NXP Semiconductors
Application Note

Document Number: AN11993
Rev. 0, June 2017

Using the Built-in Self-Test (BIST) on the MPC5744P

by: NXP Semiconductors

Contents

1 Introduction..... 1

2 Objective..... 1

3 Overview of Built-In Self-Test (BIST)...2
3.1 LBIST and MBIST.....2
3.2 Offline and online BIST overview 2

4 Self-Test Control Unit..... 3

5 MBIST and LBIST testing and partitions..... 3
5.1 LBIST testing..... 3
5.2 LBIST partitions.....3
5.3 LBIST configuration..... 5
5.4 MBIST testing 6
5.5 MBIST partitions.....7
5.6 MBIST test cycles..... 8

6 DCF records, clients, and configuration..... 10

7 Online BIST configuration..... 11
7.1 Preparation before initiating an STCU online self-test..... 11
7.2 Online self-test initiation procedure..... 14
7.3 Online BIST execution flow..... 15

8 Offline BIST Configuration..... 16
8.1 Normal start-up self-test.. 16
8.2 No self-test execution..... 16

9 BIST results.....17

10 Definitions, acronyms, and abbreviations.....17

11 References.....17

1 Introduction

The MPC5744P device targets chassis and safety applications which require a high Automotive Safety Integrity Level (SIL). All devices in this family are built around a safety concept based on delayed lock step, targeting an ISO26262 ASIL-D (Design) integrity level.

A requirement of the standard is to detect the accumulation of latent defects. To meet this requirement, the MPC5744P has the ability to execute Built-In Self-Test (BIST) procedures.

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 a part of this program, NXP provides an MPC5744P safety manual to advice users on how to configure the MPC5744P to obtain ISO26262 ASIL-D compliance.

2 Objective

This application note provides an introduction to BIST on the MPC5744P and explains how to configure and use the offline and online BIST features of the MPC5744P. After reading this application note, the user should:

- Understand the BIST features that are available on the MPC5744P.
- Understand the difference between MBIST (Memory Build-in-self-test) and LBIST (Logic Build-in-self-test) .
- Understand the difference between online and offline BIST.
- Be able to develop application strategies for deploying online BIST testing.
- Be able to develop application strategies for deploying offline BIST testing.
- Understand what is included in each BIST partition on the MPC5744P.
- Understand when offline and online BIST takes place.
- Be able to develop a specified configuration for offline and online BIST testing and understand how to use Device Configuration Format (DCF) clients to achieve this.

- Be able to invoke the desired offline and online BIST sequence using the Self-Test Control Unit (STCU2).

The application note also provides examples for offline and online self-test configurations can be directly implemented by the user. The examples are provided in the software package accompanying this document and is also explained in detail.

To aid in understanding of this document and software package, the reader should obtain the MPC5744P reference and Safety manuals from the NXP website.

3 Overview of Built-In Self-Test (BIST)

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 Microcontroller Unit (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 MPC5744P 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 such as 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 Offline and online BIST overview

It is anticipated that the MPC5744P 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 online

For offline 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 online testing.

The intended usage of online 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. online 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 online 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 online 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 online 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 online tests, or check the results of the offline tests.

For offline 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 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.

NOTE

In MPC5744P reference manual, BISTs are named as follows:

Startup BIST = Offline BIST

Shutdown BIST = Online BIST

5 MBIST and LBIST testing and partitions

This section details the partitioning of the memory and logic on the MPC5744P.

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 always generates 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.

NOTE

The expected values provided with this application note apply only to the **0N15P**, **1N15P** and **1N65H** mask set of the MPC574xP device if it is not stated otherwise.

Table 1. LBIST partitions

LBIST number	Partition	IP module
0	C0	z4 Main Core_0
0	C0	PFLASHC
0	C0	Flash memory
0	C0	NPC
0	C0	NXMC0 and NXMC1
0	C0	MEMU
0	C0	XBAR
0	C0	DSMC
0	C0	DMA
0	C0	EIM
0	C0	INTC
0	C0	NAL
0	C0	NAP
0	C0	PRAMC
0	C0	SWT
0	C0	SMPU
0	C0	STM
1	C1	z4 Checker Core_0s
1	C1	CMU1
1	C1	RCCU
2	P0	DMA_CH_MUX0
2	P0	FlexRay
2	P0	WKPU
2	P0	MC_ME
2	P0	MC_PCU
2	P0	SIUL2
2	P0	IO-muxing logic
2	P0	CRC
2	P0	PIT
2	P0	DSPI0 and DSPI1
2	P0	LIN1

Table continues on the next page...

Table 1. LBIST partitions (continued)

LBIST number	Partition	IP module
2	P0	FlexPWM1
2	P0	eTimer1
2	P0	CTU1
2	P0	SENT0
2	P0	SIPI
2	P0	LFAST
2	P0	BAM
2	P0	FlexCAN0, FlexCAN1, and FlexCAN2
2	P0	ADC1 and ADC3
2	P0	ENET
3	P1	DMA_CH_MUX1
3	P1	FCCU
3	P1	FOSU
3	P1	CMU0, CMU2, CMU3, and CMU4
3	P1	DSPI2 and DSPI3
3	P1	SGEN
3	P1	LIN0
3	P1	FlexPWM0
3	P1	eTimer0 and eTimer2
3	P1	CTU0
3	P1	SENT1
3	P1	ADC0 and ADC2

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 for cut 2.1 (1N65H)

Partition	Clock Config	PRPG SEED	MISR SEED	Number of Patterns	Number of Shifts	Expected MISR	Coverage
C0	1	0x3FFFFFFFFF	ffffffffffffffff	2700	65	64'hF80DA87A23449471	90%
C1	1	0x3FFFFF		1850	50	64'h8C363533900F8372	90%
P0	1	0x3FFFFFFFFF		2816	61	64'h2D2A280DD1933F3A	90%
P1	1	0x3FFFFFF		1856	50	64'hAC087BB45D1E5519	90%

Table 3. LBIST configuration by partition and expected MISR for cut 2.1b (0N15P) and newer

Partition	Clock Config	PRPG SEED	MISR SEED	Number of Patterns	Number of Shifts	Expected MISR	Coverage
C0	1	0x3FFFFFFFFF	ffffffffffffffff	2650	65	64'hBD3805126D73AE0D	90%
C1	1	0x3FFFFF		1344	50	64'h18916197F1B392F7	90%
P0	1	0x3FFFFFFFFF		2900	61	64'hFA8FB16EC97B3FA7	90%
P1	1	0x3FFFFFF		1900	50	64'h108069355D01B854	90%

For LBIST online 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 C0
- LBRMSW1: corresponds to LBIST C1
- LBRMSW2: corresponds to LBIST P0
- LBRMSW3: corresponds to LBIST P1

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 MPC5744P memory is split into 27 different MBIST partitions which are numbered 0 – 26.

MBIST contains these three types of tests:

- Full Test – Diagnostic test
- Reduced Test – Long self-test mode
- Auto test – Normal self-test mode

NOTE

Always refer to current reference manual for details about types of tests.

The MBIST controller has three types of March tests. It can apply those tests while 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. Therefore, this mode is recommended for the offline 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 online BIST since time constraints are typically not as critical in the user's online use case.

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

Table 5. MBIST mapping

Partition Number	Memory	Corresponding MBIST Control Register
0	BAM ROM	STCU_MB_CTRL_0
1	DMA RAM	STCU_MB_CTRL_1
2	System RAM 1	STCU_MB_CTRL_2
3	System RAM 0	STCU_MB_CTRL_3

Table continues on the next page...

Table 5. MBIST mapping (continued)

Partition Number	Memory	Corresponding MBIST Control Register
4	System RAM 3	STCU_MB_CTRL_4
5	System RAM 2	STCU_MB_CTRL_5
6	System RAM 5	STCU_MB_CTRL_6
7	System RAM 4	STCU_MB_CTRL_7
8	DLMEM 1	STCU_MB_CTRL_8
9	DLMEM 0	STCU_MB_CTRL_9
10	iCache Data Memory 3	STCU_MB_CTRL_10
11	iCache Data Memory 2	STCU_MB_CTRL_11
12	iCache Data Memory 1	STCU_MB_CTRL_12
13	iCache Data Memory 0	STCU_MB_CTRL_13
14	iCache Tag Memory	STCU_MB_CTRL_14
15	dCache Data Memory 3	STCU_MB_CTRL_15
16	dCache Data Memory 2	STCU_MB_CTRL_16
17	dCache Data Memory 1	STCU_MB_CTRL_17
18	dCache Data Memory 0	STCU_MB_CTRL_18
19	dCache Tag Memory	STCU_MB_CTRL_19
20	FlexCAN RAM 2	STCU_MB_CTRL_20
21	FlexCAN RAM 1	STCU_MB_CTRL_21
22	FlexCAN RAM 0	STCU_MB_CTRL_22
23	FlexRay LRAM 1	STCU_MB_CTRL_23
24	FlexRay LRAM 0	STCU_MB_CTRL_24
25	FlexRay DRAM	STCU_MB_CTRL_25
26	ENET Memory	STCU_MB_CTRL_26

5.6 MBIST test cycles

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

Partition Number	Memory	Size	Full Test		Reduced Test		Auto Test	
			TCKs	MEM Clks	TCKs	MEM Clks	TCKs	MEM Clks
0	BAM ROM	2048x32	22260	20139	22260	17545	22260	20139
1	DMA RAM	128x72	22260	17449	22260	17409	22260	34898
2	System RAM 0 L2	8192x72	22260	275957	22260	265337	22260	551914
3	System RAM 0 L1	8192x72	22260	275957	22260	265337	22260	551914
4	System RAM 1 L2	8192x72	22260	275957	22260	265337	22260	551914
5	System RAM 1 L1	8192x72	22260	275957	22260	265337	22260	551914
6	System RAM 2 L2	8192x72	22260	275957	22260	265337	22260	551914
7	System RAM 2 L2	8192x72	22260	275957	22260	265337	22260	551914
8	DLMEM L2	8192x40	22260	144704	22260	139352	22260	578816
9	DLMEM L1	8192x40	22260	144704	22260	139352	22260	578816
10	iCache Data Memory L4	256x72	22260	17696	22260	17336	22260	70784
11	iCache Data Memory L3	256x72	22260	17696	22260	17336	22260	70784
12	iCache Data Memory L2	256x72	22260	17696	22260	17336	22260	70784
13	iCache Data Memory L1	256x72	22260	17696	22260	17336	22260	70784
14	iCache Tag Memory	128x65	22260	15600	22260	15368	22260	62400

Table continues on the next page...

Table continued from the previous page...

Partition Number	Memory	Size	Full Test		Reduced Test		Auto Test	
			TCKs	MEM Clks	TCKs	MEM Clks	TCKs	MEM Clks
15	dCache Data Memory L4	128x80	22260	15600	22260	15368	22260	62400
16	dCache Data Memory L3	128x80	22260	15600	22260	15368	22260	62400
17	dCache Data Memory L2	128x80	22260	15600	22260	15368	22260	62400
18	dCache Data Memory L1	128x80	22260	15600	22260	15368	22260	62400
19	dCache Tag Memory	64x71	22260	14560	22260	14384	22260	58240
20	FlexCAN RAM L3	192x104	22260	25456	22260	25612	22260	26456
21	FlexCAN RAM L2	192x104	22260	25456	22260	25612	22260	26456
22	FlexCAN RAM L1	192x104	22260	25456	22260	25612	22260	26456
23	FlexRay LRAM L2	64x63	22260	17548	22260	17548	22260	17548
24	FlexRay LRAM 0 L1	64x63	22260	17548	22260	17548	22260	17548
25	FlexRay DRAM	128x26	22260	18794	22260	18401	22260	37588
26	ENET Memory	512x66	22260	30389	22260	29217	22260	60778

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. Reset sequence on page 11 represents microcontroller execution flow during preset phases.

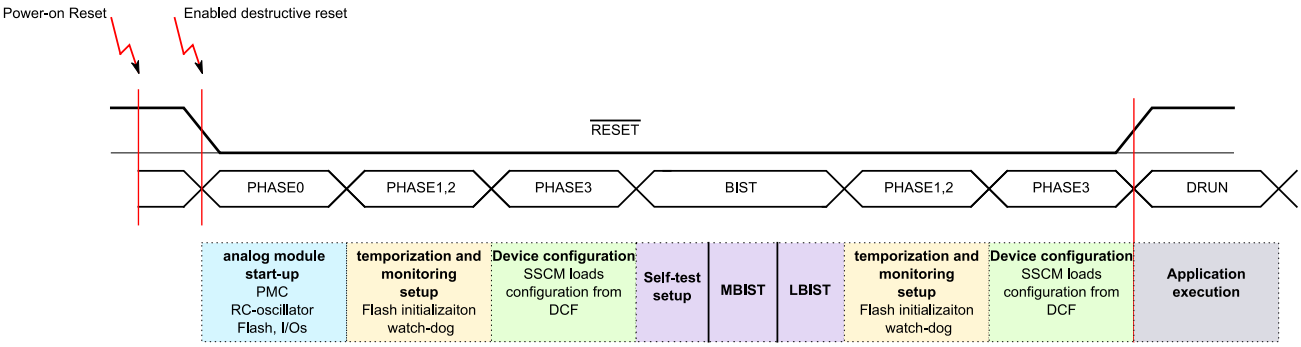

Figure 1. Reset sequence

The offline BIST tests are configured using DCF records that automatically configure the STCU2 to enable and run BIST at start-up or following a destructive reset or long functional 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 of which 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.

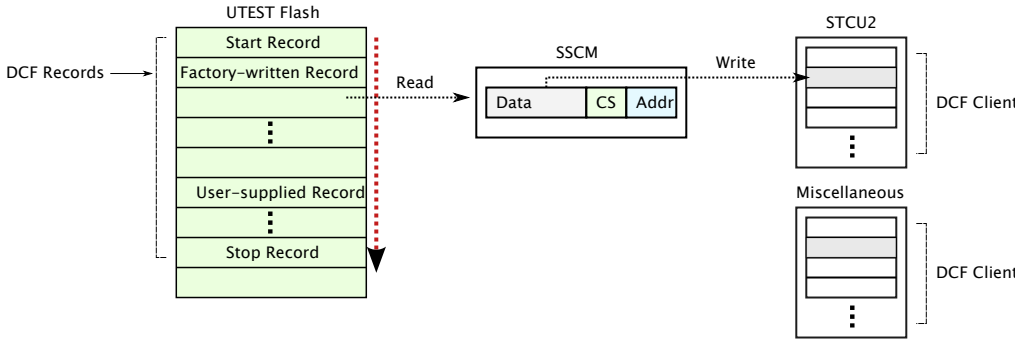

Figure 2. DCF record mechanism

7 Online BIST configuration

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

7.1 Preparation before initiating an STCU online self-test

Before start of an STCU online self-test, it is required to complete the following prerequisite procedure to prepare the chip:

1. End the currently running application, including smooth termination of peripherals as required by the application.
2. Configure FlexRay, CAN clock sources, and divider for self-test:
   a. Configure FRAY_PLL_CLK clock as PLL0 PHI divided by 2: MC_CGM_AC1_DC0[DIV]=1h.

```
MC_CGM.AC1_DC0.R = 0x80000000 | 0x10000;
```

   b. Select FRAY_PLL_CLK as the FRAY_CLK clock source: FR_MCR[CLKSEL]=1.

```
FR_0.MCR.B.CLKSEL = 1;
```

c. Select CAN_PLL_CLK as the CAN_CLK clock source: CAN_CTRL1[CLKSRC]=1.

```
CAN_0.CTRL1.B.CLKSRC=1;
```

3. Ensure PBRIDGE clock is not over clocked (maximum frequency is 50MHz) when SYS_CLK is switched to 200MHz.

```
MC_CGM.SC_DC0.R = 0x80000000 | 0x3;
```

4. Configure DRUN to turn off PLL1 (to save power), turn off PLL0 so PLL0 can be re-configured for the required 200MHz self-test clock frequency, turn on the XOSC as it is needed to provide the PLL0 reference clock during self-test, and select IRCOSC to source SYS_CLK:

a. Select IRCOSC as the SYS_CLK clock source: MC_ME_DRUN_MC[SYSCLK]=0b.

```
MC_ME.DRUN_MC.B.SYSCLK = 0;
```

b. Enable XOSC: MC_ME_DRUN_MC[XOSCON]=1 (will be required for PLL0).

```
MC_ME.DRUN_MC.B.XOSCON = 1;
```

c. Disable PLL0: MC_ME_DRUN_MC[PLL0ON]=0.

```
MC_ME.DRUN_MC.B.PLL0ON = 0;
```

d. Disable PLL1: MC_ME_DRUN_MC[PLL1ON]=0.

```
MC_ME.DRUN_MC.B.PLL1ON = 0;
```

e. Gate off all peripheral clocks because they should not be running: MC_ME_RUN_PC0[DRUN]=0, MC_ME_PCTLn[RUN_CFG]=0b.

```
MC_ME.RUN_PC[0].B.DRUN = 0;
MC_ME.PCTL9.B.RUN_CFG = 0;       /* LFAST_0 Peripheral Control Register*/
MC_ME.PCTL11.B.RUN_CFG = 0;      /* SIPI_0 Peripheral Control Register */
MC_ME.PCTL12.B.RUN_CFG = 0;      /* ENET_0 Peripheral Control Register */
MC_ME.PCTL30.B.RUN_CFG = 0;      /* PIT_0 Peripheral Control Register */
MC_ME.PCTL36.B.RUN_CFG = 0;      /* DMAMUX_0 Peripheral Control Register */
MC_ME.PCTL38.B.RUN_CFG = 0;      /* CRC_0 Peripheral Control Register */
MC_ME.PCTL77.B.RUN_CFG= 0;       /* CAN_2 Peripheral Control Register */
MC_ME.PCTL78.B.RUN_CFG = 0;      /* CAN_1 Peripheral Control Register */
MC_ME.PCTL79.B.RUN_CFG= 0;       /* CAN_0 Peripheral Control Register */
MC_ME.PCTL91.B.RUN_CFG= 0;       /* LINFlex_1 Peripheral Control Register */
MC_ME.PCTL98.B.RUN_CFG= 0;       /* DSPI_1 Peripheral Control Register */
MC_ME.PCTL99.B.RUN_CFG= 0;       /* DSPI_0 Peripheral Control Register */
MC_ME.PCTL104.B.RUN_CFG= 0;      /* SENT_0 Peripheral Control Register */
MC_ME.PCTL107.B.RUN_CFG= 0;      /* FLEXRAY Peripheral Control Register */
MC_ME.PCTL124.B.RUN_CFG= 0;      /* ADC_3 Peripheral Control Register */
MC_ME.PCTL126.B.RUN_CFG= 0;      /* ADC_1 Peripheral Control Register */
MC_ME.PCTL137.B.RUN_CFG= 0;      /* ETIMER_1 Peripheral Control Register */
MC_ME.PCTL141.B.RUN_CFG= 0;      /* CTU_1 Peripheral Control Register */
MC_ME.PCTL144.B.RUN_CFG= 0;      /* PWM_1 Peripheral Control Register */
MC_ME.PCTL146.B.RUN_CFG= 0;      /* DMAMUX_1 Peripheral Control Register */
MC_ME.PCTL204.B.RUN_CFG= 0;      /* LINFlex_0 Peripheral Control Register */
MC_ME.PCTL208.B.RUN_CFG= 0;      /* DSPI_3 Peripheral Control Register */
MC_ME.PCTL209.B.RUN_CFG= 0;      /* DSPI_2 Peripheral Control Register */
MC_ME.PCTL214.B.RUN_CFG= 0;      /* SENT_1 Peripheral Control Register */
MC_ME.PCTL235.B.RUN_CFG= 0;      /* ADC_2 Peripheral Control Register */
MC_ME.PCTL237.B.RUN_CFG= 0;      /* ADC_0 Peripheral Control Register */
```

```
        MC_ME.PCTL239.B.RUN_CFG= 0;      /* SGEN_0 Peripheral Control Register */
        MC_ME.PCTL245.B.RUN_CFG= 0;       /* ETIMER_2 Peripheral Control Register */
        MC_ME.PCTL247.B.RUN_CFG= 0;       /* ETIMER_0 Peripheral Control Register */
        MC_ME.PCTL251.B.RUN_CFG= 0;       /* CTU_0 Peripheral Control Register */
        MC_ME.PCTL255.B.RUN_CFG= 0;      /* PWM_0 Peripheral Control Register*/
```

5. Activate the new configuration by performing mode change to DRUN: MC_ME_MCTL[TARGET_MODE]=0011b (see the MC_ME chapter for mode change request details).

```
     /* Mode transition to apply the IRC setup and set Normal mode with PLL */
      MC_ME.MCTL.R = 0x30005AF0;                              //DRUN Mode & Key
      MC_ME.MCTL.R = 0x3000A50F;                              //DRUN Mode & Key
      while(MC_ME.GS.B.S_MTRANS);                             //Waiting for end of transaction
      while(MC_ME.GS.B.S_CURRENT_MODE != DRUN_MODE);        // ME_GS Check DRUN mode has
successfully been entered
```

6. Configure PLL0 for the required 200MHz self-test frequency:
   a. Configure PLL0 to provide 200MHz at its PHI output: PLLDIG_PLL0DV (values depend on XOSC frequency).

```
      PLLDIG.PLL0CR.B.CLKCFG = 1;          //Bypass mode PLL0 on
      // RFDPHI1 = 10, RFDPHI = 2, PREDIV =2, MFD = 14
      PLLDIG.PLL0DV.R = 0x50000000 |0x00020000 |0x00002000|0x0014;
```

   b. Select XOSC as PLL0's reference clock: MC_CGM_AC3_SC[SELCTL]=1.

```
     MC_CGM.AC3_SC.B.SELCTL =1;     //connect (8..40MHz) XTALL to the PLL0
```

7. Configure DRUN mode to turn on PLL0 and select PLL0 PHI to source SYS_CLK:
   a. Enable XOSC: MC_ME_DRUN_MC[XOSCON]=1.

```
     MC_ME.DRUN_MC.B.XOSCON = 1;
```

   b. Enable PLL0: MC_ME_DRUN_MC[PLL0ON]=1.

```
     MC_ME.DRUN_MC.B.PLL0ON = 1;
```

   c. Disable PLL1: MC_ME_DRUN_MC[PLL1ON]=0.

```
     MC_ME.DRUN_MC.B.PLL1ON = 0;
```

   d. Select PLL0 PHI as the SYS_CLK clock source: MC_ME_DRUN_MC[SYSCLK]=0010b.

```
     MC_ME.DRUN_MC.B.SYSCLK = 2;
```

8. Activate the new configuration by performing mode change to DRUN: MC_ME_MCTL[TARGET_MODE]=0011b (see the MC_ME chapter for mode change request details).

```
       MC_ME.MCTL.R = 0x30005AF0;                       //DRUN Mode & Key
       MC_ME.MCTL.R = 0x3000A50F;                      //DRUN Mode & Key
       while(!MC_ME.GS.B.S_PLL0);                      //ME_GS Wait for PLL stabilization.
       while(MC_ME.GS.B.S_MTRANS);                     //Waiting for end of transaction
       while(MC_ME.GS.B.S_CURRENT_MODE != DRUN_MODE);  // ME_GS Check DRUN mode has
successfully been entered
```

9. Select PLL0 PHI to source all auxiliary clocks used during self-test:

a. Select PLL0 PHI as the MOTC_CLK, SGEN_CLK, and ADC_CLK clock source: MC_CGM_AC0_SC[SELCTL]=2h.

```
MC_CGM.AC0_SC.B.SELCTL =0x2; //connect PLL0 to AXU_0
```

b. Select PLL0 PHI as the LFAST PLL clock source: MC_CGM_AC5_SC[SELCTL]=2h.

```
MC_CGM.AC5_SC.B.SELCTL =0x2; //connect PLL0 to AXU_5
```

7.2 Online self-test initiation procedure

To initiate a shutdown self-test, execute the following steps for the desired self-test mode. All accesses must be processed without interruption.

1. Configure the STCU_SKC register by first writing the key1 value and then writing the key2 value.

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

2. Configure the STCU_MBn_CTRL registers.

```
STCU.MB_CTRL[0].R = 0x91000000;       /* MBIST CTRL00, next in sequence is MBIST 1*/
STCU.MB_CTRL[1].R = 0x98000000;       /* MBIST CTRL01, next in sequence is MBIST 8*/
STCU.MB_CTRL[2].R = 0x93000000;       /* MBIST CTRL02, next in sequence is MBIST 3*/
STCU.MB_CTRL[3].R = 0x94000000;       /* MBIST CTRL03, next in sequence is MBIST 4*/
STCU.MB_CTRL[4].R = 0x95000000;       /* MBIST CTRL04, next in sequence is MBIST 5*/
STCU.MB_CTRL[5].R = 0x96000000;       /* MBIST CTRL05, next in sequence is MBIST 6*/
STCU.MB_CTRL[6].R = 0x97000000;       /* MBIST CTRL06, next in sequence is MBIST 7*/
STCU.MB_CTRL[7].R = 0x10000000;       /* MBIST CTRL07, next in sequence is MBIST 16*/
STCU.MB_CTRL[8].R = 0x99000000;       /* MBIST CTRL08, next in sequence is MBIST 9*/
STCU.MB_CTRL[9].R = 0x9A000000;       /* MBIST CTRL09, next in sequence is MBIST 10*/
STCU.MB_CTRL[10].R = 0x9B000000;      /* MBIST CTRL10, next in sequence is MBIST 11*/
STCU.MB_CTRL[11].R = 0x9C000000;      /* MBIST CTRL11, next in sequence is MBIST 12*/
STCU.MB_CTRL[12].R = 0x9D000000;      /* MBIST CTRL12, next in sequence is MBIST 13*/
STCU.MB_CTRL[13].R = 0x9E000000;      /* MBIST CTRL13, next in sequence is MBIST 14*/
STCU.MB_CTRL[14].R = 0x9F000000;      /* MBIST CTRL14, next in sequence is MBIST 15*/
STCU.MB_CTRL[15].R = 0xA0000000;      /* MBIST CTRL15, next in sequence is MBIST 16*/
STCU.MB_CTRL[16].R = 0xA1000000;      /* MBIST CTRL16, next in sequence is MBIST 17*/
STCU.MB_CTRL[17].R = 0xA2000000;      /* MBIST CTRL17, next in sequence is MBIST 18*/
STCU.MB_CTRL[18].R = 0xA3000000;      /* MBIST CTRL18, next in sequence is MBIST 19*/
STCU.MB_CTRL[19].R = 0xA4000000;      /* MBIST CTRL19, next in sequence is MBIST 20*/
STCU.MB_CTRL[20].R = 0xA5000000;      /* MBIST CTRL20, next in sequence is MBIST 21*/
/* Write key 2 to service the watchdog */
STCU.SKC.R = 0x8AC06DB1;
STCU.MB_CTRL[21].R = 0xA6000000;      /* MBIST CTRL21, next in sequence is MBIST 22*/
STCU.MB_CTRL[22].R = 0xA7000000;      /* MBIST CTRL22, next in sequence is MBIST 23*/
STCU.MB_CTRL[23].R = 0xA8000000;      /* MBIST CTRL23, next in sequence is MBIST 24*/
STCU.MB_CTRL[24].R = 0xA9000000;      /* MBIST CTRL24, next in sequence is MBIST 25*/
STCU.MB_CTRL[25].R = 0xAA000000;      /* MBIST CTRL25, next in sequence is MBIST 26*/
STCU.MB_CTRL[26].R = 0x00000000;      /* MBIST CTRL26, next in sequence is MBIST 0 */
```

3. Configure the STCU_CFG register.

```
/* Auto test - Normal self-test mode */
STCU.CFG.R = 0x12000008;
```

4. Configure all STCU_LBn_CTRL and STCU_LBn_PCS registers.

```
STCU.LB[0].CTRL.R = 0x83031107;
STCU.LB[0].PCS.R = 0x00006978;

STCU.LB[1].CTRL.R = 0x82031107;
STCU.LB[1].PCS.R = 0x00000708;

STCU.LB[2].CTRL.R = 0x7F031107;
STCU.LB[2].PCS.R = 0x00000B00;

STCU.LB[3].CTRL.R = 0x01031107;
STCU.LB[3].PCS.R = 0x00000740;
```

5. Configure all STCU_LBn_MISRELSW and STCU_LBn_MISREHSW registers according to used cut of device.

```
/* valid for cut 1N15P and 0N15P */
STCU.LB[0].MISRELSW.R = 0x6D73AE0D;
STCU.LB[0].MISREHSW.R = 0xBD380512;
STCU.LB[1].MISRELSW.R = 0xF1B392F7;
STCU.LB[1].MISREHSW.R = 0x18916197;
STCU.LB[2].MISRELSW.R = 0xC97B3FA7;
STCU.LB[2].MISREHSW.R = 0xFA8FB16E;
STCU.LB[3].MISRELSW.R = 0x5D01B854;
STCU.LB[3].MISREHSW.R = 0x10806935;
```

6. Configure the STCU_WDG register.

```
/* Auto test - Normal self-test mode */
STCU.WDG.R = 0xFFFFFFFF; /* max value */
```

7. Configure the STCU_LBRMSW register.

```
STCU.LBRMSW.R = 0x0000000F;
```

8. Configure the DCL_IPS0 register.

```
DCL.IPS0.R = 0x03008214;
```

9. Configure the STCU_RUNSW register. This access immediately initiates the shutdown self-test execution.

```
STCU.RUNSW.R = 0x00000301;
```

7.3 Online BIST execution flow

The configured online BIST configuration executes as the following figure represents:

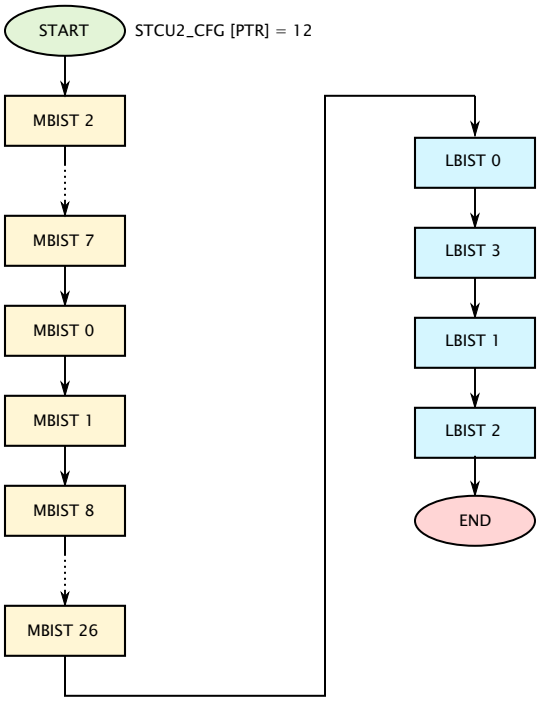

Figure 3. Online BIST execution flow

8 Offline BIST Configuration

To successfully perform offline BIST, the correct configuration must be done before BIST execution. The default STCU2 configuration is already pre-programmed by NXP during testing of the production devices. User can select from two following available modes:

1. Normal start-up self-test
2. No self-test execution

8.1 Normal start-up self-test

This configuration is already present in the device and offline BIST is by default enabled and running. The diagram given above describes BIST flow already programed by NXP.

8.2 No self-test execution

To bypass the BIST execution, it is necessary to program following STCU2 DCF client in UTEST user memory area. This sets the BIST pointer in STCU2 CFG register to 0x7F which represents no BIST execution.

Bypass BIST DCF record: 0x7F00_0000_0008_000C

Example of Lauterbach programing script:

```
data.set 0x00400218 %QUAD 0x7F0000000008000C ;Bypass Off-line BIST
```

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)

Registers dedicated to offline self-test:

- STCU2 Offline LBIST Status Register (STCU2_LBS)
- STCU2 Offline LBIST End Flag Register (STCU2_LBE)
- STCU2 Offline MBIST Status Low Register (STCU2_MBSL)
- STCU2 Offline MBIST End Flag Low Register (STCU2_MBEL)

Registers dedicated to online self-test:

- STCU2 Online LBIST Status Register (STCU2_LBSSW)
- STCU2 Online LBIST End Flag Register (STCU2_LBESW)
- STCU2 Online MBIST Status Low Register (STCU2_MBSLSW)
- STCU2 Online MBIST End Flag Low Register (STCU2_MBELSW)

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 ID: AN5131)
- MPC574 4P Reference Manual (Document ID: MPC5744PRM)

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

AN11993
Rev. 0
June 2017