Basic PBIST Configuration and Influence on Current Consumption in a TMS570LS Series Device

Application Report

Literature Number: SPNA128
February 2011
The purpose of this document is to provide help for configuring the PBIST module of TMS570LS2x series microcontrollers. The application note explains the configurations, the different algorithm test durations and the influence on current consumption.

1.1 Description

TMS570LS2x series microcontrollers are implemented with PBIST (Programmable Built-In Self Test) architecture. The PBIST architecture provides a memory BIST engine for varying levels of coverage across many embedded memory instances.

The on-chip memories of TMS570LS2x series microcontrollers are classified into fifteen different ROM/RAM groups for the PBIST test, including two ROM groups (STC/PBIST ROM) and thirteen RAM groups. The clock associated to ROM groups PBIST testing is the STC/PBIST ROM clock while the other RAM groups use HCLK as the PBIST testing clock.

The PBIST engine in TMS570LS2x series microcontrollers provides a list of algorithms for memory test. The different algorithms supported in application mode for a certain ROM/RAM group are listed in the TMS570LS20216 Technical Reference Manual (SPNU489), Chapter 6.6 RAM Grouping and Algorithm.

1.2 When Should PBIST Be Run?

The PBIST test is designed to test the integrity of the embedded RAMs and ROMs and it is destructive to the data stored in the target RAM. In the TMS570LS20216 Safety Manual [1], the PBIST test can be treated as:

- Proof testing: Hardware PBIST engine executes multiple algorithms targeted to SRAM physical topology on startup. This method is recognized to provide 99% DC (Diagnostic Coverage) based on TI proven-in-use manufacturing data.
- Online/diagnostic testing (optional): Periodic PBIST can be executed in slices but is destructive to contents. After test completes, the application code has to re-initialize the target RAM. A reset might be required.
2.1 Configuration Steps

This section describes the PBIST configuration, step by step.

1. Program the HCLK to STC/PBIST ROM clock ratio.
   The PBIST engine can test both ROMs (group 1 - 2) and RAMs (group 3 - 15). The ROM_DIV bits in MSTGCR register programs the HCLK to STC/PBIST ROM clock ratio. Therefore, it impacts the test duration and test current consumed during the ROM test. The test duration is proportional to the HCLK to ROM clock ratio. A higher HCLK to ROM clock ratio also means less current consumption during ROM PBIST test.
   Because all the other memories (RAMs) are running on HCLK, this clock ratio has no influence on the RAM PBIST test (group 3 - 15).
   The example below uses following code to set ROM clock the same as the HCLK frequency:
   ```
   e_SYSTEM_ST.MSTGCR_UN.MSTGCR_ST.ROM_DIV_B2 = 0; //ROM clock = HCLK
   ```

2. Enable the PBIST controller in the system module.
   The example below uses following code to enable PBIST Controller:
   ```
   e_SYSTEM_ST.MSINENA_UL=0x1;
   ```

3. Enable the PBIST self test in system module.
   The example below uses following code to enable PBIST self test:
   ```
   e_SYSTEM_ST.MSTGCR_UN.MSTGCR_ST.MSTGENA_B4=0x0A;
   ```

4. Wait until PBIST is out of reset.
   Depending on the ROM_DIV ratio, the PBIST will reset for 16~96 VBUS cycles.
   ```
   asm("nop");
   ```

5. Enable the PBIST internal clocks and ROM interface clock.
   The example below uses following code to enable the PBIST internal clocks and ROM interface clock.
   ```
   e_PBIST_ST.PACT_UN.PACT_UL=3;
   ```
6. Select the RAM/ROM under test, select the algorithm.
   If the user wants to run the algorithms on all the RAM/ROMs (that support these algorithms), please
   choose the RAM override mode; otherwise, please disable the RAM override.
   - RAM override mode enabled.
     In this case, the application code shall select the algorithms. The PBIST engine will run on all the
     valid ROM/RAMs for these algorithms.
     Example:
     ```
     e_PBIST_ST.ALGO_UN.ALGO_UL=0xFFFF; // select all the algorithms
     e_PBIST_ST.PBISTOVERRRIDE_UN.PBISTOVERRRIDE_ST.OVER=1; // default setting
     ```
   - RAM override mode disabled
     In this case, the application code shall select the algorithm run on the target ROM/RAM through
     ALGO register, and select the ROM/ROM under test through RININFO registers and set the OVER
     register to be 0. Please note that the algorithm selected must be valid for all the ROM/RAMs
     selected by RININFO. Otherwise, a PBIST test error will be generated.
     The example below uses following code to select the March13 algorithm on the 160 KByte
     eSRAM.
     Example:
     ```
     e_PBIST_ST.ALGO_UN.ALGO_UL=0x4; // select March13 on Single port RAM
     e_PBIST_ST.RINFOL_UN.RINFOL_UL=0x020; // select the main ESRAM 160 kByte.
     e_PBIST_ST.RINFOL_UN.RINFOL_UL=0; // it is always 0 for TMS570LS2x series.
     e_PBIST_ST.PBISTOVERRRIDE_UN.PBISTOVERRRIDE_ST.OVER=0;
     ```

7. Select both Algorithm and ROM/RAM information from on chip PBIST ROM
   The attached example uses following code to select algorithm and ROM/RAM information from on chip
   PBIST ROM.
   ```
   e_PBIST_ST.ROM_UN.ROM_ST.ROM=0x03;
   ```

8. Configure PBIST to run in ROM Mode and kickoff PBIST test
   PBIST test will start after writing 0x14 to the DLA register.
   Example:
   ```
   e_PBIST_ST.DLR_UN.DLR_UL=0x14;
   ```

9. Wait for PBIST test to complete by polling MSTDONE bit in system module.
   ```
   while(e_SYSTEM_ST.MSTCGSTAT_UN.MSTCGSTAT_ST.MSTDONE_B1 != 0x1);
   ```

10. Once self test is complete, check the fail status registers FSRF0 and FSRF1.
    In case there is a failure (FSRF0 or FSRF1=0x01)
    (a) Read RAMT register which indicates the RGS and RDS values of the failure ROM/RAM.
        RGS (register group select) and RDS (register data select) stand for an unique ROM/RAM select
        id. Refer to the device Datasheet to find out the failing ROM/RAM.
        Example:
        ```
        RGS = e_PBIST_ST.RAM_UN.RAM_ST.RAMGROUPSELECT;
        RDS = e_PBIST_ST.RAM_UN.RAM_ST.RETURNDATASELECT;
        ```
    (b) Read FSRC0 and FSRC1 registers which contains the failure count.
        Example:
        ```
        FSCP0=e_PBIST_ST.FSRC0_UN.FSRC0_UL;
        FSCP1=e_PBIST_ST.FSRC1_UN.FSRC1_UL;
        ```
    (c) Read FSRA0 and FSRA1 registers which contains the address of first failure.
        Example:
        ```
        FSADDRP0 = e_PBIST_ST.FSRA0_UN.FSRA0_UL;
        FSADDR1 = e_PBIST_ST.FSRA1_UN.FSRA1_UL;
        ```
    (d) Read FSRDL0 and FSRDL1 registers which contains the failure data.
        Example:
        ```
        FSDATAP0 = e_PBIST_ST.FSRDL0_UN.FSRDL0_UL;
        FSDATAP1 = e_PBIST_ST.FSRDL1_UN.FSRDL1_UL;
        ```
    (e) Resume the test if required using program control register STR = 2.
        Example:
        ```
        e_SYSTEM_ST.MSTCGSTAT_UN.MSTCGSTAT_ST.MSTDONE_B1 = 0x1;
        ```
e_PBIST_ST.STR.UN.STR_UL=0x02;//restart the test

In case there is no failure (FSRF0 and FSRF1=0x00) the memory self test is complete.

(a) Disable the PBIST internal and ROM clocks.
Example:
   e_PBIST_ST.PACT.UN.PACT_UL=0x00; /*Disable PBIST internal ROM Clocks*

(b) Disable the PBIST self test.
Example:
   e_SYSTEM_ST.MSTGCR_UN.MSTGCR_ST.MSTGENA_B4=0x05;/*Disable PBIST selftest*/
PBIST Test Duration

2.2 PBIST Test Duration

The device Datasheet lists the test duration in clock cycles for any combination of RAM/ROM and algorithm. For STC/PBIST ROM, the unit is ROM clock period. For the rest of RAM, the unit is HCLK clock period. Please note that, the test time in this table does NOT include the time downloading PBIST code from PBIST ROM. Therefore, the actual test time is slightly higher than what is specified here. An empty cell means such an algorithm is not available for such a RAM/ROM. For example, the PBIST and STC ROM can only be tested through ‘triple slow read’ and ‘triple fast read’ algorithm.

In the attached example, the PBIST will start with GIOA[5] high and end with GIOA[5] low. At the end of the test, GIOA[4] will tie to 3.3V if the algorithm runs successfully. Otherwise, GIOA[4] will tie to ground. The test time can be measured as the GIOA[5] high duration. Figure 2-1 shows the waveform captured by the oscilloscope when the March13 algorithm is running on the main 160 Kbyte eSRAM (select ‘3’ in the Hyper Terminal when running the attached example). The blue curve is the current measured on VCC supply and the pink curve indicates when the PBIST test starts and stops. The current measured here is the total device current consumption and not just the PBIST portion. The March13 algorithm on eSRAM takes 266320 HCLK cycles. HCLK is 100MHz in this case. Therefore, the calculated test duration is 2.66ms, which matches the measurement in Figure 2-1 very well.

![Figure 2-1. PBIST Current Waveform Capture (March13 Algorithms on eSRAM)](image)

Note: All the waveforms do not show the absolute worst case current consumption and are just an example.
The March13 algorithm is the most recommended algorithm for the memory self test. The following waveform illustrates when running the March13 algorithm on all the available RAMs (select ‘2’ in the Hyper Terminal when running the attached example). Based on Table 2-7 in the Datasheet, the March13 algorithm on all RAMs takes \((12600 + 12600 + 6360 + 266320 + 50160 + 8400 + 18960 + 25440 + 6480 + 37800 + 175040 = 624360)\) HCLK cycles. Considering the 10ns HCLK cycles, the test duration is about 6.24ms. The measured test duration in Figure 2-2 is slightly longer than this number due to PBIST test microcode loading time.

The current in Figure 2-2 bursts for 5 times, which is the test pattern number of the March13 algorithm. These burst current peaks are generated by the 160 Kbyte eSRAM test. This also applies to the other algorithms except that the other algorithms (see Figure 2-3) only burst twice because they only have two test patterns. Each burst (period) is composed by two parts, beginning with a higher current peak and ending with a lower current peak.

Note: All the waveforms do not show the absolute worst case current consumption and are just an example.

**Figure 2-2. PBIST Current Waveform Capture (March13 Algorithms on All RAMs)**
### 2.3 PBIST Test Current Consumption

Usually, running PBIST consumes more current in VCC domain than running in normal mode at the same HCLK frequency. The PBIST engine runs the PBIST in parallel mode, for example, it runs PBIST on ten eSRAM banks in parallel. By doing this, PBIST test time is reduced but the current is increased.

Across different ROM/RAM groups, the eSRAM consumes the highest current because it has the highest number of banks running in parallel – 10 for ROM/RAM group 6 and 20 for ROM/RAM group 15 in TMS570LS2x devices. Please note that, group 6 and group 15 are duplicated. Running either of them will test the entire 160 Kbyte eSRAM. Group 15 takes only half the time of group 6 but consumes 25% more current. Group 15 is only available in RAM Override Mode Disabled.

Figure 2-3 shows the waveform captured by the oscilloscope when all algorithms are running on all available RAM/ROMs (select ‘1’ in the Hyper Terminal when running the attached example). The current amplitude and duration of different algorithm are represented by different colors. Across different algorithms, Down1A algorithm consumes the highest current.

In each algorithm, the current peaks are generated by the 160 Kbyte eSRAM test. Table 2-1 illustrates the duration of each peak and general height in Figure 2-3. Please refer to Figure 2-2 for the definition of duration and general height.

#### Note: All the waveforms do not show the absolute worst case current consumption and are just an example.

**Figure 2-3. PBIST Current Waveform Capture (All Algorithms on All ROM/RAMs)**
Table 2-1. Current Peak Height and Duration\(^{(1)}\)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NO of Bursts</th>
<th>Peak Current Duration</th>
<th>Peak Current General Height</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Higher Peak</td>
<td>Lower Peak</td>
</tr>
<tr>
<td>March 13</td>
<td>5</td>
<td>2.66 ms</td>
<td>2.66 ms</td>
</tr>
<tr>
<td>Down1A</td>
<td>2</td>
<td>130 us</td>
<td>130 us</td>
</tr>
<tr>
<td>PreCharge</td>
<td>2</td>
<td>103 us</td>
<td>103 us</td>
</tr>
<tr>
<td>Map column</td>
<td>2</td>
<td>83 us</td>
<td>83 us</td>
</tr>
<tr>
<td>DTXN 2A</td>
<td>2</td>
<td>450 us</td>
<td>450 us</td>
</tr>
<tr>
<td>PMOS open</td>
<td>2</td>
<td>1.03 ms</td>
<td>1.03 ms</td>
</tr>
</tbody>
</table>

\(^{(1)}\) All the waveforms do not show the absolute worst case current consumption and are just an example.

Running at a lower HCLK frequency is a feasible method to reduce the PBIST current. One caveat is the longer test time, which is inverse proportional to the HCLK/ROM clock frequency. Another caveat is that the memory is not tested at speed. Figure 2-4 shows the VCC PBIST peak current versus HCLK frequency measured on one unit.

![Figure 2-4. PBIST Peak Current versus HCLK Frequency (March13 Algorithms on eSRAM)](image_url)
2.4 Example

The attached example communicates with PC through SCI using RS232 protocols. User can type the command in Hyper Terminal to use PBIST to test all the RAM/ROM with all the available algorithms or test the main 160 Kbyte SRAM with the March13 algorithm.

It includes following source files:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intvecs.asm</td>
<td>Interrupt vectors setup file.</td>
</tr>
<tr>
<td>Boot.asm</td>
<td>Initialize stack, call _init and _main.</td>
</tr>
<tr>
<td>Cinit.asm</td>
<td>Initialize global variables.</td>
</tr>
<tr>
<td>Swi_util.asm</td>
<td>Swi interrupt service routine.</td>
</tr>
<tr>
<td>Startup.c</td>
<td>Initialize the system.</td>
</tr>
<tr>
<td>System_init</td>
<td>Initialize the system, enable peripherals.</td>
</tr>
<tr>
<td>SCI1_RS232.c</td>
<td>RS232 functions.</td>
</tr>
<tr>
<td>TMS570_PBIST_Test.c</td>
<td>Demonstrate the PBIST.</td>
</tr>
<tr>
<td>TMS570_PBIST_Test.pjt</td>
<td>Project file.</td>
</tr>
</tbody>
</table>

To run the demo:

- **Hardware**
  - Connect the TI EVM board USB port to a PC; open a HyperTerminal; configure the right COM port and the baud rate to be 19.2 kHz, no parity, one stop bit.
- **Software**
  - Program TMS570_PBIST.out to the device flash using nowFlash tool; reset the device.

Then, the following menu prompts out from the HyperTerminal:

- Please input a character to start PBIST testing:-
- Run all algorithms on all RAM/ROMs ------------ 1
- Run March13 on all Single/Dual Port RAM------- 2
- Run March13 on 160K ESRAM only --------------- 3
  
After that, the user should input the function index in the HyperTerminal to run the function. The PBIST will start with GIOA[5] high and end with GIOA[5] low. At the end of the test, GIOA[4] will tie to 3.3V if no error is detected. Otherwise, GIOA[4] will tie to ground. Meanwhile, the HyperTerminal will also output information about the PBIST test status.
References

3.1 Reference Documents

For more information refer to the following documents:

1. TMS570LS20216 Safety Manual (NDA)
2. TMS570LS20216 Technical Reference Manual (SPNU489)
3. TMS570LS20216 Data Sheet (SPNS141)
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and/or changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI’s standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal and regulatory requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or “enhanced plastic.” Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions: