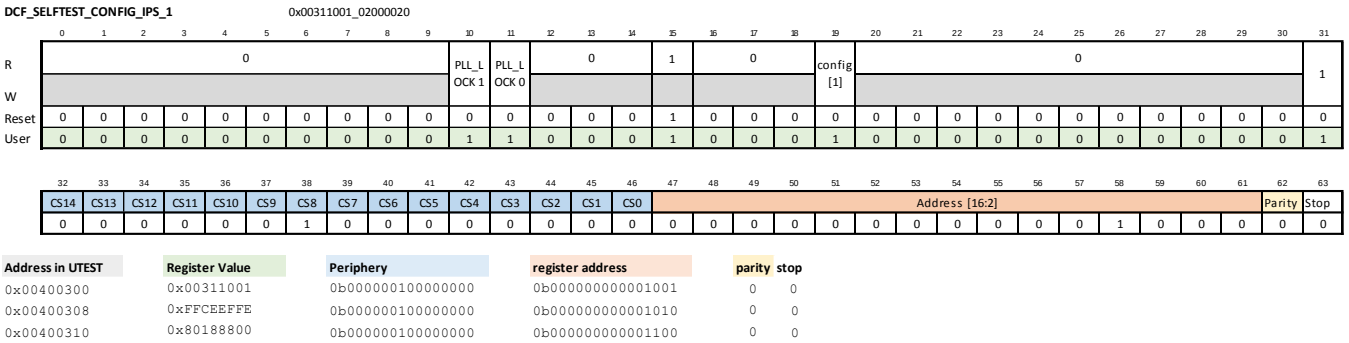

Figure 20. IPS_1 DCF record construction

DCF records

```
0x00400300          0x0031100102000024
0x00400308          0xFFCEEFFE02000028
0x00400310          0x8018880002000030
```

Figure 21. IPS_1 DCF record format

Lauterbach script example:

```
data.set 0x00400300 %QUAD 0x0031100102000024 ;DCF_SELFTEST_CONFIG_IPS_1
data.set 0x00400308 %QUAD 0xFFCEEFFE02000028
data.set 0x00400310 %QUAD 0x8018880002000030
```

Figure 22. IPS_1 Lauterbach script example

NOTE

Configure PLL_Lock 0 or PLL_Lock 1 according to application needs.
Configuration depend on source clock selected for BISTs.

8.7.2. DCF_SELFTEST_CONFIG_IPS_2 DCF client (optional)

The DCF_SELFTEST_CONFIG_IPS_2 DCF client sets division values of clock generation module dividers. This dividers configuration is set and used only during BIST.

Configure the system clock dividers in DCF_SELFTEST_CONFIG_IPS_2 DCF client to divide by 1, 1 and 3 (FXBAR 100MHz; SXBAR 100MHz; PBRIDGE 33.3MHz)

DCF_SELFTEST_CONFIG_IPS_2[20:22] = 2 -> PBRIDGE CLK

DCF_SELFTEST_CONFIG_IPS_2[23:25] = 0 -> CMU_SXBAR

DCF_SELFTEST_CONFIG_IPS_2[26:28] = 0 -> CMU_FXBAR

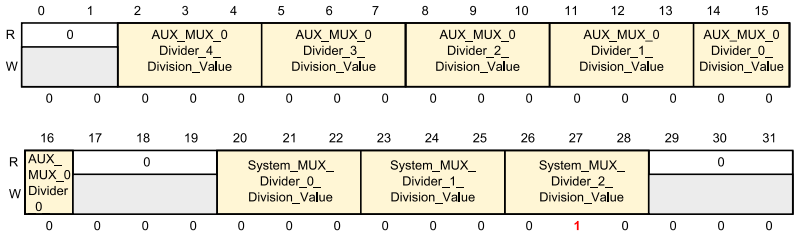

Figure 23. DCF_SELFTEST_CONFIG_IPS_2 DCF client

Dividers which can be configured by IPS_2 client:

- System Clock Divider Configuration 0
- System Clock Divider Configuration 1
- System Clock Divider Configuration 2
- Aux Clock 0 Divider Configuration 0
- Aux Clock 0 Divider Configuration 1
- Aux Clock 0 Divider Configuration 2
- Aux Clock 0 Divider Configuration 3
- Aux Clock 0 Divider Configuration 4

Configuration example for DCF record calculation:

DCF_SELFTEST_CONFIG_IPS_2 0x00000010_02000040

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		Aux_MUX_0_Divider_4_Division_Value				Aux_MUX_0_Divider_3_Division_Value		Aux_MUX_0_Divider_2_Division_Value			Aux_MUX_0_Divider_1_Division_Value			Aux_MUX_0_Divider_0_Division_Value			0			System_MUX_Divider_0_Division_Value			System_MUX_Divider_1_Division_Value			System_MUX_Divider_2_Division_Value			0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
User	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	CS14	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Address [16:2]															Parity	Stop
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 24. IPS_2 DCF record construction

DCF records

```
0x00400330          0x0000001002000044
0x00400338          0xFFFFFFEF02000048
0x00400340          0x0000000802000050
```

Figure 25. IPS_2 DCF record format

Lauterbach script example:

```
data.set 0x00400318 %QUAD 0x0000001002000044 ;DCF_SELFTEST_CONFIG_IPS_2
data.set 0x00400320 %QUAD 0xFFFFFFEF02000048
data.set 0x00400328 %QUAD 0x0000000802000050
```

Figure 26. IPS_2 Lauterbach script example

8.7.3. DCF_SELFTEST_CONFIG_IPS_3 DCF client (optional)

The DCF_SELFTEST_CONFIG_IPS_3 register controls the division values for the dividers. This dividers configuration is set and used only during BIST.

- Aux Mux 1 Divider 0 Division Value
- Aux Mux 2 Divider 0 Division Value
- Aux Mux 5 Divider 0 Division Value
- Aux Mux 5 Divider 1 Division Value
- Aux Mux 6 Divider 0 Division Value
- Aux Mux 7 Divider 0 Division Value
- Aux Mux 8 Divider 0 Division Value
- Aux Mux 9 Divider 0 Division Value
- Aux Mux 10 Divider 0 Division Value
- Aux Mux 11 Divider 0 Division Value

8.7.4. DCF_SELFTEST_CONFIG_IPS_4 DCF client

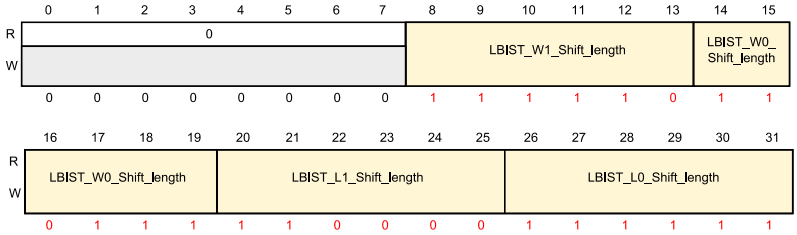

Figure 27. User configuration of DCF_SELFTEST_CONFIG_IPS_4 DCF client

The correct configuration of LBIST shift registers must be done in order to successfully execute LBIST. The LBIST shift length for IPS_4 register is described in reference manual and can be also found at LBIST Configuration by Partition and Expected MISR. Follow the recommended shift lengths as this chapter describes in order to get correct expected MISR.

DCF_SELFTEST_CONFIG_IPS_4 DCF client construction:

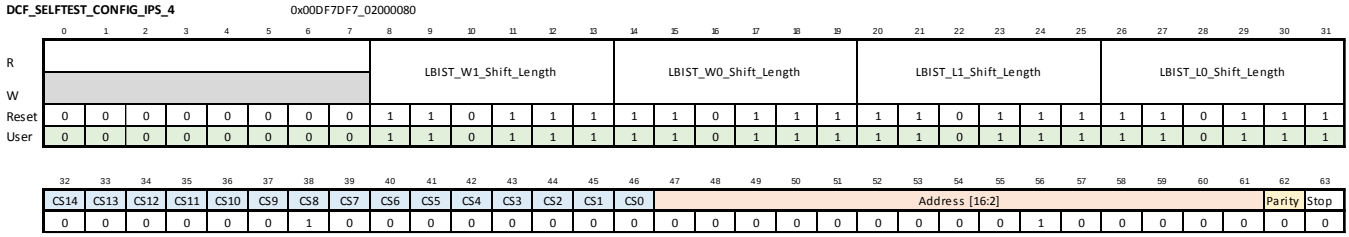

Figure 28. IPS_4 DCF record construction

When the DCF client is constructed the special strategy must be used in order to correctly write to DCF STCU2 client. For IPS_4 client spread address and triple voting special DCF record strategy are used. This means 3 DCF records need to be written in order to correctly configure IPS_4 DCF record as shown at Figure 29IPS_1 DCF record format. For more details refer to reference manual of MPC5746R.

DCF records

```
0x00400318          0x00FB7C3F02000084
0x00400320          0xFF0483C002000088
0x00400328          0x807DBE1F02000090
```

Figure 29. IPS_4 DCF record format

Lauterbach script example:

```
data.set 0x00400318 %QUAD 0x00FB7C3F02000084 ;DCF_SELFTEST_CONFIG_IPS_4
data.set 0x00400320 %QUAD 0xFF0483C002000088
data.set 0x00400328 %QUAD 0x807DBE1F02000090
```

Figure 30. IPS_4 Lauterbach script example

8.8. Start Off-line BIST

Program the RUN bit in STCU2 Run Register (STCU2_RUN) to start Off-line BIST and the L/MBSWPLLEN bits to enable the PLL lock signal monitor during L/MBIST execution.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

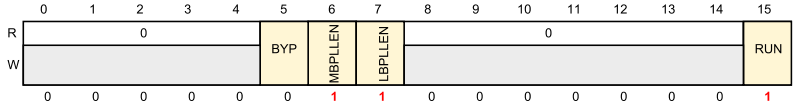

Figure 31. STCU2_RUN register

DCF STCU2_RUN client construction:

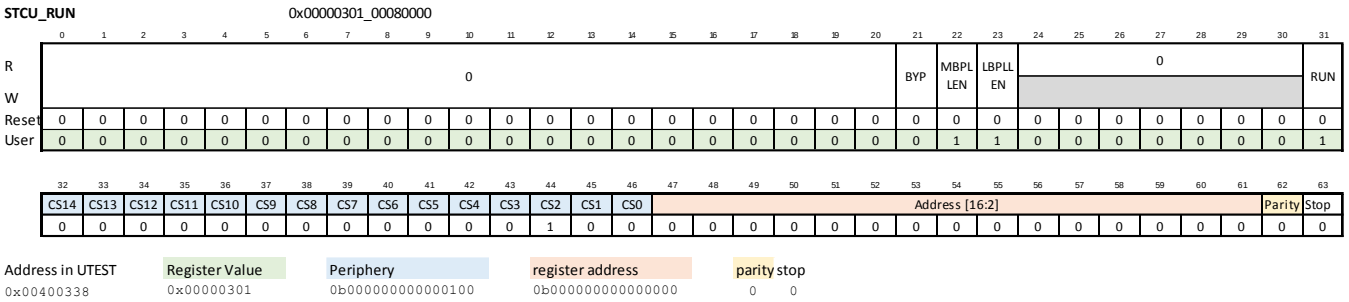

Figure 32. DCF STCU2_RUN client construction

For STCU2_RUN register no special DCF strategy is used.

Lauterbach script example:

```
data.set 0x00400330 %QUAD 0x0000030100080000 ;STCU_RUN
```

Figure 33. lauterbach script Example for IPS 1

9. BIST results

STCU2 provides set of registers where results of self-tests are stored corresponding to the execution of the selected BIST. Below is the list of these registers.

- STCU2 Error Register (STCU2_ERR_STAT)
- STCU2 Error FM Register (STCU2_ERR_FM)

Registers dedicated to Off-line self-test:

- STCU2 Off-Line LBIST Status Register (STCU2_LBS)
- STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)
- STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)
- STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)
- STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)
- STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)
- STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLn)
- STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRHn)

Registers dedicated to On-line self-test:

- STCU2 On-Line LBIST Status Register (STCU2_LBSSW)
- STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)
- STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)
- STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)
- STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)
- STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)
- STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSWn)
- STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSWn)

10. Definitions, Acronyms, and Abbreviations

Table 6. Acronyms Table

Term/Acronym	Definition
STCU2	Self-Test Control Unit
DCF	Device Configuration Format
BIST	Build In Self-Test
MBIST	Memory Build In Self-Test
LBIST	Logic Build In Self-Test
FCCU	Fault Collection and Control Unit
SSCM	System Status and Configuration Module
MISR	Multiple Input Signature Register
eDMA	Enhanced Direct Memory Access

11. References

- Using the Built-in Self-Test (BIST) on the MPC5777M (document AN5131)
- MPC5746R Reference Manual (document MPC5746RRM)

12. Revision History

Table 7. Revision history

Revision Number	Date	Substantive changes
0.1.0	10/2016	Initial Release
0.2.0	10/2016	Correction after first review
0.3.0	12/2016	Corrections after apps team review
0.4.0	1/2017	BIST clock fixes
1.0.1	9/2017	Removed LBIST delay from appendix

Appendix A. On-line BIST Code examples

A.1. STCU2_MB_CTRL configuration example

Example 1.

```
void MBIST_Config(void)
{
/* MBIST Config - MBIST Will run first */

/* Configure MBIST Control registers */
    STCU2.MB_CTRL[0].R = 0x91000000;        /* MBIST CTRL00 Run concurrently, next in
sequence is MBIST 1 */
    STCU2.MB_CTRL[1].R = 0x92000000;        /* MBIST CTRL01 Run concurrently, next in
sequence is MBIST 2 */
    STCU2.MB_CTRL[2].R = 0x93000000;        /* MBIST CTRL02 Run concurrently, next in
sequence is MBIST 3 */
    STCU2.MB_CTRL[3].R = 0x94000000;        /* MBIST CTRL03 Run concurrently, next in
sequence is MBIST 4 */
    STCU2.MB_CTRL[4].R = 0x95000000;        /* MBIST CTRL04 Run concurrently, next in
sequence is MBIST 5 */
    STCU2.MB_CTRL[5].R = 0x96000000;        /* MBIST CTRL05 Run concurrently, next in
sequence is MBIST 6 */
    STCU2.MB_CTRL[6].R = 0x97000000;        /* MBIST CTRL06 Run concurrently, next in
sequence is MBIST 7 */
    STCU2.MB_CTRL[7].R = 0x98000000;        /* MBIST CTRL07 Run concurrently, next in
sequence is MBIST 8 */
    STCU2.MB_CTRL[8].R = 0x99000000;        /* MBIST CTRL08 Run concurrently, next in
sequence is MBIST 9 */
    STCU2.MB_CTRL[9].R = 0x9A000000;        /* MBIST CTRL09 Run concurrently, next in
sequence is MBIST 10 */
    STCU2.MB_CTRL[10].R = 0x9B000000;       /* MBIST CTRL10 Run concurrently, next in
sequence is MBIST 11 */
    STCU2.MB_CTRL[11].R = 0x9C000000;       /* MBIST CTRL11 Run concurrently, next in
sequence is MBIST 12 */
    STCU2.MB_CTRL[12].R = 0x9D000000;       /* MBIST CTRL12 Run concurrently, next in
sequence is MBIST 13 */
    STCU2.MB_CTRL[13].R = 0x9E000000;       /* MBIST CTRL13 Run concurrently, next in
sequence is MBIST 14 */
    STCU2.MB_CTRL[14].R = 0x9F000000;       /* MBIST CTRL14 Run concurrently, next in
sequence is MBIST 15 */
    STCU2.MB_CTRL[15].R = 0xA0000000;       /* MBIST CTRL15 Run concurrently, next in
sequence is MBIST 16 */
    STCU2.MB_CTRL[16].R = 0xA1000000;       /* MBIST CTRL16 Run concurrently, next in
sequence is MBIST 17 */
    STCU2.MB_CTRL[17].R = 0xA2000000;       /* MBIST CTRL17 Run concurrently, next in
sequence is MBIST 18 */
    STCU2.MB_CTRL[18].R = 0xA3000000;       /* MBIST CTRL18 Run concurrently, next in
sequence is MBIST 19 */
    STCU2.MB_CTRL[19].R = 0xA4000000;       /* MBIST CTRL19 Run concurrently, next in
sequence is MBIST 20 */
```

```
    STCU2.MB_CTRL[20].R = 0xA5000000;       /* MBIST CTRL20 Run concurrently, next in
sequence is MBIST 21 */

/* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

    STCU2.MB_CTRL[21].R = 0xA6000000;       /* MBIST CTRL21 Run concurrently, next in
sequence is MBIST 22 */
    STCU2.MB_CTRL[22].R = 0xA7000000;       /* MBIST CTRL22 Run concurrently, next in
sequence is MBIST 23 */
    STCU2.MB_CTRL[23].R = 0xA8000000;       /* MBIST CTRL23 Run concurrently, next in
sequence is MBIST 24 */
    STCU2.MB_CTRL[24].R = 0xA9000000;       /* MBIST CTRL24 Run concurrently, next in
sequence is MBIST 25 */
    STCU2.MB_CTRL[25].R = 0xAA000000;       /* MBIST CTRL25 Run concurrently, next in
sequence is MBIST 26 */
    STCU2.MB_CTRL[26].R = 0xAB000000;       /* MBIST CTRL26 Run concurrently, next in
sequence is MBIST 27 */
    STCU2.MB_CTRL[27].R = 0xAC000000;       /* MBIST CTRL27 Run concurrently, next in
sequence is MBIST 28 */
    STCU2.MB_CTRL[28].R = 0xAD000000;       /* MBIST CTRL28 Run concurrently, next in
sequence is MBIST 29 */
    STCU2.MB_CTRL[29].R = 0xAE000000;       /* MBIST CTRL29 Run concurrently, next in
sequence is MBIST 30 */
    STCU2.MB_CTRL[30].R = 0xAF000000;       /* MBIST CTRL30 Run concurrently, next in
sequence is MBIST 31 */
    STCU2.MB_CTRL[31].R = 0xB0000000;       /* MBIST CTRL31 Run concurrently, next in
sequence is MBIST 32 */
    STCU2.MB_CTRL[32].R = 0xB1000000;       /* MBIST CTRL32 Run concurrently, next in
sequence is MBIST 33 */
    STCU2.MB_CTRL[33].R = 0xB2000000;       /* MBIST CTRL33 Run concurrently, next in
sequence is MBIST 34 */
    STCU2.MB_CTRL[34].R = 0xB3000000;       /* MBIST CTRL34 Run concurrently, next in
sequence is MBIST 35 */
    STCU2.MB_CTRL[35].R = 0xB4000000;       /* MBIST CTRL35 Run concurrently, next in
sequence is MBIST 36 */
    STCU2.MB_CTRL[36].R = 0xB5000000;       /* MBIST CTRL36 Run concurrently, next in
sequence is MBIST 37 */
    STCU2.MB_CTRL[37].R = 0xB6000000;       /* MBIST CTRL37 Run concurrently, next in
sequence is MBIST 38 */
    STCU2.MB_CTRL[38].R = 0xB7000000;       /* MBIST CTRL38 Run concurrently, next in
sequence is MBIST 39 */


/* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

    STCU2.MB_CTRL[39].R = 0xB8000000;       /* MBIST CTRL39 Run concurrently, next in
sequence is MBIST 40 */
    STCU2.MB_CTRL[40].R = 0xB9000000;       /* MBIST CTRL40 Run concurrently, next in
sequence is MBIST 41 */
    STCU2.MB_CTRL[41].R = 0xBA000000;       /* MBIST CTRL41 Run concurrently, next in
sequence is MBIST 42 */
    STCU2.MB_CTRL[42].R = 0xBB000000;       /* MBIST CTRL42 Run concurrently, next in
sequence is MBIST 43 */
```

```
    STCU2.MB_CTRL[43].R = 0xBC000000;       /* MBIST CTRL43 Run concurrently, next in
sequence is MBIST 44 */
    STCU2.MB_CTRL[44].R = 0xBD000000;       /* MBIST CTRL44 Run concurrently, next in
sequence is MBIST 45 */
    STCU2.MB_CTRL[45].R = 0xBE000000;       /* MBIST CTRL45 Run concurrently, next in
sequence is MBIST 46 */
    STCU2.MB_CTRL[46].R = 0x00000000;       /* MBIST CTRL46 Run sequentially, next in
sequence is LBIST 0 */

/* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

}//Online_BIST
```

A.2. STCU2_LB_CTRL configuration example

Example 2.

```
void LBIST_Config (void)
{
/* Configure Online LBIST partition coverage                              */
/* Configure LBIST for 90% coverage using KEYOFF #define          */

    /* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

    STCU2.LB[0].CTRL.R = 0x0103000F;       /* LBIST CTRL0 Run sequentially, next in sequence
is LBIST 1 */
    STCU2.LB[0].PCS.R = 0x00000866;        /* LBIST 0 pattern count */
    STCU2.LB[0].MISRELSW.R = 0xFDC31B19;   /* STCU2 On-Line LBIST MISR Expected Low Register
*/
    STCU2.LB[0].MISREHSW.R = 0x18F2A1F5;   /* STCU2 On-Line LBIST MISR Expected High Register
*/

    /* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

    STCU2.LB[1].CTRL.R = 0x0203000F;       /* LBIST CTRL1 Run sequentially, next in sequence
is LBIST 2 */
    STCU2.LB[1].PCS.R = 0x000007D0;        /* LBIST 1 pattern count */
    STCU2.LB[1].MISRELSW.R = 0xD8E3627D;   /* LBIST MISREL Expected Low */
    STCU2.LB[1].MISREHSW.R = 0xAC001775;   /* LBIST MISREH Expected High */

    /* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

    STCU2.LB[2].CTRL.R = 0x0303000F;       /* LBIST CTRL2 Run sequentially, next in sequence
is LBIST 3 */
    STCU2.LB[2].PCS.R = 0x00000834;        /* LBIST 2 pattern count */
    STCU2.LB[2].MISRELSW.R = 0x4A32B225;   /* LBIST MISREL Expected Low */
    STCU2.LB[2].MISREHSW.R = 0x765D7C3D;   /* LBIST MISREH Expected High */

    /* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;
```

```
    STCU2.LB[3].CTRL.R = 0x7F03000F;        /* LBIST CTRL3 Run sequentially */
    STCU2.LB[3].PCS.R = 0x00000ABE;         /* LBIST 3 pattern count */
    STCU2.LB[3].MISRELSW.R = 0xEED3E3B7;    /* LBIST MISREL Expected Low */
    STCU2.LB[3].MISREHSW.R = 0xB489FD55;    /* LBIST MISREH Expected High */

    /* Write key 2 to service the watchdog */
    STCU2.SKC.R = 0x8AC06DB1;

}//STCU2LBIST_Config
```

Appendix B. Off-line BIST Lauterbach script example

Example 3.

```
data.set 0x00400310 %QUAD 0xD3FEA98B00080008 ;STCU SKU Key 1
data.set 0x00400318 %QUAD 0x2C01567400080008 ;STCU SKU Key 2 - Unlocks writes to the STCU

data.set 0x00400320 %QUAD 0x100000080008000C ;STCU CFG Start with MBIST partition 0
data.set 0x00400328 %QUAD 0x0401001900080010 ;STCU PLL Settings for 100MHz from 16MHz IRC
data.set 0x00400330 %QUAD 0x02DC6C0000080014 ;STCU WDG Watchdog time out to Max value

;MBIST Config - MBIST Will run first
data.set 0x00400338 %QUAD 0x9100000000080600 ;MBIST CTRL0 Run concurrently, next in sequence
is MBIST 1
data.set 0x00400340 %QUAD 0x9200000000080604 ;MBIST CTRL1 Run concurrently, next in sequence
is MBIST 2
data.set 0x00400348 %QUAD 0x9300000000080608 ;MBIST CTRL2 Run concurrently, next in sequence
is MBIST 3
data.set 0x00400350 %QUAD 0x940000000008060C ;MBIST CTRL3 Run concurrently, next in sequence
is MBIST 4
data.set 0x00400358 %QUAD 0x9500000000080610 ;MBIST CTRL4 Run concurrently, next in sequence
is MBIST 5
data.set 0x00400360 %QUAD 0x9600000000080614 ;MBIST CTRL5 Run concurrently, next in sequence
is MBIST 6
data.set 0x00400368 %QUAD 0x9700000000080618 ;MBIST CTRL6 Run concurrently, next in sequence
is MBIST 7
data.set 0x00400370 %QUAD 0x980000000008061C ;MBIST CTRL7 Run concurrently, next in sequence
is MBIST 8
data.set 0x00400378 %QUAD 0x9900000000080620 ;MBIST CTRL8 Run concurrently, next in sequence
is MBIST 9
data.set 0x00400380 %QUAD 0x9A00000000080624 ;MBIST CTRL9 Run concurrently, next in sequence
is MBIST 10
data.set 0x00400388 %QUAD 0x9B00000000080628 ;MBIST CTRL10 Run concurrently, next in sequence
is MBIST 11
data.set 0x00400390 %QUAD 0x9C0000000008062C ;MBIST CTRL11 Run concurrently, next in sequence
is MBIST 12
data.set 0x00400398 %QUAD 0x9D00000000080630 ;MBIST CTRL12 Run concurrently, next in sequence
is MBIST 13
data.set 0x004003A0 %QUAD 0x9E00000000080634 ;MBIST CTRL13 Run concurrently, next in sequence
is MBIST 14
data.set 0x004003A8 %QUAD 0x9F00000000080638 ;MBIST CTRL14 Run concurrently, next in sequence
is MBIST 15
data.set 0x004003B0 %QUAD 0xA000000000008063C ;MBIST CTRL15 Run concurrently, next in sequence
is MBIST 16
data.set 0x004003B8 %QUAD 0xA100000000080640 ;MBIST CTRL16 Run concurrently, next in sequence
is MBIST 17
data.set 0x004003C0 %QUAD 0xA200000000080644 ;MBIST CTRL17 Run concurrently, next in sequence
is MBIST 18
data.set 0x004003C8 %QUAD 0xA300000000080648 ;MBIST CTRL18 Run concurrently, next in sequence
is MBIST 19
data.set 0x004003D0 %QUAD 0xA400000000008064C ;MBIST CTRL19 Run concurrently, next in sequence
is MBIST 20
data.set 0x004003D8 %QUAD 0xA500000000080650 ;MBIST CTRL20 Run concurrently, next in sequence
is MBIST 21
```

```
data.set 0x004003E0 %QUAD 0xA600000000080654 ;MBIST CTRL21 Run concurrently, next in sequence
is MBIST 22
data.set 0x004003E8 %QUAD 0xA700000000080658 ;MBIST CTRL22 Run concurrently, next in sequence
is MBIST 23

;Write SKU Key 2 again to service register Watchdog
data.set 0x004003F0 %QUAD 0x2C01567400080008

;MBIST Config-Cont
data.set 0x004003F8 %QUAD 0xA80000000008065C ;MBIST CTRL23 Run concurrently, next in sequence
is MBIST 24
data.set 0x00400400 %QUAD 0xA900000000080660 ;MBIST CTRL24 Run concurrently, next in sequence
is MBIST 25
data.set 0x00400408 %QUAD 0xAA00000000080664 ;MBIST CTRL25 Run concurrently, next in sequence
is MBIST 26
data.set 0x00400410 %QUAD 0xAB00000000080668 ;MBIST CTRL26 Run concurrently, next in sequence
is MBIST 27
data.set 0x00400418 %QUAD 0xAC0000000008066C ;MBIST CTRL27 Run concurrently, next in sequence
is MBIST 28
data.set 0x00400420 %QUAD 0xAD00000000080670 ;MBIST CTRL28 Run concurrently, next in sequence
is MBIST 29
data.set 0x00400428 %QUAD 0xAE00000000080674 ;MBIST CTRL29 Run concurrently, next in sequence
is MBIST 30
data.set 0x00400430 %QUAD 0xAF00000000080678 ;MBIST CTRL30 Run concurrently, next in sequence
is MBIST 31
data.set 0x00400438 %QUAD 0xB000000000008067C ;MBIST CTRL31 Run concurrently, next in sequence
is MBIST 32
data.set 0x00400440 %QUAD 0xB100000000080680 ;MBIST CTRL32 Run concurrently, next in sequence
is MBIST 33
data.set 0x00400448 %QUAD 0xB200000000080684 ;MBIST CTRL33 Run concurrently, next in sequence
is MBIST 34
data.set 0x00400450 %QUAD 0xB300000000080688 ;MBIST CTRL34 Run concurrently, next in sequence
is MBIST 35
data.set 0x00400458 %QUAD 0xB400000000008068C ;MBIST CTRL35 Run concurrently, next in sequence
is MBIST 36
data.set 0x00400460 %QUAD 0xB500000000080690 ;MBIST CTRL36 Run concurrently, next in sequence
is MBIST 37
data.set 0x00400468 %QUAD 0xB600000000080694 ;MBIST CTRL37 Run concurrently, next in sequence
is MBIST 38
data.set 0x00400470 %QUAD 0xB700000000080698 ;MBIST CTRL38 Run concurrently, next in sequence
is MBIST 39
data.set 0x00400478 %QUAD 0xB800000000008069C ;MBIST CTRL39 Run concurrently, next in sequence
is MBIST 40
data.set 0x00400480 %QUAD 0xB9000000000806A0 ;MBIST CTRL40 Run concurrently, next in sequence
is MBIST 41
data.set 0x00400488 %QUAD 0xBA000000000806A4 ;MBIST CTRL41 Run concurrently, next in sequence
is MBIST 42
data.set 0x00400490 %QUAD 0xBB000000000806A8 ;MBIST CTRL42 Run concurrently, next in sequence
is MBIST 43
data.set 0x00400498 %QUAD 0xBC000000000806AC ;MBIST CTRL43 Run concurrently, next in sequence
is MBIST 44
data.set 0x004004A0 %QUAD 0xBD000000000806B0 ;MBIST CTRL44 Run concurrently, next in sequence
is MBIST 45
data.set 0x004004A8 %QUAD 0xBE000000000806B4 ;MBIST CTRL45 Run concurrently, next in sequence
is MBIST 46
```

```
data.set 0x004004B0 %QUAD 0x00000000000806B8 ;MBIST CTRL46 Run sequentialy,  next in sequence
is LBIST 0

;Write SKU Keys again to service register Watchdog
data.set 0x004004B8 %QUAD 0x2C01567400080008

;LBIST Config for 90% Coverage
data.set 0x004004C0 %QUAD 0x0102000F00080100 ;LBIST CTRL0 Run concurrently, next in sequence
is LBIST 1 shift =pll_cfg/3
data.set 0x004004C8 %QUAD 0x0000086600080104 ;LBIST 0 pattern count -2150
data.set 0x004004D0 %QUAD 0xFDC31B1900080110 ;LBIST MISREL Expected Low
data.set 0x004004D8 %QUAD 0x18F2A1F500080114 ;LBIST MISREH Expected High

;LBIST1
data.set 0x004004E0 %QUAD 0x0202000F00080140 ;LBIST CTRL1 Run concurrently, next in sequence
is LBIST 2  shift =pll_cfg/3
data.set 0x004004E8 %QUAD 0x000007D000080144 ;LBIST 1 pattern count-2000
data.set 0x004004F0 %QUAD 0xD8E3627D00080150 ;LBIST MISREL Expected Low
data.set 0x004004F8 %QUAD 0xAC00177500080154 ;LBIST MISREH Expected High

;LBIST2
data.set 0x00400500 %QUAD 0x0302000F00080180 ;LBIST CTRL2 Run concurrently, next in sequence
is LBIST 3  shift =pll_cfg/3
data.set 0x00400508 %QUAD 0x0000083400080184 ;LBIST 2 pattern count -2100
data.set 0x00400510 %QUAD 0x4A32B22500080190 ;LBIST MISREL Expected Low
data.set 0x00400518 %QUAD 0x765D7C3D00080194 ;LBIST MISREH Expected High

;LBIST3
data.set 0x00400520 %QUAD 0x7F02000F000801c0 ;LBIST CTRL3 End of BIST Sequence  shift
=pll_cfg/3
data.set 0x00400528 %QUAD 0x00000ABE000801c4 ;LBIST 3 pattern count-2750
data.set 0x00400530 %QUAD 0xEED3E3B7000801d0 ;LBIST MISREL Expected Low
data.set 0x00400538 %QUAD 0xB489FD55000801d4 ;LBIST MISREH Expected High

;Configure all ips DCF Clients - Tripple Voting - To be updated with better comments

;DCF_SELFTEST_CONFIG_IPS_1 DCF client
;PLL_LOCK0 = 1 | config[1] = 1
data.set 0x00400540 %QUAD 0x0011100102000024 ;DCF_IPS_1 for Config1 clock configuration
data.set 0x00400548 %QUAD 0xFFEEEFFE02000028
data.set 0x00400550 %QUAD 0x8008880002000030

;DCF_SELFTEST_CONFIG_IPS_2 DCF client
data.set 0x00400558 %QUAD 0x0000001002000044 ;DCF_IPS_2 STCU_CLK_PLL0_SRC_OSC
data.set 0x00400560 %QUAD 0xFFFFFFEF02000048
data.set 0x00400568 %QUAD 0x0000000802000050

;DCF_SELFTEST_CONFIG_IPS_3 DCF client
data.set 0x00400570 %QUAD 0x0000000002000064 ;DCF_IPS_3
data.set 0x00400578 %QUAD 0xFFFFFFFF02000068
data.set 0x00400580 %QUAD 0x0000000002000070

;DCF_SELFTEST_CONFIG_IPS_4 DCF client
data.set 0x00400588 %QUAD 0x00FB7C3F02000084 ;DCF_IPS_4
data.set 0x00400590 %QUAD 0xFF0483C002000088
```

```
data.set 0x00400598 %QUAD 0x0009078002000090

;START OFFLINE BIST ( when SSCM reach this DCF record the BIST is started)
data.set 0x004005A0 %QUAD 0x0000030100080000 ;STCU RUN Set RUN for offline BIST enable PLL
for MBIST and LBIST

data.set 0x004005A8 %QUAD 0x0000000000080028  ;STCU2_ERR_FM, All Recoverable Fault Mapping
data.set 0x004005B0 %QUAD 0x0000000000080040  ;STCU2_LBUFM,  All Recoverable Fault Mapping
data.set 0x004005B8 %QUAD 0x0000000000080074  ;STCU2_MBUFML, All Recoverable Fault Mapping
data.set 0x004005C0 %QUAD 0x0000000000080078  ;STCU2_MBUFMM, All Recoverable Fault Mapping
```

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN5427
Rev. 1.0.1
10/2016