

LMK5Cxxxxx/LMK5Bxxxxx Programming

July 15 2024

Clocks and Timing Solutions

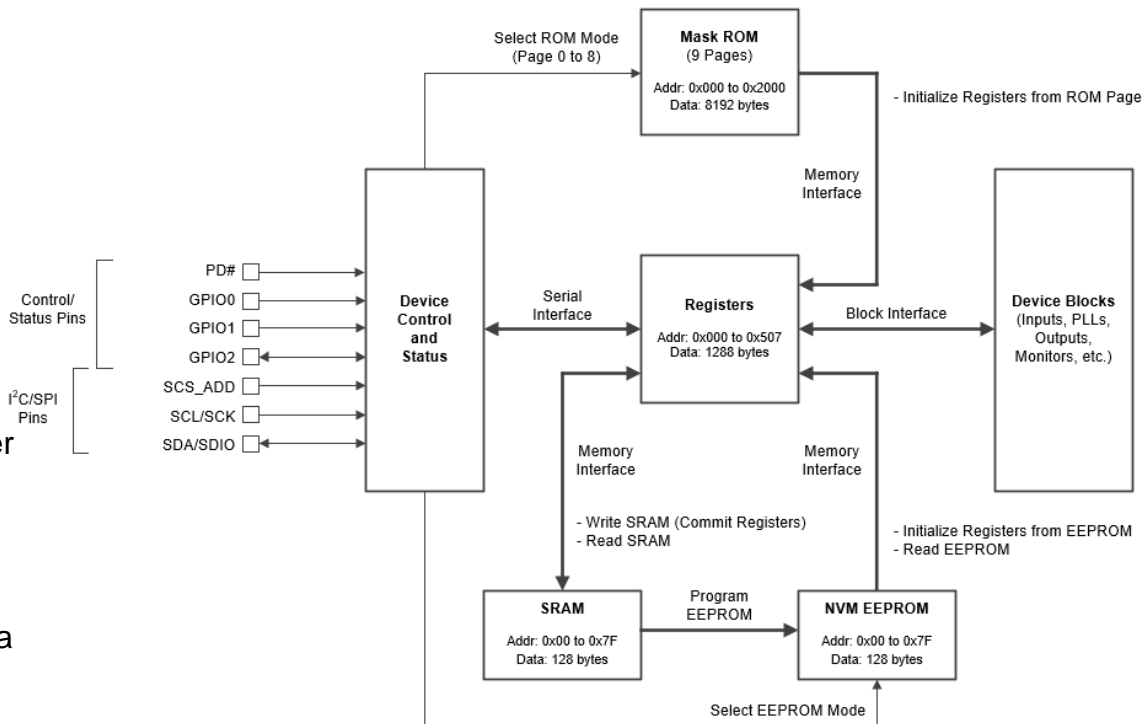
Revision History

Date	Description
2022-09-23	Initial release
2022-11-29	Updated to poll or delay before setting NVMUNLK = 0 after starting programming. Added notes on ROM page selection and outputs
2023-01-03	Generalized title for LMK5Cxxxxx and LMK5Bxxxxx
2023-05-17	Added Memory Overview
2023-08-11	Updated SRAM/EEPROM address: 0x00 to 0x7F (128 bytes) Updated register address in Dec and Hex Updated RAMDAT register Updated direct write method steps Added LMK5C ROM page
2023-11-03	Added in-system programming section, changed title
2024-01-24	Added start-up details and clarified memory overview
2024-02-14	Clarified start-up sequence and added DPLLx_SWRST details.
2024-07-15	Added Read EEPROM and NVMDAT information

Memory Overview

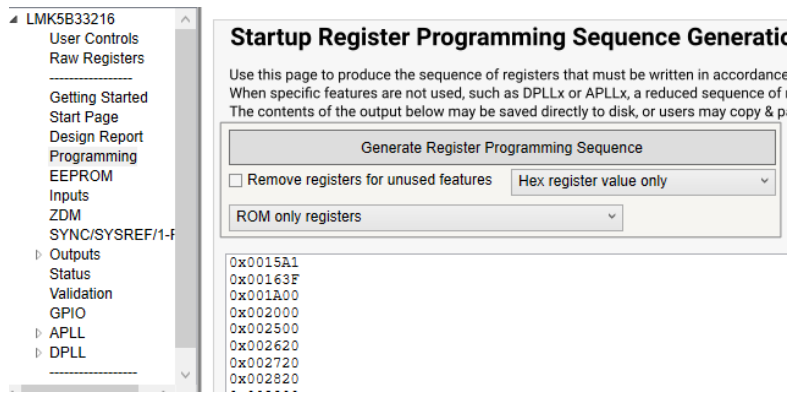
Memory Overview

- ROM
 - Contains **all** register settings (DPLL, APLL, output, SYSREF, GPIO).
 - One-time programming by TI.
- EEPROM
 - Contains **partial** register settings (APLL and output).
 - Up to 100 programming cycles by user
- SRAM
 - Intended use is for EEPROM programming only.
 - Matches the EEPROM address & data mapping.
- Registers
 - Current/active settings used by device.



About Registers Not Stored in EEPROM

- The EEPROM contains partial register settings and does not store the DPLL settings.
 - With **only** the EEPROM settings loaded, the outputs can be present but will not be synchronized (locked) to INx inputs and instead will lock to XO input.
 - If desired **DPLL settings** are not available in an existing ROM page, in-system programming (I2C/SPI) must occur after boot-up to configure the remaining registers.
- The registers needed after an EEPROM boot-up can be determined in TICSPRO.
 - Navigate to the LMK5B33216 profile.
 - Load the desired .tcs file.
 - Go to the “Programming” page.
 - Select **ROM only registers** in the drop-down menu.
 - Click **Generate Register Programming Sequence**.



Start-Up Flow

Device Start-Up Flow Options to Get Desired Registers Configured

- **Opt. #1: ROM**
 - Use when both DPLL & APLL settings match a ROM page.
- **Opt. #2: ROM → EEPROM**
 - Use when desired DPLL settings match a ROM page but APLL settings do **not**.
 - Also, use for free-run mode (APLL only) configurations.
- **Opt. #3: ROM → EEPROM → in-system programming**
 - Use when desired DPLL and APLL settings do **not** match a ROM page.
- **Opt. #4: ROM → in-system programming**
 - Use when EEPROM is not used and *all* registers are configured in-system.

SWRST and DPLLx_SWRST

- **SWRST, R23[6]**
 - Global software reset
 - Recommended to toggle (0→1→0) when **APLL** settings are modified in-system.
 - Recalibrates the APLLs.
 - Causes a momentary glitch on the outputs.
- **DPLLx_SWRST, R23[5:3]**
 - DPLL software reset
 - Recommended to toggle (0→1→0) when **DPLL** settings are modified in-system.
 - Restarts the DPLL state machine.
 - Does not cause glitch on the outputs.

Option #1: ROM Only

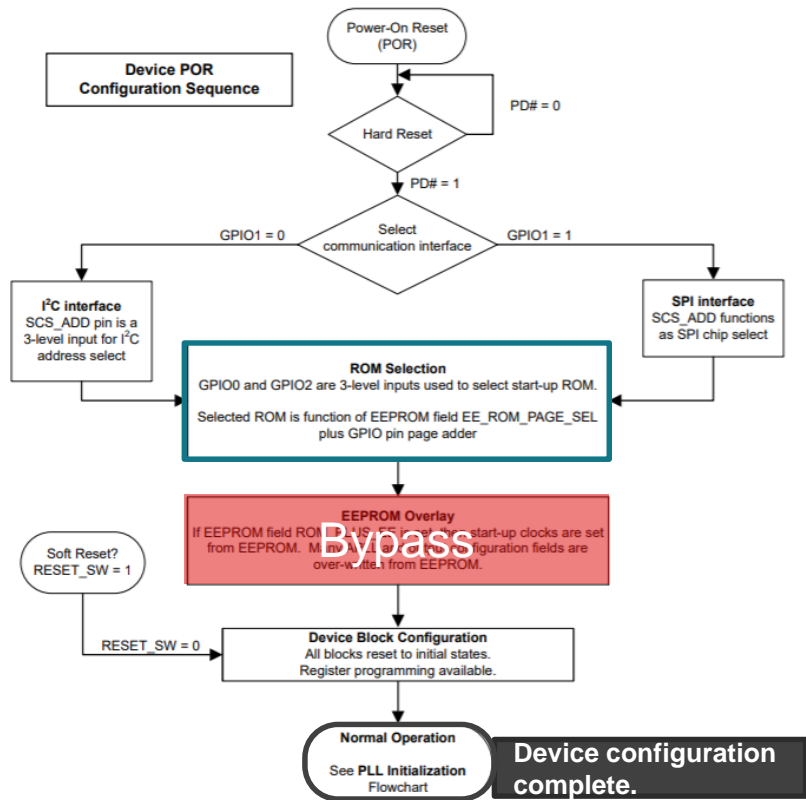
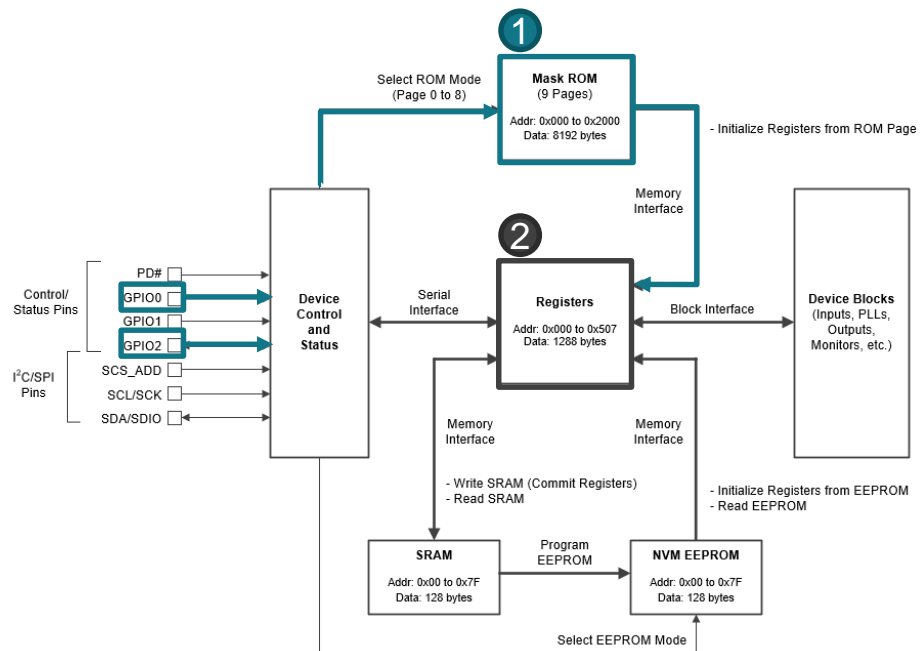


Figure 9-36. Device POR Configuration Sequence



Option #2: ROM → EEPROM

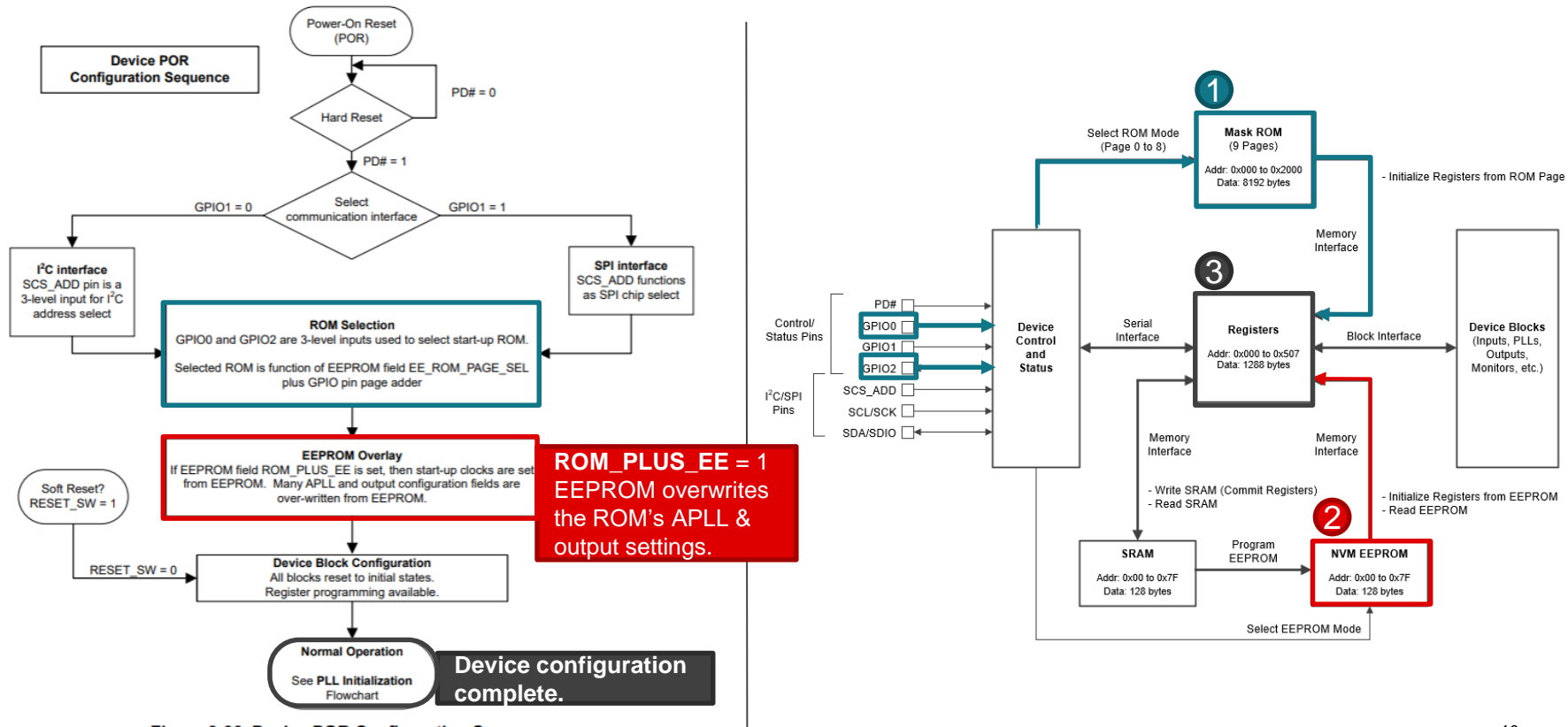


Figure 9-36. Device POR Configuration Sequence

Option #3: ROM → EEPROM → In-System

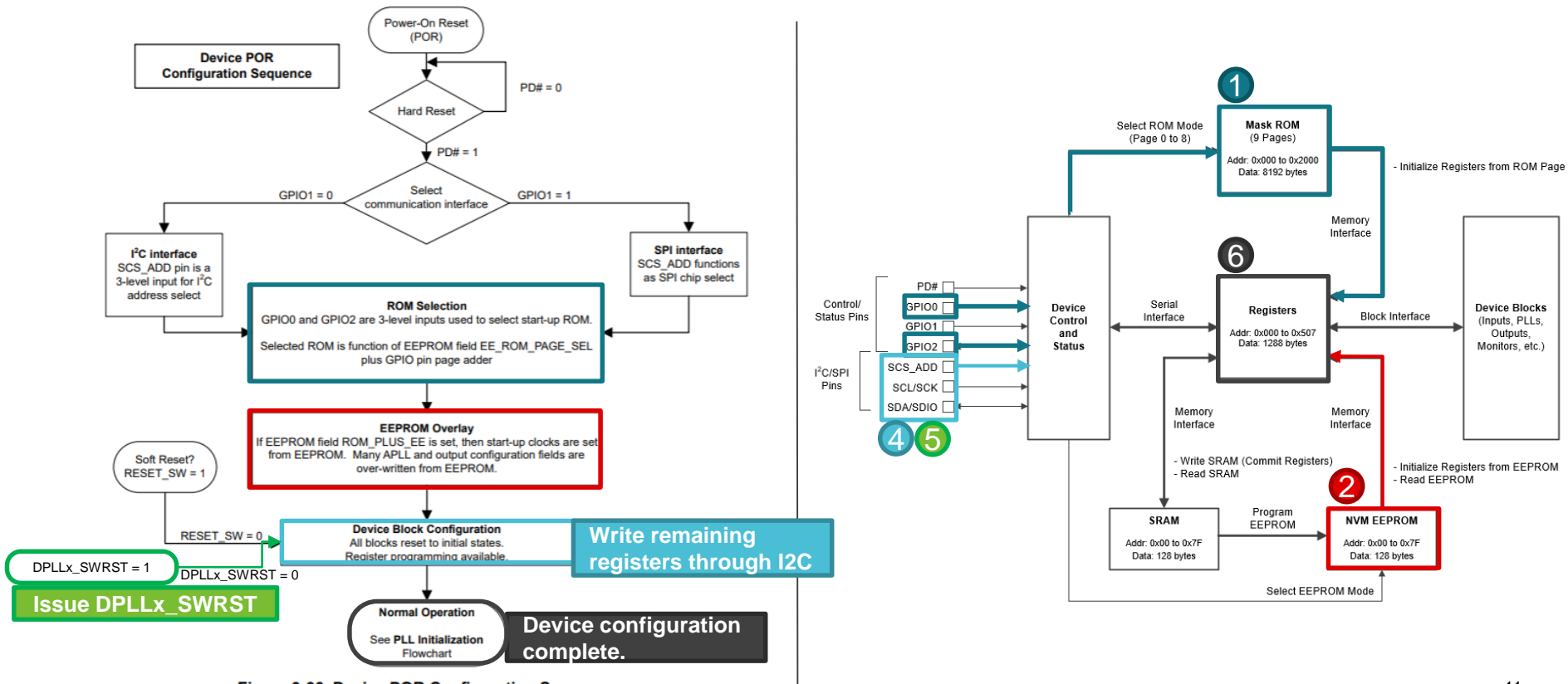


Figure 9-36. Device POR Configuration Sequence

Option #4: ROM → In-System

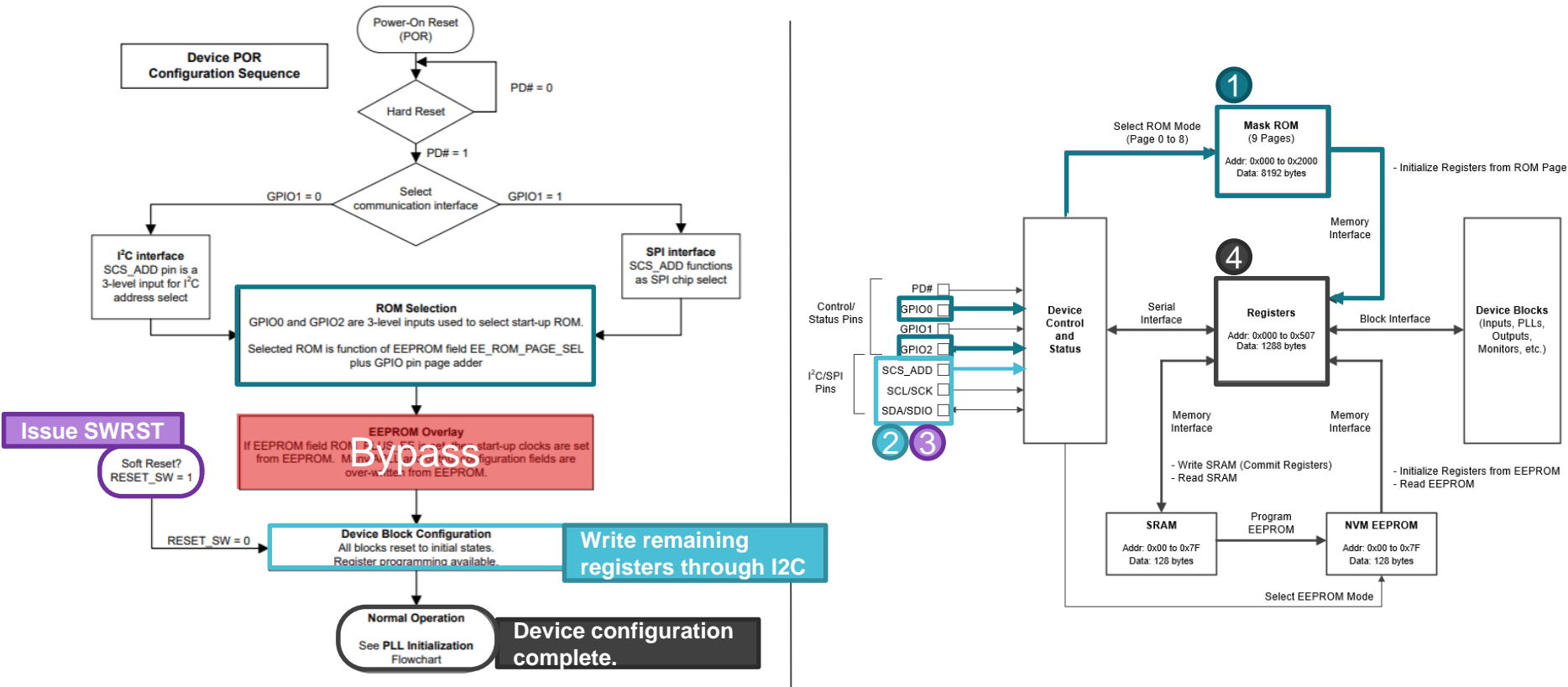


Figure 9-36. Device POR Configuration Sequence

Additional Start-Up Details

Part Identification

	Prototype (never released to ti.com)	Released to ti.com
Date	Jan 2022	July 2022
Top Marking	PK5B33216	K5B33216
PRODID, R2	65 (0x41)	65 (0x41)
REVID, R3	0	1
PARTID, R4-R9	Unique to every part	
NVMCNT, R16	1 + number of EEPROM programming cycles made by user	
EEREV	Set by user when programming EEPROM using SRAM direct write method	

I2C and APLL Lock Times

- I2C communication to device can begin about 30ms after power-up.
- After power-up, issuing a SWRST, or toggling the PD pin, the APLL lock time is listed (from datasheet):

$t_{\text{APLL1-LOCK}}$	APLL1 lock time	Time between soft or hard reset and stable APLL1 output.	20	35	ms
$t_{\text{APLL2-LOCK}}$	APLL2 lock time	Time between soft or hard reset and stable APLL2 output.	350	460	ms
$t_{\text{APLL3-LOCK}}$	APLL3 lock time.	Time between soft or hard reset and stable APLL3 output.	12.5	13	ms

Start-up Sequence Recommendations from TICSPRO

1. Open TICSPRO and load your desired .tcs settings.
2. Navigate to the “Programming” page.
3. Check the “Generate for Startup Programming” box.
4. (Optional) Check the “Remove registers for unused features” box.
 1. This will not list registers for unused settings. For example, if DPLL2 is unused, then DPLL2 disable register will be listed but DPLL2 loop filter registers will not be listed.
5. Hit “Generate Register Programming Sequence” button.
6. To get the instructions as a .txt, click “Save Contents to Text File”.

Startup Register Programming Sequence Generation

Use this page to produce the sequence of registers that must be written in accordance with the datasheet requirements. When specific features are not used, such as DPLLx or APLLx, a reduced sequence of registers may be produced instead. The contents of the output below may be saved directly to disk, or users may copy & paste into Excel, Notepad, or another application.

Remove registers for unused features Generate for Startup Programming

Hex register value only

```
# The comments below are the steps for implementing glitchless device programming.
# The function block("0=LOL_PLLx",timeout_seconds = 1) polls the status of the APLLs to determine if they are
# locked, max 1 s.
# During this time, TICS Pro usage is blocked.
# The function block("0=LOL_PLLx",timeout_seconds = 1) can be replaced with open_loop_delay(seconds=1) to delay
# for 1 s
# This is as opposed to checking the APLL lock status.

# Step 1: Set all outputs to static low to ensure that there are no glitches at startup.
# Set OUT0 to static low.
0x03c230 # OUT0 L
0x03c322 # OUT0 Static DC
# Set OUT1 to static low.
0x03c500 # OUT1 L
```


Start-up CRC ERROR Debugging

- If NVMCRCERR is set ($R171[5] = 1$), then there was an error loading EEPROM contents onto the registers.
- At power-up, it is expected for NVMSCRC (R170) to equal NVMLCRC (R172).
- Debugging tips
 - Power cycle the board. Does the NVMCRCERR clear?
 - Reprogram the EEPROM ensuring the proper programming sequence is followed. Does the NVMCRCERR clear?

About NVMCNT, R16

- NVMCNT is the number of successful EEPROM programming cycles.
- All LMK5Bxxxx parts from [ti.com](https://www.ti.com) start with NVMCNT = 1 because they are preprogrammed with the default EEPROM configuration.
 - Default EEPROM contains the APLL and output driver settings from ROM0.

GPIO2 at POR	GPIO0 at POR	ROM page with EE_ROM_PAGE_SEL = 0
L	L	ROM page 0. XO= 48 MHz, REFCLK = 25MHz, outputs 25 MHz, 100 MHz, 155.52 MHz, 156.25 MHz, 161.128125 MHz, 312.5MHz.

- If NVMCNT > 1, then the user has modified the EEPROM beyond factory setting.

EEPROM Programming

EEPROM Programming Methods

- Two methods available:
 - **REGCOMMIT (recommended)**
 - Stores current configuration (all active registers) to EEPROM.
 - TICS Pro uses this method to program when you press **Program EEPROM** button.
 - Recommended method when modifying the I2C_ADDR or EEREV is not required.
 - **SRAM Direct Write**
 - Programs EEPROM one register/address at a time.
 - Use if you want to do an in-system update without disrupting clocks. On next restart new config takes effect.
 - Use to change the 5 MSBs of the I2C MSB address (TARGET_ADR_MSB) by writing to SRAM address 12.
 - Use to change the EEPROM Revision Number (EEREV) by writing to SRAM address 13.
 - Does require you to utilize the saved ".EPR" file.

EEPROM Programming: REGCOMMIT Method (recommended)

1. Power cycle (toggle PD#)
2. Program active registers and confirm the current configuration outputs as desired
3. Enable EEPROM overlay
 - Set ROM_PLUS_EE (R20[7]) = 1 → R20 = 0x80
4. Commit active registers to SRAM
 - Set REGCOMMIT (R171[6]) = 1 → R171 = 0x40
 - Note: REGCOMMIT is auto-cleared to 0 when transfer is completed**
5. Unlock EEPROM
 - Set NVMUNLK = 234 → R180[7:0] = 0xEA
6. Erase EEPROM and initiate EEPROM programming
 - Set NVMERASE (R171[1]) = 1 → R171 = 0x03
 - Set NVMPROG (R171[0]) = 1
 - Note: Step 5 & 6 must be atomic writes without any other register transactions in-between**
7. Wait for EEPROM programming to finish
 - Poll NVMBUSY, R171[2], until cleared or wait ~ 500 ms
8. Lock EEPROM
 - Set NVMUNLK = 0 → R180[7:0] = 0x00
9. Power cycle and check outputs to confirm EEPROM programming was successful

The screenshot shows the TICS Pro software interface for the LMK5B33414 device. The interface is divided into several sections:

- EEPROM / NVM Status (read only):** This section contains a "Read EEPROM/NVM Status" button, a "CRC Error Status" checkbox, a "Stored CRC" field, an "NVM Program Count" field (set to 36), an "I2C Target Address MSB" field (set to 12), and an "EEREV" field (set to 0).
- Device start-up behavior:** This section includes a "ROM Page Selection, GPIO0 & 2 add to this value on POR" dropdown (set to 0) and a checked checkbox for "Enable EEPROM overlay on POR".
- Write EEPROM:** This section has a "Program EEPROM" button.
- EEPROM Register Programming Sequence Generation:** This section includes a "Design Name" field, a "User Notes" field, and a "Generate register programming sequence using:" dropdown menu with "Register commit method" selected.
- OUTPUT (display only):** This section displays a list of register addresses and their values, such as R1188: 0x04440C, R1189: 0x04A521, R1190: 0x04A6A3, R1192: 0x04A800, R1194: 0x04AA10, R1217: 0x04C149, R1218: 0x04C20C, R1219: 0x04C34E, R1220: 0x04C40C, R1221: 0x04C521, R1222: 0x04C6A3, R1224: 0x04C800, R1226: 0x04CA09, R1248: 0x-00001, R1249: 0x-00001, R1250: 0x-00001, R1253: 0x-00001, R1255: 0x-00001, R1280: 0x-00001, R1281: 0x-00001, R1282: 0x-00001, R1285: 0x-00001, R1287: 0x-00001.

Annotations in the image include:

- A blue bracket on the left side of the interface, spanning from the "EEPROM" section down to the "Write EEPROM" section.
- A red bracket on the left side of the interface, spanning from the "EEPROM" section down to the "EEPROM Register Programming Sequence Generation" section.
- A blue box around the "Enable EEPROM overlay on POR" checkbox.
- A red box around the "Program EEPROM" button.
- A green box around the "Register commit method" dropdown.
- A green box around the "Save EEPROM output to text file" button.
- Green arrows pointing from the "Register commit method" dropdown to the text "Export the register programming sequence" and from the "Save EEPROM output to text file" button to the text "Save EEPROM output to text file".

Export the register programming sequence

EEPROM Programming: SRAM Direct Write Method

Write to
SRAM

1. Enable EEPROM overlay
2. Commit active registers to SRAM, set REGCOMMIT (R171[6]) = 1
Note: REGCOMMIT is auto-cleared to 0 when transfer is completed
3. Write the most significant five bits of the SRAM address to R173[4:0] (MEMADR_12:8).
4. Write the least significant eight bits of SRAM address to R174 (MEMADR).
5. Write the SRAM data byte to R176 (RAMDAT).
6. Repeat steps 3-5 for all desired SRAM values

Program
EEPROM


7. Unlock EEPROM
8. Erase EEPROM and initiate EEPROM programming
 - Set NVMERASE (R171[1]) = 1
 - Set NVMPROG (R171[0]) = 1**Note: Step 7 & 8 must be atomic writes without any other register transactions in-between**
9. Wait for EEPROM programming to finish
 - Poll NVMBUSY, R171[2], until cleared or wait ~ 500 ms
10. Lock EEPROM
 - Set NVMUNLK = 0
11. Power cycle and check outputs to confirm EEPROM programming was successful.

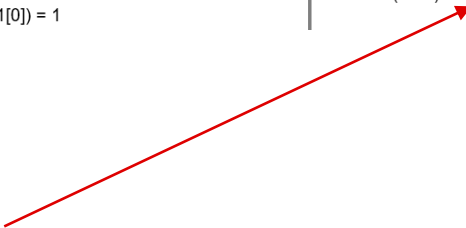
Example for updating the EEREV (R19) to value of 2:

- | | | |
|-----|---|-----------------------------------|
| 1. | R20 (0x14) = 0x80 | # ROM_PLUS_EE = 1 |
| 2. | R171 (0xAB) = 0x40 | # REGCOMMIT = 1 |
| 3. | R173 (0xAD) = 0x00 | # EEREV addrH is 0x00 |
| 4. | R174 (0xAE) = 0x0D | # EEREV addrL is 0x0D |
| 5. | R176 (0xB0) = 0x02 | # EEPROM Rev ID to 2 |
| 7. | R180 (0xB4) = 0xEA | # NVMUNLK = 234 |
| 8. | R171 (0xAB) = 0x03 | # NVMERASE = 1 &
NVMPROG = 1 |
| 9. | Poll NVMBUSY, R171[2], until cleared or wait ~ 500 ms | |
| 10. | R180 (0xB4) = 0x00 | # NVMUNLK = 0 |

Important Notes on EEPROM

- The I2C bus should not be **interrupted** (such as writing to another I2C address) in between NVMUNLK and NVMERASE/NVMPROG transactions.
 - If interrupted, EEPROM write will fail.

- 
7. Unlock EEPROM
 8. Erase EEPROM and initiate EEPROM programming
 - Set NVMERASE (R171[1]) = 1
 - Set NVMPROG (R171[0]) = 1

- 
7. R180 (0xB4) = 0xEA
 8. R171 (0xAB) = 0x03
- # NVMUNLK = 234
NVMERASE = 1 &
NVMPROG = 1

- NVMERASE and NVMPROG are atomic writes.
 - Ensure both are written in **one** register transaction.

Read from EEPROM

1. Write the most significant five bit of the EEPROM address to R173[4:0] (MEMADR byte 1).
 2. Write the least significant eight bits of the EEPROM address to R174 (MEMADR byte 0).
 3. Read R175 (NVMDAT byte) to get the EEPROM data at the specified address.
- **Any additional read transfer that is part of the same transaction will cause the EEPROM address pointer to be auto-incremented and a subsequent read will take place of the next address.**
 - Byte or Block read transfers from R161 can be used to read the entire EEPROM map sequentially from Byte 0 to 252.
 - Access to EEPROM will terminate at the end of current register transaction.

Example for reading the EEREV (R19) to value of 2:

- | | |
|----------------------|------------------------------|
| 1. Write(0xAD, 0x00) | # EEREV addrH is 0x00 |
| 2. Write(0xAE, 0x0D) | # EEREV addrL is 0x0D |
| 3. Read(0xAF) → 0x02 | # Readback EEPROM REVID as 2 |

R20 ROM_PLUS_EE and EE_ROM_PAGE_SEL

```
# Use EEPROM overlay with ROM  
R20 0x001480
```



```
ROM_PLUS_EE = 1  
EE_ROM_PAGE_SEL = 0
```

1.8 R20 Register (Offset = 0x14) [Reset = 0x0]

R20 is shown in [Table 1-10](#).

Return to the [Summary Table](#).

Note: Turn off EEPROM overlay by programming again using ROM_PLUS_EE = 0

Note: GPIO0, GPIO1, and EE_ROM_PAGE_SEL will select starting ROM page. ROM page will impact some clock output configuration pertaining to which clocks are SYSREF. ROM8 allows any clock to startup as a non-SYSREF output and is a good general startup ROM.

Table 1-10. R20 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	ROM_PLUS_EE	R/W	0x0	When set, the thin EEPROM settings are loaded. This is user writeable to EEPROM only through SRAM register address 14. ROM=N, EEPROM=Y
6:3	EE_ROM_PAGE_SEL	R/W	0x0	EE_ROM_PAGE_SEL value is added to the GPIO pin value for selecting the start-up ROM. ROM=N, EEPROM=Y
2:0	RESERVED	R	0x0	Reserved

R171 REGCOMMIT

```
# Write EEPROM sequence
```

```
# REGCOMMIT, regs to SRAM, self clearing
```

```
R171 0x00AB40
```



REGCOMMIT = 1

1.95x0]

R171 is shown in [Table 1-97](#).

Return to the [Summary Table](#).

Table 1-97. R171 Register Field Descriptions

Bit	Field	Type	Reset	Description
7	RESERVED	R	0x0	Reserved
6	REGCOMMIT	R/WSC	0x0	Copy fields which also exist in SRAM to SRAM memory. The REGCOMMIT bit is automatically cleared to 0 when the transfer is complete. Next an EEPROM programming operation may be performed to update NVM EEPROM. When programming to alter an NVM profile, it is suggested to toggle PD# to assure default conditions, change the desired fields, then assert the REGCOMMIT bit. ROM=N, EEPROM=N

R173 MEMADR_12:8

1.91 R173 Register (Offset = 0xAD) [Reset = 0x00]

Return to the [Summary Table](#).

Table 1-93. R173 Register Field Descriptions

Bit	Field	Type	Reset	Description
7:5	RESERVED	R	0x0	Reserved
4:0	MEMADR_12:8	R/W	0x0	See Register 174

R174 MEMADR

1.92 R174 Register (Offset = 0xAE) [Reset = 0x00]

Return to the [Summary Table](#).

Table 1-94. R174 Register Field Descriptions

Bit	Field	Type	Reset	Description
7:0	MEMADR	R/W	0x0	Memory Address. The MEMADR value determines the starting address for access to the on-chip memories. This same MEMADR value is used for EEPROM and SRAM access which share the same memory map and also ROM access. The NVMDAT field is used to read and write from EEPROM. The RAMDAT field is used to read and write from SRAM. The ROMDAT field is used to read and write from ROM.

R176 RAMDAT

1.93 R176 Register (Offset = 0xB0) [Reset = 0x00]

Return to the [Summary Table](#).

Table 1-95. R176 Register Field Descriptions

Bit	Field	Type	Reset	Description
7:0	RAMDAT	R/W	0x0	RAM Read/Write Data. The first time an I2C/SMBus read or write transaction accesses the RAMDAT register address, either because it was explicitly targeted or because the address was auto-incremented, a read transaction will return the RAM data located at the address specified by the MEMADR register and a write transaction will cause the current I2C/SMBus data to be written to the address specified by the MEMADR register. Any additional accesses which are part of the same transaction will cause the RAM address to be incremented and a read or write access will take place to the next SRAM address. The I2C/SMBus address will no longer be auto-incremented (that is, the I2C/SMBus address will be locked to the RAMDAT register after the first access). Access to the RAMDAT register will terminate at the end of the current I2C/SMBus transaction. ROM=N, EEPROM=N

R175 NVMDAT

- EEPROM Read Data.
- The first time an I2C/SMBus read transaction accesses the NVMDAT register address, either because it was explicitly targeted or because the address was auto-incremented, the read transaction will return the EEPROM data located at the address specified by the MEMADR register. Any additional read's which are part of the same transaction will cause the EEPROM address to be incremented and the next EEPROM data byte will be returned. The I2C/SMBus address will no longer be auto-incremented, i.e the I2C/SMBus address will be locked to the NVMDAT register after the first access. Access to the NVMDAT register will terminate at the end of the current I2C/SMBus transaction.

R180 NVMUNLK

NVMUNLK = 234 (0xea)
#R180 0x00B4EA

—————→ NVMUNLK = 0xEA

1.99 R180 Register (Offset = 0xB4) [Reset = 0x0]

R180 is shown in [Table 1-101](#).

Return to the [Summary Table](#).

Table 1-101. R180 Register Field Descriptions

Bit	Field	Type	Reset	Description
7:0	NVMUNLK	R/W	0x0	NVM Prog Unlock. The NVMUNLK register must be written immediately prior to setting the NVMERASE and NVMPROG bit, otherwise the Erase/Program cycle will not be triggered. NVMUNLK must be written with a value of 0xEA. ROM=N, EEPROM=N

R171 NVMERASE and NVMPROG

NVMERASE & PROG = 1, self clearing
#R171 0x00AB43

REGCOMMIT = 1
NVMERASE = 1
NVMPROG = 1

Table 1-97. R171 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	NVMCRCERR	R	0x0	NVM CRC Error Indication. The NVMCRCERR bit is set to 1 if a CRC Error has been detected when reading back from on-chip EEPROM during device configuration. ROM=N, EEPROM=N
4	RESERVED	R	0x0	Reserved
3	RESERVED	R	0x0	Reserved
2	NVMBUSY	R	0x0	NVM Program Busy Indication. The NVMBUSY bit is 1 during an on-chip EEPROM Erase/Program cycle. While NVMBUSY is 1 the on-chip EEPROM cannot be accessed. Toggling PD# or removing power while NVMBUSY is asserted will corrupt the EEPROM. ROM=N, EEPROM=N
1	NVMERASE	R/WSC	0x0	NVM Erase Start. The NVMERASE bit is used to begin an on-chip EEPROM Erase cycle. The Erase cycle is only initiated if the immediately preceding I2C/SMBus transaction was a write to the NVMUNLK register with the appropriate code. The NVMERASE bit is automatically cleared to 0.
0	NVMPROG	R/WSC	0x0	NVM Program Start. The NVMPROG bit is used to begin an on-chip EEPROM Program cycle. The Program cycle is only initiated if the immediately preceding I2C/SMBus transaction was a write to the NVMUNLK register with the appropriate code. The NVMPROG bit is automatically cleared to 0.

R180 NVMUNLK

NVMUNLK = 0
#R180 0x00B400 → NVMUNLK = 0

1.99 R180 Register (Offset = 0xB4) [Reset = 0x0]

R180 is shown in [Table 1-101](#).

Return to the [Summary Table](#).

Table 1-101. R180 Register Field Descriptions

Bit	Field	Type	Reset	Description
7:0	NVMUNLK	R/W	0x0	NVM Prog Unlock. The NVMUNLK register must be written immediately prior to setting the NVMERASE and NVMPROG bit, otherwise the Erase/Program cycle will not be triggered. NVMUNLK must be written with a value of 0xEA. ROM=N, EEPROM=N

LMK5C ROM page

Table 9-6. ROM page selection

GPIO2 ROM_ADD[1]	GPIO0 ROM_ADD[0]	ROM page with EE_ROM_PAGE_SEL = 0
H	H	Low power mode. All PLLs off, all outputs off.
H	M	IN1 = 10 MHz SE 50-Ω termination, XO = 38.88 MHz. All outputs = 100 MHz LVDS from DPLL1 (OUT0 to 3, 14, 15) and DPLL2 (OUT4 to 13).
L	L	IN1 = 10 MHz CMOS, XO = 38.88 MHz, OUT2 = 125 MHz HSDS, OUT3 = 156.25 MHz LVDS, OUT10 = 122.88 MHz LVDS.

LMK5B ROM page

Table 8-5. ROM Page Selection

GPIO2 at POR	GPIO0 at POR	ROM page with EE_ROM_PAGE_SEL = 0
L	L	ROM page 0. XO= 48 MHz, REFCLK = 25MHz, outputs 25 MHz, 100 MHz, 156.25 MHz, 161.128125 MHz, 312.5MHz.
L	H	ROM page 1. XO = 48MHz, outputs 25 MHz, 50 MHz, 100 MHz.
H	L	ROM page 2. XO= 48 MHz, REFCLK = 25MHz, all outputs 156.25 MHz.
H	H	ROM page 3. Low power mode. All PLLs off, all outputs off.
L	M	ROM page 4. XO = 49.152 MHz, REFCLK = 19.44 MHz, outputs 100MHz, 312.5 MHz, 800 MHz.
M	L	ROM page 5. XO= 48 MHz, REFCLK = 156.25MHz, outputs 100 MHz, 125 MHz, 156.25 MHz
M	M	ROM page 6. XO= 48 MHz, REFCLK = 25MHz, all outputs 312.5 MHz.
M	H	ROM page 7. XO= 48 MHz, REFCLK = 156.25MHz, outputs 100MHz, 125 MHz, 156.25 MHz.
H	M	ROM page 8. XO= 48.008 MHz, REFCLK = 156.25MHz, outputs 25 MHz, 50 MHz, 100 MHz, 156.25 MHz