DDR Tuning and Calibration Guide on NXP i.MX 6 Quad

ScanWorks® Processor-based Functional Test for DDR

Tune-to-Test

by Larry Osborn

ASSET

Larry Osborn

Larry Osborn, Senior Product Manager, at ASSET InterTech, has over 30 years of experience in product management, hardware/software product design and development, product delivery to the marketplace and user support. Over the years, Larry has established a proven track record for identifying user needs and opportunities in the marketplace, providing innovative solutions and exceeding the expectations of users. At ASSET, Larry is responsible for the profit and loss for a product group. Prior to ASSET, he has held positions with Lockheed Martin, OCD Systems, Wind River, Hewlett-Packard, Ford Aerospace and Intel® Corporation. He holds a Bachelor's Degree in Computer Science from the University of Kansas and various technical and marketing training certifications.

Table of Contents

Table of Figures

DDR Tuning and Calibration with ScanWorks Processor-based Functional Test for DDR

DDR Calibration Background

Designing with DDR memories can be challenging, and the move from DDR3 to DDR4 and beyond, doesn't lessen the challenge. By now we all know that configuration of DDR memories in a board design involves dozens to hundreds of register-writes to the memory controller. The DDR controller can be complex and require hours to days of effort reading the technical reference manual (TRM). The memory controller registers set the timing parameters and operating mode of the DDR interface. There are two distinct types of values used in the configuration of the controller. These values can be thought of as those that are static, not subject to board, component or environmental conditions; and dynamic, specifically used to account for timing variations caused by board layout or environmental conditions. Programming of the memory controller is influenced by several aspects of the board design. If any of these items are changed, then the memory controller settings must also change.

1. Processor/memory controller used
2. Number and organization of physical memory chips used
3. Speed grade of the memory chips
4. Operating speed of the DDR interface
5. Board layout, including trace length and impedance considerations.

Static DDR Configuration Parameters

Many of the memory controller configuration parameters are static. These parameters are fixed based on the processor, memory chips, and operating speed of the DDR interface. Changes to the board layout and termination on the DDR interface don't change the static configuration parameters. These static values are found in the memory vendor's data sheet and are the timing values as defined by the JEDEC standard. All that is required of the user is to enter the data for the appropriate register. This is a straightforward process, however very tedious, given dealing with the register name conversion to address and working in hex.

DDR Tuning and Calibration

In addition to the static DDR parameters, some memory controllers provide settings to calibrate the operation of the memory bus to adjust for board layout variations. Each board design should determine the optimal calibration settings to improve signal integrity and reliability of the memory bus.

Common DDR Calibration parameters include:

- Write leveling
- DQS gating
- Read data eye
- Write data eye

The JEDEC standard requires that DDR3 and DDR4 memories support write leveling, but other calibration types are optional.

DDR calibration should be run against a sample of boards during the board bring up stage while the boards are running at ambient temperature. Processor vendors recommend sampling 5-10 boards and checking that the calibration results across the boards are consistent. Once consistent DDR calibration results are determined, the optimal DDR setting should be applied, and the boards tested under the required environmental conditions.

Write Leveling

The goal of write leveling is to adjust the timing of the write DQS signals relative to the DDR clock. The DDR PHY in the memory controller adds a programmable delay to the DQS signal in order to meet the timing requirement of the memory part.

The JEDEC specifications for DDR3 and DDR4 memories recommend using fly-by topology to improve signal integrity. With traditional topologies, all memory bus signals are routed to each memory chip point-to-point. With fly-by topology, the address, command, and clock signals are routed to each memory chip in series, while the DQS and data signals are routed point-to-point.

When the fly-by topology is used, the write leveling must be calibrated for each byte lane individually to ensure correct operation.

DQS Gating

The goal of DQS gating calibration is to determine when the read DQS signal is valid. Memory operations includes periods where the DQS signal is invalid and must not be sampled to prevent incorrect behavior. The invalid states include when the read DQS signal is tri-stated or it is actively driven by the memory controller PHY. This is an optional calibration step, depending on the processor and memory controller.

Read Data Eye

The goal of read data eye calibration is to align the read DQS signal to the center of the valid read data. This is an optional calibration step, depending on the processor and memory controller.

Write Data Eye

The goal of write data eye calibration is to align the write DQS signal to the center of the valid write data. This is an optional calibration step, depending on the processor and memory controller.

Calibration/tuning is not just writing a value in a register to adjust the DQS delay or other timing parameter, it involves repetition where a timing parameter is changed, testing memory access to see that it passes and repeating the process until the memory access fails. The failure will indicate that the timing settings are either too slow or too fast for the memory to respond. This would suggest an algorithmic approach is required and the level of confidence of the optimal settings is determined by the robustness of the memory test.

Programming of the memory controller is influenced by several aspects of the board design. If any of these items are changed, then the memory controller settings must also change.

1. Processor/memory controller used
2. Number and organization of physical memory chips used
3. Speed grade of the memory chips

4. Operating speed of the DDR interface
5. Board layout, including trace length and impedance considerations
6. Memory Vendor change

To address the tedious nature of entering the register values, poring over the technical reference manuals for both the SoC and the memory devices, and then finding the optimal values, ASSET has a solution to shorten the design engineering effort and provide robust memory testing to ensure the manufacturing process efficiency. This paper provides the details about that solution.

Processor-based Functional Test for DDR Software Based Tuning

ScanWorks Processor-based Functional Test for DDR (PFTDDR) provides commands to tune and calibrate DDR memory. These commands automatically adjust one or more calibration parameters and test memory at each setting, generating a scoreboard of settings that pass and fail.

The specific tuning process varies based on the processor type and the features of the memory controller. We will cover the specific flow on the i.MX6 later. However, the general tuning process follows this procedure.

1. Set the write leveling delay for one-byte lane to a fixed value
2. Optional - calibrate other DDR3/DDR4 parameters (DQS gating, read data eye, write data eye, etc.)
3. Run a memory cell test, using burst transactions
4. Run a memory bus noise test, using burst transactions
5. Log the results of the memory tests
6. Increment the write leveling delay value and repeat at step 2

The procedure is repeated for all valid byte lanes for the design. For example, if your board uses a 64-bit wide data bus to the DDR memory, the procedure is repeated a total of 8 times.

Unlike hardware-based calibration, the software calibration procedure does not use DQ prime bits to obtain the leveling feedback. Instead, the results of the memory tests determine whether a specific write leveling setting is valid. At the successful completion of the procedure, PFTDDR

reports the window of valid write leveling delay values for all byte lanes and sets the calibrated result to the middle of the window.

Write Leveling Examples

At the completion of DDR tuning, the PFx Developer generates a summary of the tuning results. The following sections illustrate specific examples of write leveling calibration performed by PFx Developer. The output format will vary based upon the architecture.

Point-to-point Topology with DDR4 Memory

The first example is the write leveling calibration result for a reference board using DDR4 memory chips using point-to-point routing for all signals. In the table below (Figure 1) each write leveling delay value represents 1/8 of a clock period. Byte Lane 0 passed with delay values from 4/8 clocks to 11/8 clocks, while Byte Lane 3 passed with delay values from 1/8 clocks to 8/8 clocks.

```
          0     4     8     12    16    20    24    28
ByteLane ---------------------------------------------
0x00    | 0000 1111 1111   0000  0000  0000  0000  0000
0x01    | 0000 1111 1111   0000  0000  0000  0000  0000
0x02    | 0001 1111 1110   0000  0000  0000  0000  0000
0x03    | 0111 1111 1000   0000  0000  0000  0000  0000
0x04    | 0000  0000  0000 0000  0000  0000  0000  0000
0x05    | 0000  0000  0000 0000  0000  0000  0000  0000
0x06    | 0000  0000  0000 0000  0000  0000  0000  0000
0x07    | 0000  0000  0000 0000  0000  0000  0000  0000
0x08    | 0000  0000  0000 0000  0000  0000  0000  0000
```

Figure 1: Write Leveling Pass Matrix

After completion of tuning, PFx Developer recommended the following settings, which represent the middle of the operating window for each byte lane as shown in Figure 2.

Byte Lane	Write Level Delay
0	7/8 clocks
1	7/8 clocks
2	6/8 clocks
3	4/8 clocks

Figure 2: Write Level Delay DDR

Note that even though the board layout used the point-to-point topology, empirical testing showed that the optimal write level delay varied by 3/8 of a clock between byte lane 0 and byte lane 3.

Fly-by Topology with DDR3 Memory

The second example, shown in Figure 3 is the write leveling calibration result for a reference board using DDR3 memory chips on a DIMM module. The DIMM module has a total of 9 memory chips (8 for data, 1 for ECC) and the signals are routed using the fly-by topology. As in the other example, each delay value represents 1/8 of a clock period.

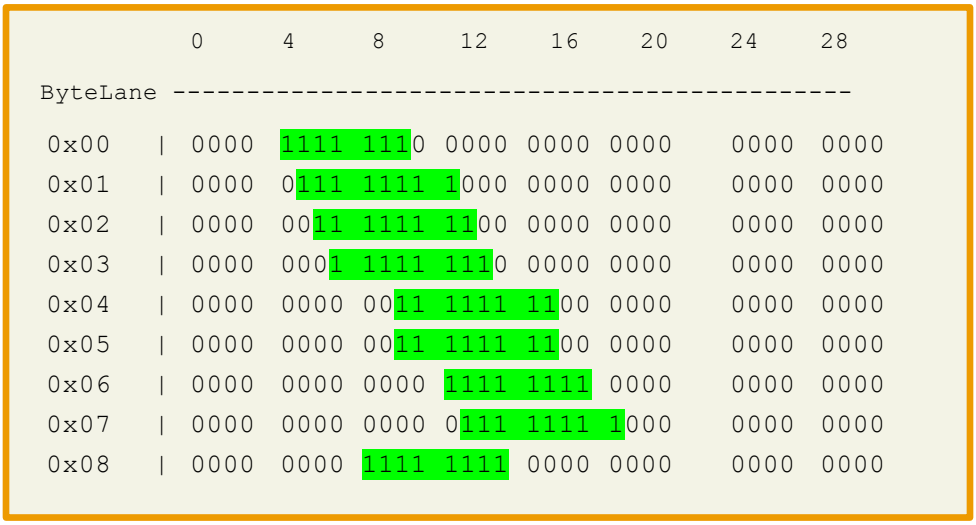


Figure 3: Write Leveling Example 2

After completion of tuning, PFx Developer recommended the following settings (Figure 4), which represent the middle of the operating window for each byte lane.

Byte Lane	Write Level Delay
0	7/8 clocks
1	8/8 clocks
2	9/8 clocks
3	10/8 clocks
4	138 clocks
5	13/8 clocks
6	15/8 clocks
7	16/8 clocks
8 (ECC byte)	11/8 clocks

Figure 4: Write level Delay DDR3

As expected when using the fly-by topology, there is a lot of variation in the optimal write leveling setting for each byte lane. Of particular note is that byte lane 0 and byte lane 7 do not share any common write leveling delay value that passed all memory tests, which makes the tuning process an absolute requirement.

Hardware-Assisted Tuning

The JEDEC specification for DDR3 and DDR4 memory details a hardware-assisted process for calibrating the write leveling setting. This process involves putting the memory chips into write leveling mode and then having the memory controller sample a specific data bit (called the prime data bit) while the write leveling delay value is changed.

There are some drawbacks to relying on the hardware-assisted write leveling:

- Not all memory controllers support the automatic sampling of the prime data bit or have reported errata related to this capability.
- The prime data bit must be routed to a specific pin on the processor for correct operation. Specifically, DIMM modules can swap data signals to improve signal integrity causing the prime data bit to get mapped to the wrong pin on the processor.

Theory of Operation

Overview

The primary purpose of ScanWorks® Processor-based Functional Test for DDR (PFTDDR) is twofold. At the highest level the concept is tune-to-test. Tuning is to be accomplished by the design engineering team and test to be conducted by the manufacturing test team. However, in order to have a valid tuning process, testing must accompany the tuning at the design phase to validate the tuning values chosen. At the basic level, PFTDDR is used to enter DDR controller settings, automatically tune DDR settings, interactively validate those settings, and then provide a means to export these settings and desired test sequenced for automated testing. PFTDDR software is comprised of two major components: PFx Developer host software, and PFTDDR target agent. These components communicate with each other through a communication channel via a JTAG connection as shown in Figure 5.

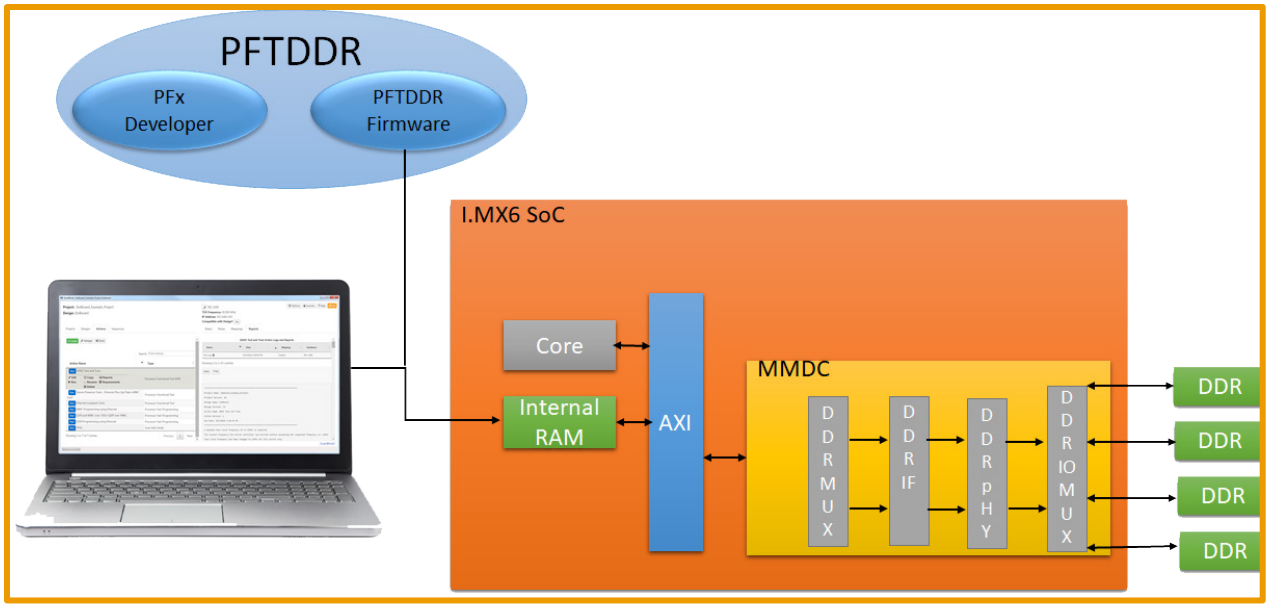


Figure 5: PFx System Overview

The PFTDDR target agent runs from the on-chip memory of your embedded system, allowing you to fully test your external DDR memory using the embedded processor. The agent provides a rich set of capabilities for configuring, testing, and optimizing your DDR memory and PHY.

The PFx Developer host software runs on your Windows host PC and provides a user interface to the PFTDDR firmware. The PFx Developer provides two main functions: a tool for optimizing and validating DDR memory; and a tool for exporting DDR test sequences for use within the ScanWorks platform.

Understanding the i.MX6 technical landscape

Before we get too focused on applying PFTDDR to the SABRE Lite board, some background on the infrastructure of the i.MX6 and the communications path between the processing subsystem and the external DDR3 memories is warranted. In Figure 6, we see a very sophisticated and capable SoC. With sophistication, information is generally necessary, and the TRM for the i.MX6 provides 5,800 pages of information in 71 chapters to cover this chip. However, we are only interested in the communications between the A9 core and the DDR3 to conduct the tuning or calibration of the DDR3. The communications involve transmitting information between the A9-Core via the AXI interface to the Multi-mode DDR Controller (MMDC) and out to the DDR3 memories which are not AXI bus knowledgeable. The MMDC module is a DDR

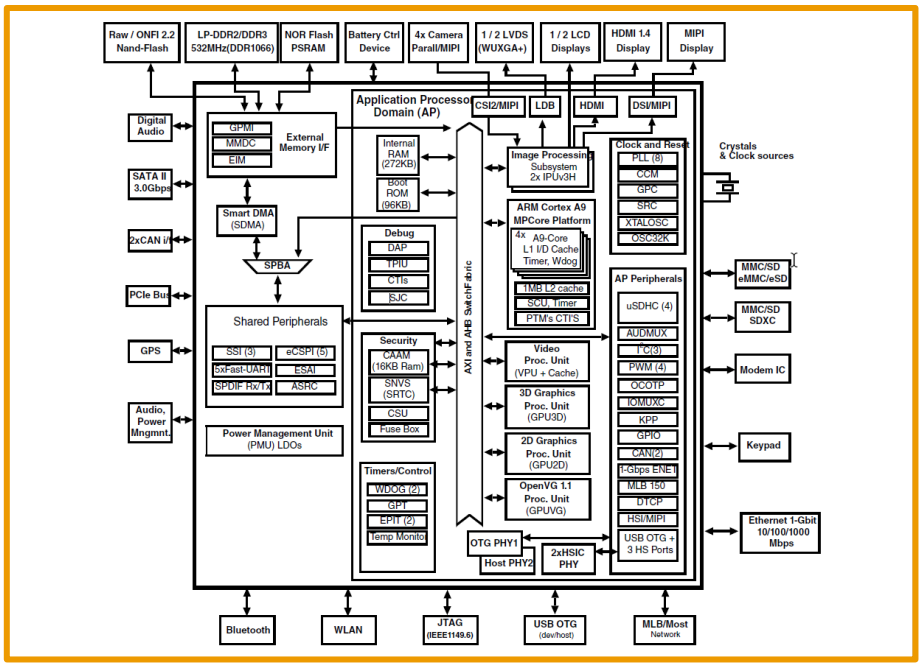


Figure 6: NXP i.MX 6Dual/ i.mx 6Quad

controller that can support several types of DDR memories and two channel x32 and x64 memory widths and is responsible for managing the AXI to DDR transactions. The MMDC simplified block diagram in Figure 7 provides a clear picture on the dual channel nature of the control but does hide the complexity of the device. The embedded device is discussed in chapter 44 of the TRM for 185 pages. Of that chapter, the calibration (both hardware and software) calibration covers 22 pages.

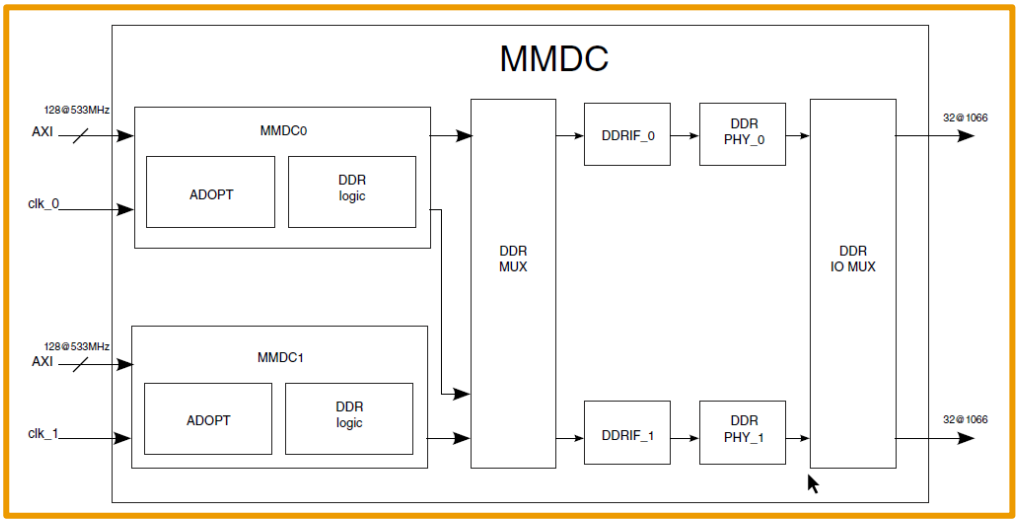


Figure 7: MMDC Block Diagram

The calibration covers eight different possible calibrations tasks. They are:

1. ZQ calibration for external DDR device
2. ZQ calibration for i.MX DDR I/O pads for calibrating the DDR driving strength
3. Read DQS gating calibration for DDR3 only
4. Read data calibration
5. Write data calibration
6. Write leveling calibration
7. Read fine tuning. Adjustment of up to 7 delay-line units for each read data bit
8. Write fine tuning. Adjustment of up to 3 delay-line units for each read data bit

If you look into the TRM about calibration, you will see that there are 20 to 35 steps involved in the calibration actions depending upon what signal you are attempting to calibrate. As you can see from this cursory overview, calibration is a complex process.

SABRE Lite (i.MX6 Quad) Application

DDR3 PHY Calibration

PFTDDR supports two modes for calibrating the DDR3 PHY.

1. Software calibration
2. Automatic hardware calibration (no longer recommended by NXP)

Software Calibration

PFTDDR provides a software calibration procedure for the DDR3 PHY on the i.MX6 processor family. This procedure calibrates the DDR3 PHY by executing the following steps:

1. Set the write leveling delay for one-byte lane to a fixed value.
2. Calibrate of the DQS gating
3. Calibrate the read data DQS delay
4. Calibrate the write data DQS delay
5. Run a memory cell test
6. Run a memory bus noise test
7. Increment the write leveling delay value and repeat at step 2.

The procedure is repeated for all valid byte lanes for the design. For example, if your board uses a 64-bit wide data bus to the DDR memory, the procedure is repeated a total of 8 times.

Unlike hardware-based calibration, the empirical software calibration procedure does not use DQ prime bits to obtain the leveling feedback. Instead, the results of the memory tests determine whether a specific write leveling setting is valid. At the successful completion of the procedure, PFTDDR reports the window of valid write leveling delay values for all byte lanes and sets the calibrated result to the middle of the window.

The board and SoC setup

There are several setup steps required before we begin the DDR PHY training process. The best way to understand this process is via example. The example provided here is conducted on the SABRE Lite. Should you require a more active visual approach, please refer to the video found here.

This setup will cover the major steps used in the tuning of the DDR PHY. This is not intended to be a step for step process. The information provided will provide enough data so that the process can be understood.

The process begins by loading the project file for the SABRE Lite within ScanWorks and then selecting Edit of the DDR RAM Test (Figure 8). For more information about the ScanWorks platform, check us out.

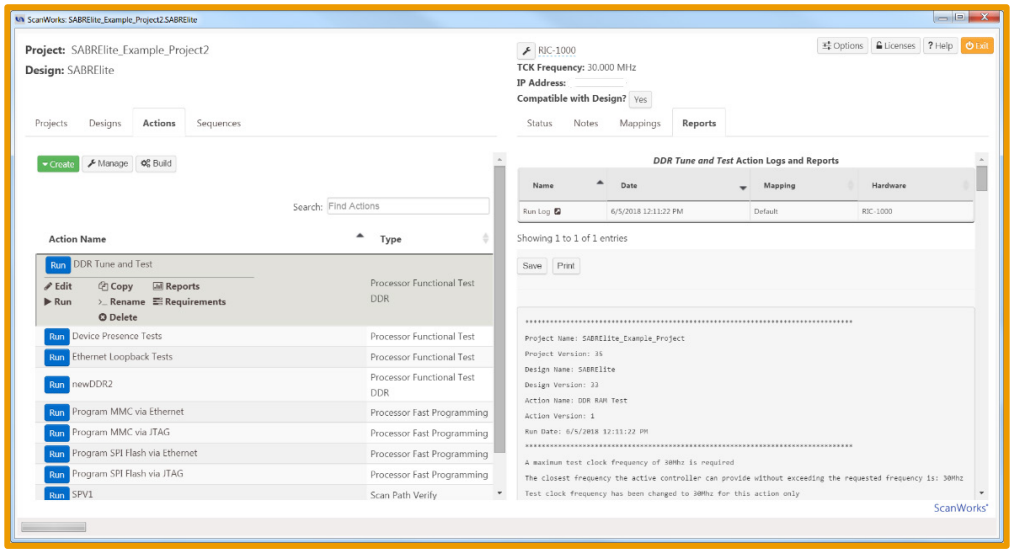


Figure 8: SABRE Lite Example Project

The purpose of editing the action is so the user can see all the aspects of what is provided by the user interface to support tuning and testing of the DDR memories. Editing the action will launch the PFx Developer user interface as shown below in Figure 9.

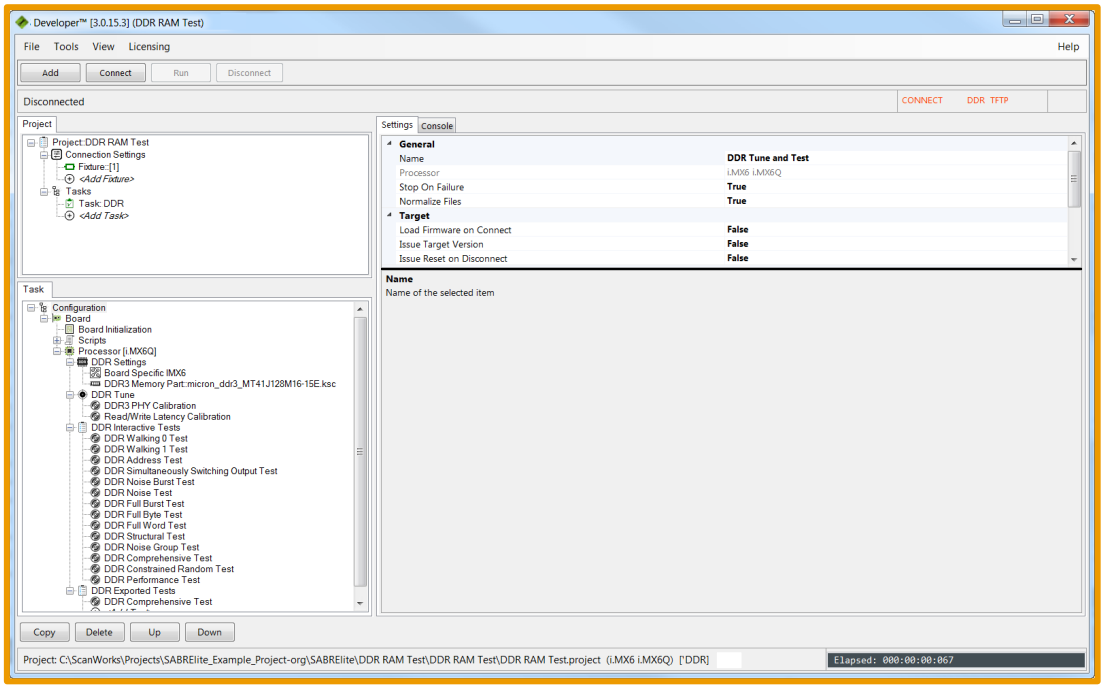


Figure 9: PFx Developer

We'll start by examining the configuration tree.

Understanding the Configuration Tree

The configuration tree (Figure 10) maintains an optional board initialization file, optional script files and commands, DDR settings, and a list of memory tests to be exported. The tree is organized as a single Configuration node which contains a single Board node. Each node can be expanded or collapsed (shown expanded). The Board node contains a Board Initialization node, a Scripts node, and a Processor node. The Processor node contains a DDR Settings node, DDR Tune node, DDR Interactive Test node, and a DDR Exported Tests node. The DDR Settings node contains a Board Specific node and a Memory part node. The DDR Tune node contains the command for DDR PHY Training. The DDR Interactive Tests contain multiple nodes that are used to run a memory test interactively. The Exported Tests node can contain zero or more memory tests to be exported. In our example the Comprehensive Test is exported.

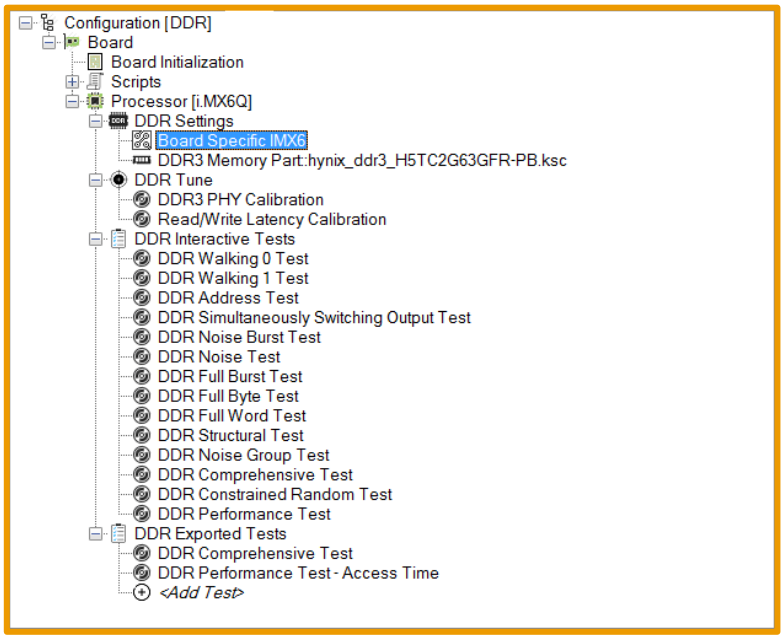


Figure 10: Configuration Tree

Note: The Memory Part node will vary based on the type of DDR memory selected under DDR Settings.

Selecting a node in the configuration tree, with the **Settings Mode** tab selected, displays all the properties available for that node. Use the **Settings Mode** panel to update the value of a given property for the highlighted node (Figure 11).

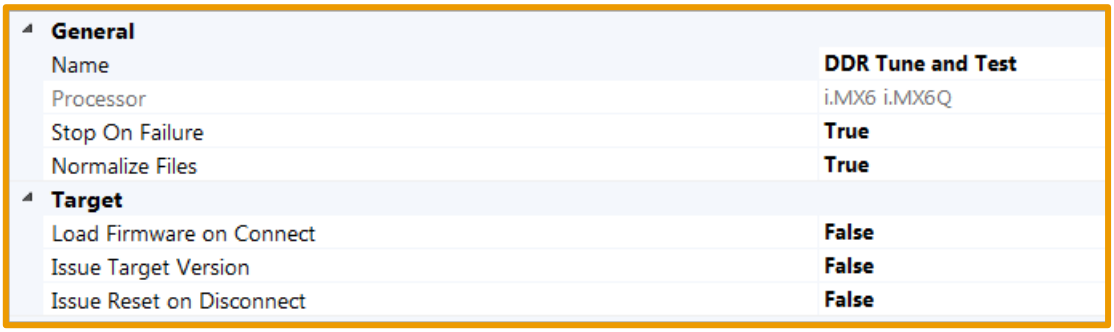


Figure 11: Settings Mode

Board Initialization

The next step is to specify if you want to run a script file that prepares the board for proper operation. This script may disable a watchdog timer, configure GPIO signals, enable clocks, configure power, or other board specific steps. If this step is enabled, the board initialization script will be executed once a new connection is established. There are no board initialization steps needed for the SABRE Lite.

Configuring the DDR Controller

PFTDDR requires several DDR controller parameters that differ by circuit board design. These parameters are stored in the Board Specific and Memory Part nodes.

Choose DDR Configuration Settings

The first step for any new configuration is to choose your DDR configuration setting (Figure 12). PFx Developer allows two choices: Automatic and Manual.

The Automatic setting uses values for memory controller registers that are derived from memory timing specifications. The manual setting uses values for memory controller registers as specified directly from numeric values. If you are starting with timing specifications from a memory data sheet as the basis for configuring the memory controller, select Automatic. If you are starting with a raw register dump, select manual. For this SABRE Lite example, we will select Automatic.

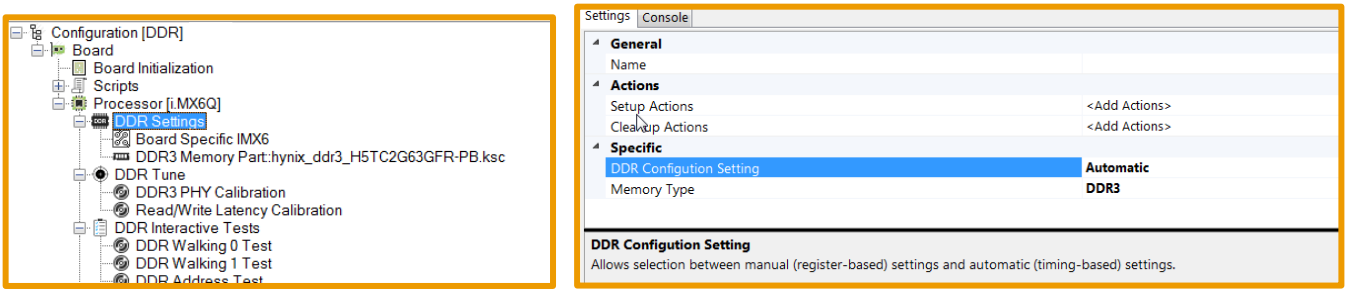


Figure 12: DDR Settings Node and Configuration

Next, click on the pull-down menu opposite of the Memory Type parameter (Figure 13). Use this pull-down to select from the various memory types, such as SDRAM, DDR2, DDR3, DDR4, and LPDDR2. For the SABRE Lite the setting is DDR3.

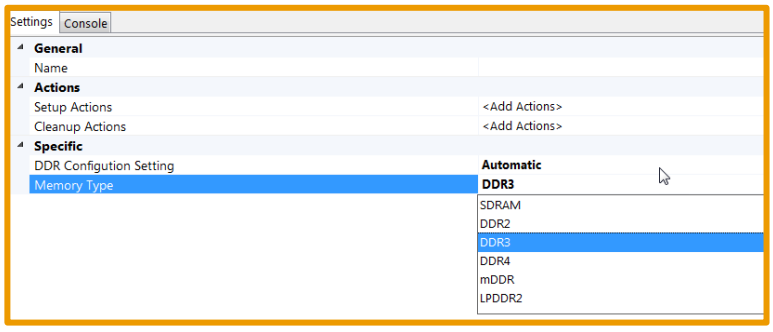


Figure 13: Memory Type Setting

Loading and Running PFTDDR Firmware

Now that we have examined the key parameters of the configuration process, we need to load the target agent onto the SoC. PFx Developer provides a user interface to the firmware running on your embedded device. This firmware is loaded and executing on the SABRE Lite using JTAG. This is accomplished by selecting "**Connect**" button within the PFx Developer UI (Figure 14).

Figure 14: Connect

When the firmware successfully boots, it prints "DUT Ready" in the console window (Figure 15) The PFTDDR agent typically boots and is ready in under one second.

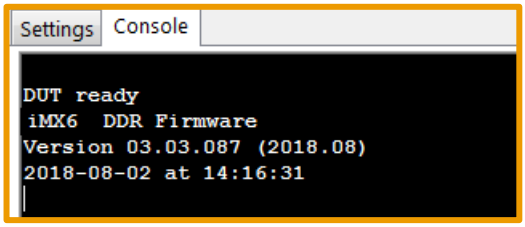


Figure 15: Target Agent Response

Update Board Specific Settings

Click on the board specific node under DDR settings and then click on the settings tab (Figure 16) to view and modify all automatic and manual settings. Depending on your Auto versus Manual setting, certain sections will be collapsed. Under the General section at the top, several read-only parameters are provided for convenience. Click on any parameter and the panel at the bottom is updated with help text.

The parameters that need to be set for the SABRE Lite are:

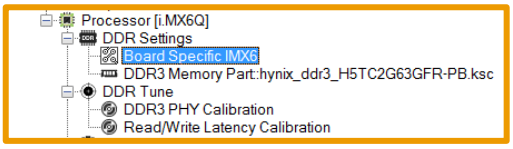
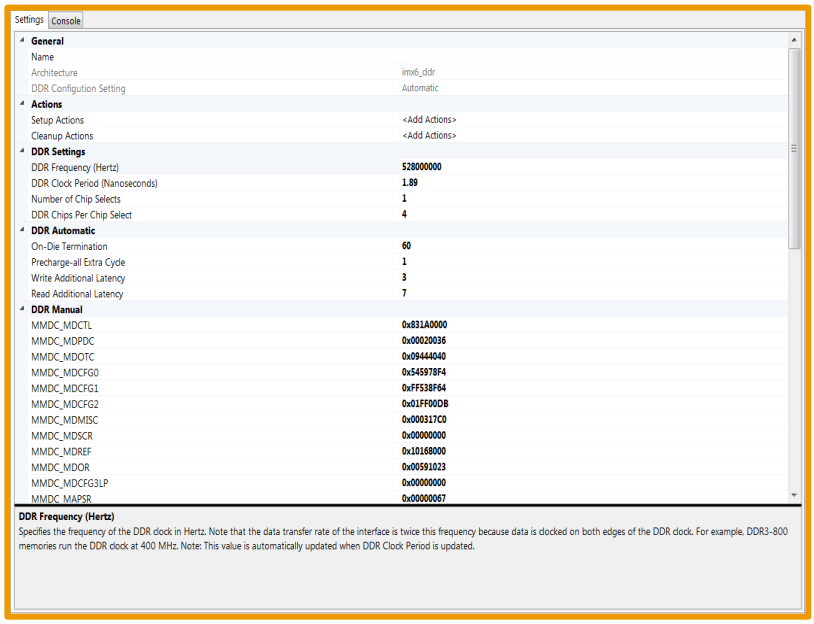


Figure 16: Board Specific



DDR Frequency 528000000
Number of Chip Selects 1
DDR Chips Per 4
On-Die Termination 60
Precharge all Extra Cycle 1
Write Additional Latency 3
Read Additional Latency 7

Then under the **DDR Automatic & Manual** settings for the SABRE Lite set:

Fixed PHY Calibration 1

Note: The data for the DDR Settings and DDR automatic were determined by the schematic, the i.MX6 TRM and user guide for the SABRE Lite

The data for the **DDR PHY** settings will initially be zero. Upon completion of the tuning process the data is imported by the user with a mouse click as shown in Figure 17.

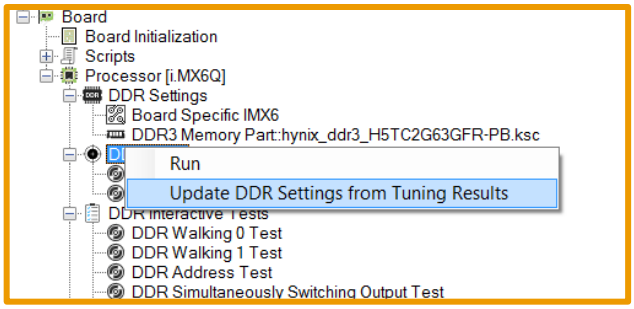


Figure 17: Update DDR Settings

Update Memory Part Settings

Next click on the **Memory Part** node under **DDR Settings** and then click on the **Setting** tab (Figure 18) to view or modify all memory part settings.

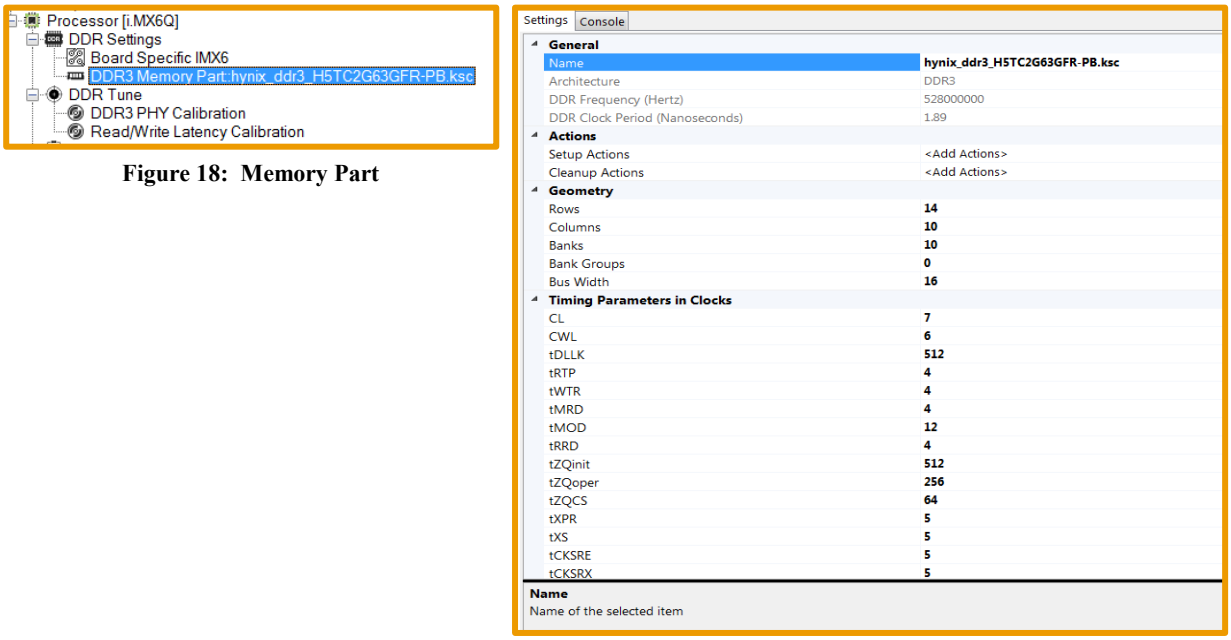


Figure 18: Memory Part

Under the General section at the top, several read-only parameters are provided for convenience. Click on any parameter and the panel at the bottom is updated with help text. By starting with the SABRE Lite example project, we have already imported the device library for the Hynix DDR3 device used in the SABRE Lite design. All the values needed are extracted from the data sheet of the Hynix device and provided to the user within the library folders which contains many memory vendors and parts.

Configure DDR Controller

You must configure the memory controller each time you load and run the PFTDDR firmware. PFTDDR calculates the proper register values for the memory controller based on the data specified in this configuration file, converting timing information into register values as necessary.

To apply the register values to the controller hardware, select "**Configure DDR Controller**" from the **DDR Setting** right-mouse menu option (Figure 19). PFTDDR will write the values to the hardware in the proper sequence to accomplish the specified configuration.

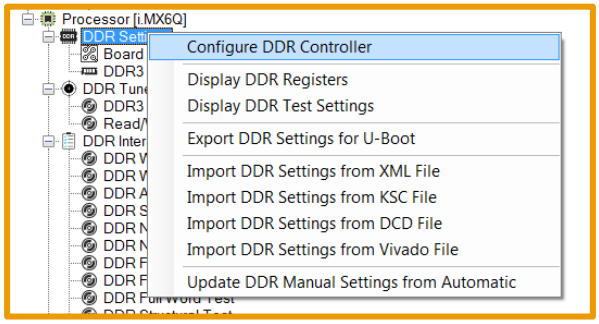


Figure 19: Configure DDR Controller action

Note: You must be connected to a target in order to run the configure operation.

DDR Configuration validation

The final step is to ensure that the DDR controller and memory can operate properly. It is important to remember that we need to have the DDR memory system working before we try to find the optimal settings via the tuning. This step involves running a memory test. We recommend using the Comprehensive Memory test as it includes both structural and stress testing. (Figure 20) With the DDR configuration complete, downloaded to the SoC, and verified operational, we can now begin the turning process.

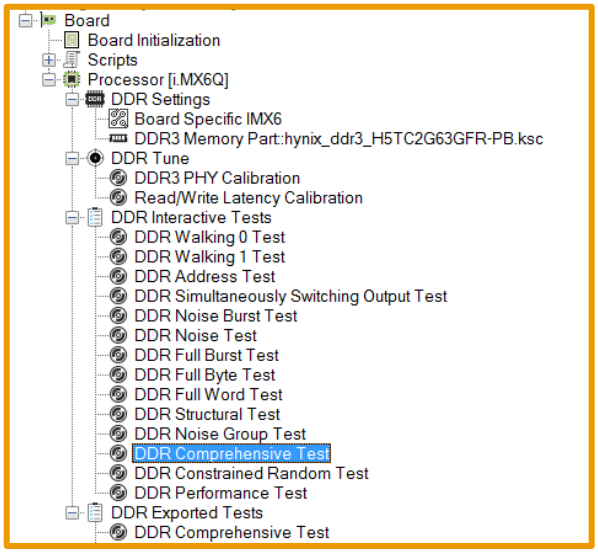


Figure 20: DDR Comprehensive Memory Test Prior to Tuning

Executing the Software Based DDR PHY Training

By selecting **DDR Tune** -> **Run** this will start the software process running in OCM of the SoC to find the optimum values for the DDR PHY (Figure 21).

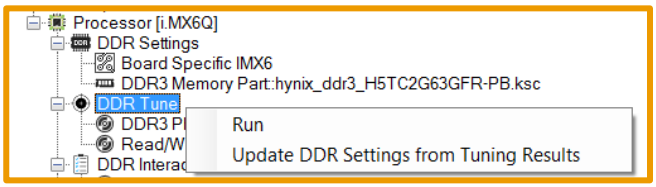


Figure 21: Start Tuning

The software-based DDR PHY training can take significant time to run, as much as 20 minutes or more to complete. The procedure is typically run on new board designs, when changing to a new memory part, or changing the DDR interface space.

Interpreting the Training Results

At the completion of the software DDR PHY training, PFTDDR prints out a summary of the results. The training results from a SABRE Lite board are shown in the following figures. Portions of the output are truncated here to save space. In Figure 22, the tuning begins with starting with a calibration seed value and testing the **upper** write level bounds on byte lane 0 and increase the value at each step.

```
ddr.tune.phy // DDR3 PHY Calibration
Starting i.MX6 DDR PHY Tuning ver 2.0: 8 byte lanes
Write leveling search using seed value: 0x004C
Searching for upper write level bounds on byte lane 0
-------------------------------------------------
MPWLDECTRL = 0x004D004C
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed
SDRAM: Full burst (64 bit) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
SDRAM: Memory bus noise (burst) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
-------------------------------------------------
MPWLDECTRL = 0x004D0054
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed
SDRAM: Full burst (64 bit) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
SDRAM: Memory bus noise (burst) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff

// Content Truncated //
MPWLDECTRL = 0x004D007C
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed
SDRAM: Full burst (64 bit) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
Bad value at address 0x00000000_100444a0
   expected 0x00000000_100444a0, actual 0x00000000_10044400
Bad value at address 0x00000000_100445c0
   expected 0x00000000_100445c0, actual 0x00000000_10044500
Bad value at address 0x00000000_10044640
   expected 0x00000000_10044640, actual 0x00000000_10044600
Bad value at address 0x00000000_10044680
// Content Truncated //
```

Figure 22: Beginning of Calibration

At some point, the value selected will cause failure (Figure 22) and then the direction is switched to find the **lower** bounds as shown in Figure 23. Once the lower bound is found, the calibration will emit the write leveling window with upper, middle and lower for byte lane 0 shown at the end of Figure 23.

```
Bad value at address 0x00000000_100a7bc0
   expected 0x00000000_100a7bc0, actual 0x00000000_100a7b00
Too many memory errors detected (> 10 errors), ending test
Test failed with error code = 0x00080030
Searching for lower write level bounds on byte 0
--------------------------------------------------
MPCWLDECTRL = 0x004D004C
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed
SDRAM: Full burst (64 bit) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
SDRAM: Memory bus noise (burst) [00000000_10000000 -
00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_000000ff
--------------------------------------------------
// Content Truncated //
--------------------------------------------------
MPCWLDECTRL = 0x004D003D
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed

Write level window (1/256 cycles) for byte 0
   lower  = 0x00000000
   upper  = 0x0000007B
   middle = 0x0000003D
```

Figure 23 Lower Bound and Bye Lane 0 data

```
Write leveling search using seed value: 0x004D
Searching for upper write level bounds on byte lane 1
--------------------------------------------------
MPWLDECTRL = 0x004D003D
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed
SDRAM: Full burst (64 bit) [00000000_10000000 - 00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_0000ff00
SDRAM: Memory bus noise (burst) [00000000_10000000 -
00000000_10ffffff]
Warning: only comparing data bus mask bits 0x00000000_0000ff00
--------------------------------------------------
// Content Truncated //
--------------------------------------------------
MPCWLDECTRL = 0x00360044
  Starting calibration of DQS gating, read delay, and write delay
  Calibration of DQS gating, read delay, and write delay completed

Write level window (1/256 cycles) for byte 7
   lower  = 0x00000000
   upper  = 0x0000006C
--------------------------------------------------
// Content Truncated //
```

Figure 24: Calibration continues

Searching for upper write level bounds on byte lane 1 continues the process as shown in Figure 24. The next step is to replicate the calibration and write leveling for the rest of the byte lanes 1 through 7.

Once the final byte lane calibration is complete, PFTDDR will emit a table with the final write leveling results as shown in Figure 25.

```
Final Write Leveling results (1/256 cycles):

          lower   upper   middle  MPWLDECTRL setting
Byte 0 : 0x0000  0x007B  0x003D     0x003D
Byte 1 : 0x0000  0x007B  0x003D     0x003D
Byte 2 : 0x0000  0x0086  0x0043     0x0043
Byte 3 : 0x0000  0x008D  0x0046     0x0046
Byte 4 : 0x0000  0x008B  0x0045     0x0045
Byte 5 : 0x0000  0x0071  0x0038     0x0038
Byte 6 : 0x0000  0x0088  0x0044     0x0044
Byte 7 : 0x0000  0x006C  0x0036     0x0036

Calibration completed.  Update your board file with the following settings.
            1 -> $ddr.memctrl.fixed_calibration

            0x003D003D -> $mmdc_mpwldectrl0
            0x00460043 -> $mmdc_mpwldectrl1
            0x426F0302 -> $mmdc_mpdgctrl0
            0x0269025F -> $mmdc_mpdgctrl1
            0x382D3033 -> $mmdc_mprddlctl
             0x31384541 -> $mmdc_mpwrdlctl
            0x00380045 -> $mmdc2_mpwldectrl0
            0x00360044 -> $mmdc2_mpwldectrl1
            0x02790310 -> $mmdc2_mpdgctrl0
            0x026F0243 -> $mmdc2_mpdgctrl1
            0x35322E3D -> $mmdc2_mprddlctl
             0x47324C39 -> $mmdc2_mpwrdlctl
If using a timing based file, copy and paste the following lines into your board file
            7  -> $ddr.memctrl.ralat
            3  -> $ddr.memctrl.walat
If using a register based file, copy and paste the following lines into your board file

            0x000317C0 -> $mmdc_mdmisc

[PASSED]
```

Figure 25: Final Write leveling results

After the successful completion of the software calibration, update your PFTDDR project file with the fixed PHY calibration settings by selecting the **DDR Tune** node and right-click and select **Update DDR Settings from Tuning Results** as shown in Figure 26 Then follow the prompts to update your project file with the tuning results. Finally, save your PFTDDR project file.

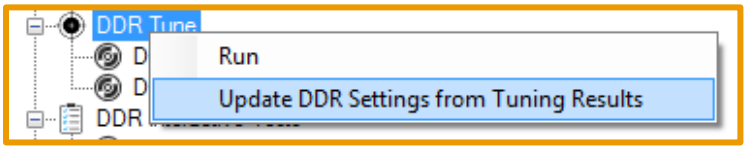


Figure 26: Update DDR Settings

Automatic Hardware Calibration

PFTDDR executes the automatic hardware calibration for all the required PHY parameters. Note that using automatic hardware calibration is no longer recommended by NXP. However, the automatic hardware calibration runs very quickly compared to the software calibration. It can be useful to initially run the automatic hardware calibration to find stable, but not necessarily optimal DDR PHY settings.

Automatic hardware calibration is enabled by setting the **Board Specific IMX6->DDR Auto & Manual->Fixed PHY Calibration** field to 0 as shown in Figure 27.

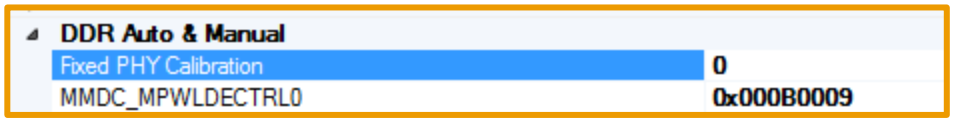


Figure 27: Fixed PHY Calibration Automatic

When automatic hardware calibration is enabled, the command DDR configuration process displays the calibration results. The output from the hardware calibration executed on a Sabre-Lite reference design from NXP is shown in Figure 28.

```
Configuring MMDC channel 1 for DDR3
Resetting MMDC channel 1
DDR total size = 1 GiB
Performing HW based calibration for DDR3
    Starting ZQ calibration
    ZQ calibration complete
    Starting write leveling calibration
    Write leveling complete
    Preparing DQS gating and delay line calibration
    Starting DQS gating calibration
    DQS gating calibration completed.
    Starting read delay line calibration.
    Read calibration completed.
    Starting write delay line calibration.
    Write calibration completed.
    Starting write delay line calibration.
    Write calibration completed.
HW based calibration completed successfully

//content truncated//
```

Figure 28: Auto Calibration Results

At this point you should run all the DDR tests to confirm the DDR settings and PHY calibration values are good. After confirming calibration values on several boards, you can use fixed calibration values by copying and pasting the calibration results back into your board file.

The manual way

Should the user want to explicitly set the values for all the DDR PHY registers, this can be accomplished, however it is not covered in this paper, but just briefly mentioned here for awareness of the capability.

Register Based Board Files

Use a Register-Based DDR board file if you want to explicitly specify all register values programmed into the DDR Memory Controller. See the user manual for more data on register-based DDR board files.

Timing Based Files

Use a Timing-Based DDR board file to have PFx DDR automatically calculate the register settings for the DDR memory controller. Register settings are based on the memory part file selected and the selected speed of the DDR interface.

In your Timing Mode board file, you must provide the following items:

- Specify the memory part
- Specify the DDR clock frequency
- Specify the number DDR chips used
- Specify DDR termination
- Specify whether hardware-based DDR training is used or whether manual DDR training settings are used.

Refer to the user manual for more details.

Conclusions

DDR tuning is a process that cannot be taken lightly. Ensuring that the settings for the DDR controller and PHY are optimal for your design is critical to producing a quality product and staying competitive. ASSET has presented a software approach to DDR tuning/calibration to eliminate the tedium and potentially error prone method of translating datasheet information. The software also provides robust memory testing, which is critical to the tuning processes and is supported on all NXP i.MX 6 SoC regardless of board architecture.

The process involved these six steps.

1. Setting the board information
    a. DDR Frequency
    b. Number of chip selects
    c. Number of memory devices
    d. CPU Frequency
2. Loading the memory device from the library
3. Run the initial DDR configuration
4. Testing the initial configuration
5. Run the Tuning Agent
6. Copying the tuning data to the project/design

Other items to consider are the environmental requirements of your product and the consequences in production with an Engineering Change Order (ECO). Either chamber testing or ECO will necessitate a methodology for tuning the DDR memory system. If the ECO involves the memory subsystem, trace lengths or memory device changes are but two areas to be mindful of, tuning must again be addressed. PFTDDR is a tool that can handle all these scenarios and many more with only minimal data provided to the tool by user.