AN5898

Programming PCI1xxxx OTP

Author: *Christopher Brisco*
Microchip Technology, Inc.

1.0 INTRODUCTION

The One-Time Programmable (OTP) memory of the PCI1xxxx family of devices is a critical component for customizing and securing device configurations. Unlike some programmable memories, the OTP in these devices is not shipped blank. It contains a default image that must be considered when making any modifications. When programming the OTP, it is important to note that bits can only be changed from 0 to 1. Trying to set a 1 to a 0 will result in an error. Therefore, any new OTP image must be carefully constructed based on the current image to ensure that only the desired bits are altered. This app note provides detailed guidance on how to dump the current OTP image and effectively program a new OTP image to the PCI1xxxx family, ensuring reliable and accurate updates.

1.1 Sections

This application note covers the following topics:

- Section 2.0, OTP Structure
- Section 3.0, Header
- Section 4.0, Memory Regions
- Section 5.0, Dumping OTP
- Section 6.0, Generating OTP Image
- Section 7.0, Programming OTP Image

1.2 References

Consult the following documents when using this application note:

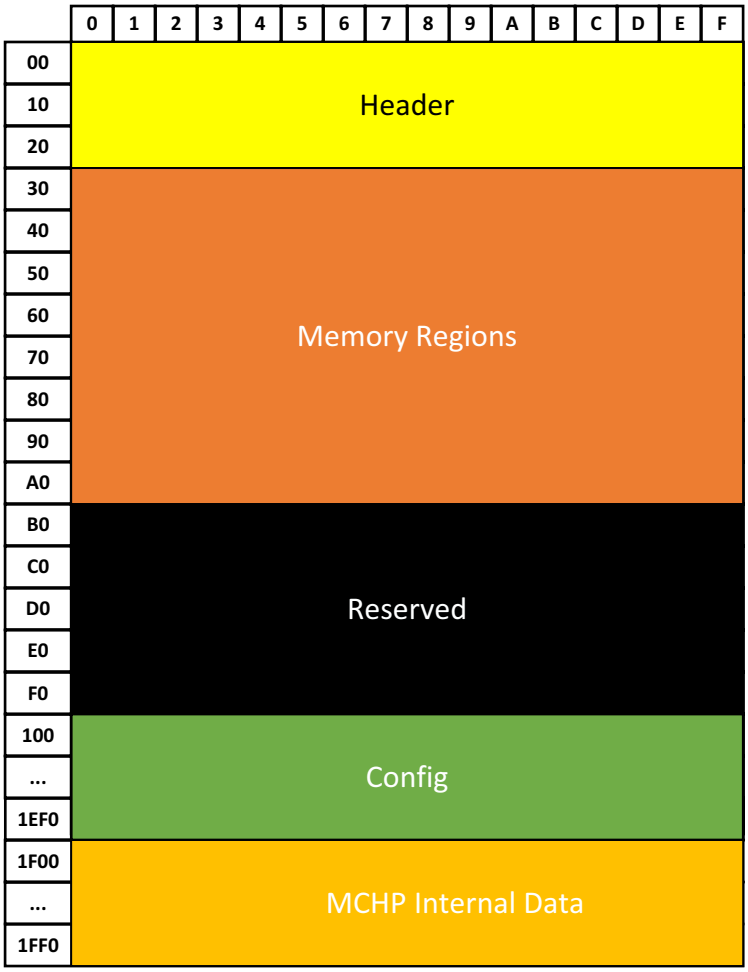
- *PCI11010/PCI11101/PCI12000/PCI11400/PCI11414 Data Sheets*
- *AN5213 Configuration and Programming Options for the PCI1xxxx*
- *MPLAB® X IDE User's Guide*

2.0 OTP STRUCTURE

2.1 OTP Map

Figure 1 gives the layout of the OTP. When programming the OTP, the Header, Memory regions, and Config sections are all updated. The MCHP Internal Data section should not be touched.

FIGURE 1: OTP MEMORY MAP



Note: 0x1F00-0x1FFF constitute Microchip specific data and should not be written to.

3.0 HEADER

The format of the header is shown in Figure 2.

FIGURE 2: HEADER MAP

3.1 Magic Byte (MB)

This should always read 0xa5. If a magic byte other than 0xa5 is detected, PCI1xxxx will skip loading from OTP entirely.

3.2 Memory Info (MI)

This byte is used by MPLAB Connect only. A value of `0x01` indicates this is an OTP image.

3.3 OTP Prog Count

OTPs are not shipped blank from the factory. This indicates how many times the OTP has been programmed and should be incremented each time it is programmed. Bit 0 corresponds to one (1) time programmed, bit 1 corresponds to two (2) times programmed, etc. Bit 15 means the OTP has been programmed at least 16 times.

3.4 Memory Size

This value indicates the total size of the configuration memory available. PCI1xxxx is 8K.

3.5 Memory Regions Programmed

This indicates which memory regions have been programmed. Bit 0 corresponds to memory region 0, bit 1 corresponds to memory region 1, etc.

3.6 OTP Regions Invalidated

This indicates which memory regions have been invalidated. Bit 0 corresponds to memory region 0, bit 1 corresponds to memory region 1, etc. A value of 1 indicates an invalidated region, and a value of 0 indicates a valid memory region.

PCI1xxxx allows for invalidating memory regions. This is used to prevent the PCI1xxxx from loading specific chunks of the OTP during configuration. Note that due to the nature of OTP, this change is permanent. If memory region 0 is invalidated, it cannot be revalidated. A new memory region must be used containing the desired register writes.

3.7 OTP Regions Write Protected

This indicates which memory regions have been write protected. Bit 0 corresponds to memory region 0, bit 1 corresponds to memory region 1, and so on. A value of 1 indicates a write protected region, and a value of 0 indicates a writable region.

PCI1xxxx allows for write protecting memory regions. This prevents accidental or purposeful changes to bits in that memory region. Note that just like invalidating memory regions, this is a permanent change to that memory region.

3.8 OTP Regions Read Protected

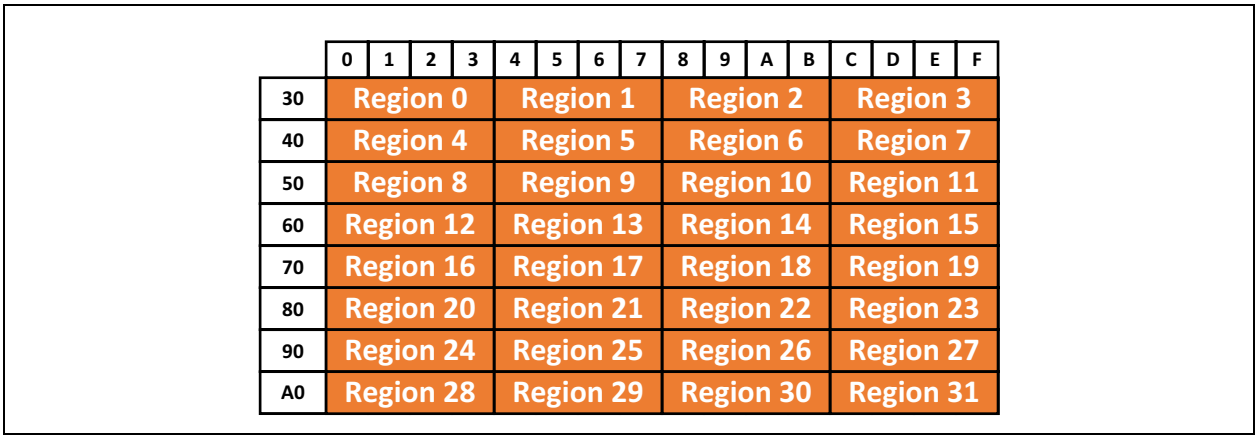
This indicates which memory regions have been read protected. Bit 0 corresponds to memory region 0, bit 1 corresponds to memory region 1, and so on. A value of 1 indicates a read protected region, and a value of 0 indicates a readable region.

PCI1xxxx allows for read protecting memory regions. This is different from invalidating memory regions in that invalidating memory regions prevents the PCI1xxxx from loading that part of configuration whereas read protecting loads that part of configuration, but will not allow read access at runtime.

4.0 MEMORY REGIONS

The purpose of the memory regions is to indicate where in memory a block of configurations starts, how long that block is, and which memory tag that block is associated with. There are 32 memory regions available and they are mapped according to Figure 3.

FIGURE 3: MEMORY REGIONS



Each memory region is four (4) bytes, and the bit definitions are given in Table 1.

TABLE 1: MEMORY REGION BIT DEFINITIONS

Bit	Name	Description
31:29	Reserved	Always read ‘0’
28:16	OTP_MEM_START_ADDRESS_VAL	Address in OTP that this region of memory starts at. Note that this value should be between `0x100` and `0x1EF0`.
15:5	Reserved	Always read ‘0’
4:0	(OTP_MEM_TAG_VAL)	Memory tag associated with this region of memory. See Table 2 for memory tags.

A memory tag indicates the area of the PCI1xxxx the associated register writes are part of. There are 16 different memory tags given in Table 2.

TABLE 2: MEMORY TAGS

Memory Tag	Tag Name	Associated Registers
5’h00	Reserved	—
5’h01	USB_SS_TAG	USB SUBSYSTEM ADDRESS BASE
5’h02	ENET_SS_TAG	ENET SUBSYSTEM ADDRESS BAS
5’h03	UART_PERI_TAG	UART PERIPHERAL ADDRESS BASE
5’h04	SMBUS_PERI_TAG	SMBUS PERIPHERAL ADDRESS BASE
5’h05	SPI_PERI_TAG	SPI PERIPHERAL ADDRESS BASE
5’h06	GEN_PERI_TAG	GENERAL PERIPHERAL ADDRESS BASE
5’h07	SW_SS_MAIN_TAG	PCIE SWITCH MAIN ADDRESS BASE
5’h08	SW_SS_P0_TAG	PCIE SWITCH PORT 0 ADDRESS BASE
5’h09	SW_SS_P1_TAG	PCIE SWITCH PORT 1 ADDRESS BASE

TABLE 2: MEMORY TAGS (CONTINUED)

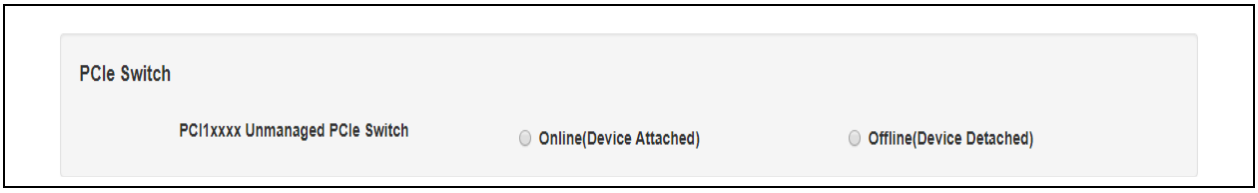
Memory Tag	Tag Name	Associated Registers
5'h0A	SW_SS_P2_TAG	PCIE SWITCH PORT 2 ADDRESS BASE
5'h0B	SW_SS_P3_TAG	PCIE SWITCH PORT 3 ADDRESS BASE
5'h0C	SW_SS_P4_TAG	PCIE SWITCH PORT 4 ADDRESS BASE
5'h0D	SYSREG_TAG	SYSTEM REGISTER ADDRESS BASE
5'h0E	PCIE_PHYA_TAG	PCIE PHY A ADDRESS BASE
5'h0F	PCIE_PHYB_TAG	PCIE PHY B ADDRESS BASE
5'h10	PCIE_PHYC_TAG	PCIE PHY C ADDRESS BASE
5'h11…5'h1F	Reserved	—

5.0 DUMPING OTP

5.1 Windows

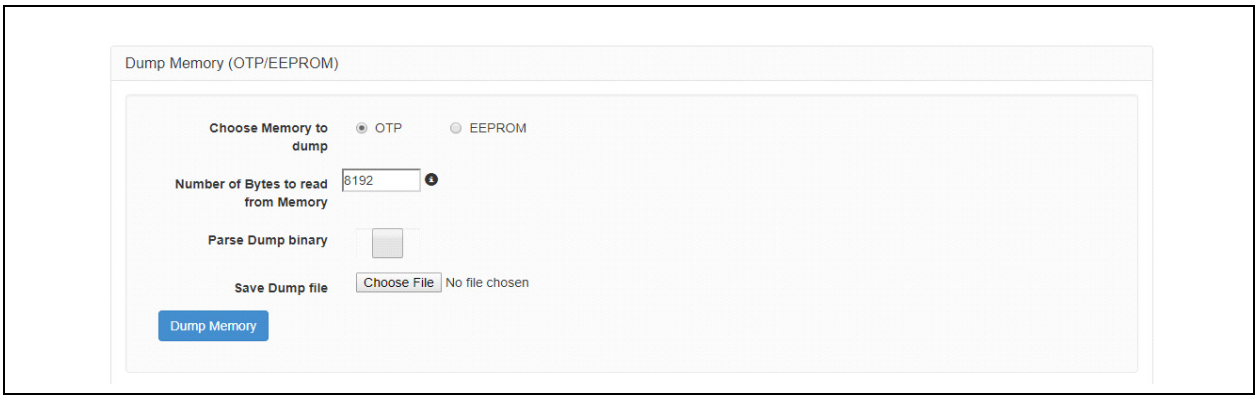
GP or Ethernet driver must be loaded. PCI1xxxx OTP can be dumped in Windows® using the MPLAB Connect® tool. On the same PC that the PCI1xxxx is connected to, open MPLAB Connect and navigate to PCIe® *Switch>Online (Device Attached)*. See Figure 4.

FIGURE 4: MPLAB CONNECT SELECTION



MPLAB Connect scans the system for any PCI1xxxx devices attached. After the scan is finished and the MPLAB Connect is connected to the desired PCI1xxxx device, expand Dump Memory (OTP/EEPROM). Select OTP, click **Choose File** to indicate a save location, and then click Dump Memory. See Figure 5.

FIGURE 5: DUMP MEMORY



5.2 Linux

To dump the OTP image, the GP driver must be loaded. This driver has been mainlined in kernel 6.1 and up. Depending on the version of the kernel being used, the OTP enumerates as a file in one of two different places.

In older kernels (kernels below 6.6), it shows under `/dev/PCI1xxxxOTP#` where # is a hexadecimal number corresponding to a specific PCI1xxxx DUT (if there is only one in the system # will be 0). See Figure 6.

FIGURE 6: OTP IN /DEV/

```
lab@lab-MS-7C56:~$ ls /dev/P*
/dev/PCI1xxxxE2P0  /dev/PCI1xxxxOTP0
```

In newer kernels (from kernel 6.6 on), it shows under `/sys/bus/nvmem/devices/pci1xxxx_otp#/nvmem` where # again is a hexadecimal number corresponding to a specific PCI1xxxx DUT. See Figure 7.

FIGURE 7: OTP IN /SYS/BUS/NVMEM/DEVICES

```
phxlab@phxlab-MS-7C56:~$ ls /sys/bus/nvmem/devices/
cmos_nvram0  pci1xxxx_eeprom0  pci1xxxx_otp0
```

There are two ways of dumping the OTP in Linux. The first is using hexdump. This is useful if a visual check is all that is required. Running `hexdump -C <PATH to OTP>` (note elevated privileges may be needed) outputs the contents of the OTP to the terminal. See Figure 8.

FIGURE 8: OTP HEX DUMP

```
lab@lab-MS-7C56:~$ sudo hexdump -C /dev/PCI1xxxxOTP0
00000000  a5 00 01 00 00 20 00 00  ff 1f 00 e8 00 00 00 00  |..... ..........|
00000010  00 00 00 0e 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  0d 00 00 01 0e 00 6b 01  0f 00 66 02 10 00 61 03  |......k...f...a.|
00000040  08 00 0c 04 09 00 d5 04  0a 00 76 05 0c 00 9f 05  |..........v.....|
00000050  01 00 be 05 03 00 2d 06  04 00 42 06 05 00 93 06  |......-...B.....|
00000060  06 00 a8 06 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000090  00 00 00 00 00 00 00 00  00 00 00 00 0d 00 50 1e  |..............P.|
000000a0  00 00 00 00 0d 00 00 1f  01 00 51 1f 06 00 a4 1f  |..........Q.....|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000100  80 00 19 24 00 01 0b f0  03 00 80 08 04 24 00 06  |...$.........$..|
00000110  0f 20 00 00 03 20 00 00  04 20 00 00 05 20 00 00  |. ... ... ... ..|
00000120  05 20 00 00 0d 20 00 00  80 44 04 24 00 04 00 20  |. ... ...D.$... |
00000130  00 00 00 20 00 00 0a 20  00 00 00 20 00 00 80 7c  |... ... ... ...||
00000140  04 24 00 04 01 20 00 00  06 20 00 00 03 20 00 00  |.$... ... ... ..|
00000150  01 20 00 00 80 b8 04 24  00 04 00 20 00 00 00 20  |. .....$... ... |
00000160  00 00 00 20 00 00 00 20  00 00 00 80 44 20 24 00  |... ... ....D $.|
00000170  01 f3 0f 00 00 80 18 20  24 00 01 04 00 00 c0 80  |....... $.......|
00000180  94 20 24 00 01 49 02 e0  01 80 98 20 24 00 01 10  |. $..I..... $...|
00000190  c4 01 00 80 c0 20 24 00  01 50 01 1e 00 80 7c 20  |..... $..P....| |
000001a0  24 00 01 50 01 03 00 80  c8 20 24 00 01 20 03 00  |$..P..... $.. ..|
000001b0  00 80 bc 20 24 00 01 02  00 00 00 80 78 20 24 00  |... $.......x $.|
000001c0  01 f3 0f 00 00 80 64 20  24 00 01 04 00 00 c0 80  |......d $.......|
000001d0  fc 20 24 00 01 49 02 e0  01 80 00 21 24 00 01 10  |. $..I.....!$...|
000001e0  c4 01 00 80 28 21 24 00  01 50 01 1e 00 80 e4 20  |....(!$..P..... |
000001f0  24 00 01 50 01 03 00 80  30 21 24 00 01 20 03 00  |$..P....0!$.. ..|
```

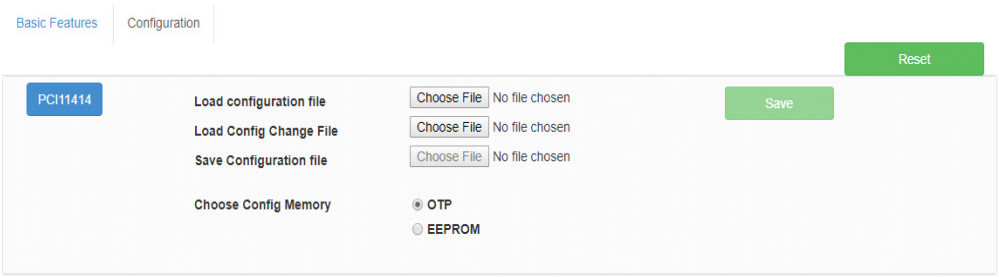
The second is using dd. This is useful if the OTP is needed to be saved as a `.bin` file. Running `dd if=<PATH TO OTP> of=<.BIN FILE> status=progress` (note that elevated privileges may be needed) saves the contents of OTP to a `.bin` file. See Figure 9.

FIGURE 9: DUMP OTP TO FILE

```
lab@lab-MS-7C56:~$ sudo dd if=OTP.bin of=/dev/PCI1xxxxOTP0 status=progress
16+0 records in
16+0 records out
8192 bytes (8.2 kB, 8.0 KiB) copied, 9.2965e-05 s, 88.1 MB/s
```

6.0 GENERATING OTP IMAGE

FIGURE 10: GENERATE OTP



A new OTP image can be generated in MPLAB Connect. Since OTPs are not shipped from the factory blank, users must use the pre-existing image as a base as shown in Figure 10 and complete the following steps:

1. Within MPLAB Connect, navigate to *PCIe Switch>Offline(Device Detached)>Configuration* and select the desired SKU.
2. Click on the **Choose File** button next to Load configuration. This will load the OTP previously dumped and preset those options in the configuration tabs.
3. Make the desired changes.
4. Upload the configuration file using the **Choose File** button next to Save Configuration file and click on **Save**. This provides a file that combines the original OTP with the changes that were implemented.

7.0 PROGRAMMING OTP IMAGE

7.1 Windows

In Windows, the OTP should be programmed using MPLAB Connect. Navigate to PCIe *Switch>Online (Device Attached)*. Upon entering this page, MPLAB Connect provides a popup indicating whether it detected a PCI1xxxx device or not. If the device is not found, verify it is powered on and connected to the PC. See Figure 11 and Figure 12.

FIGURE 11: DEVICE NOT FOUND

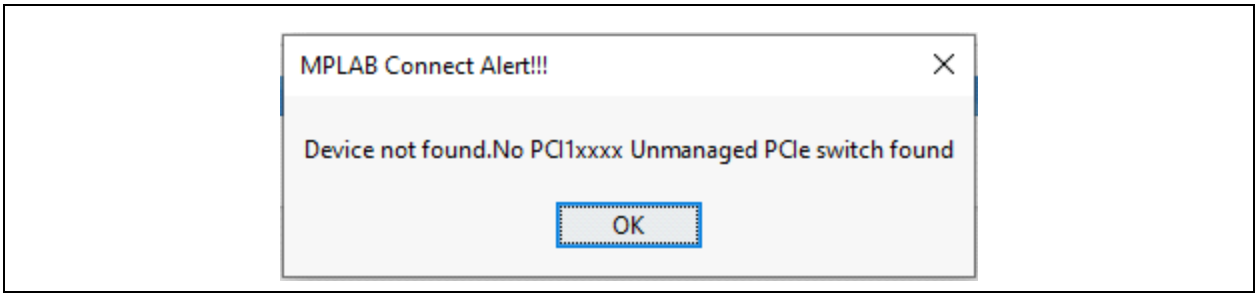
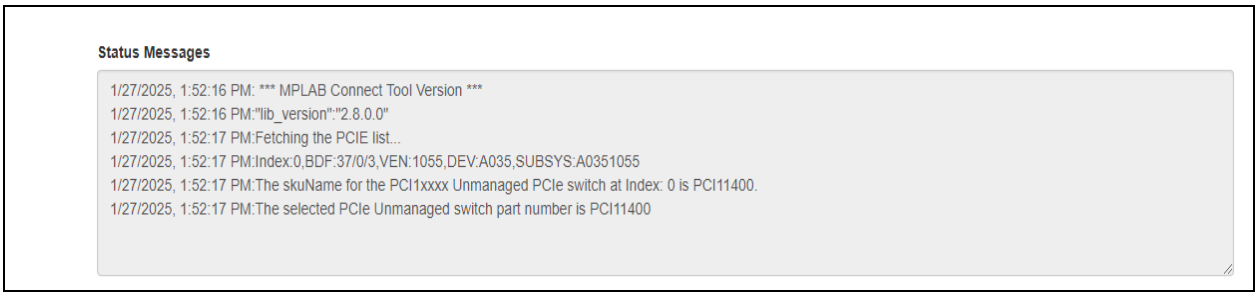
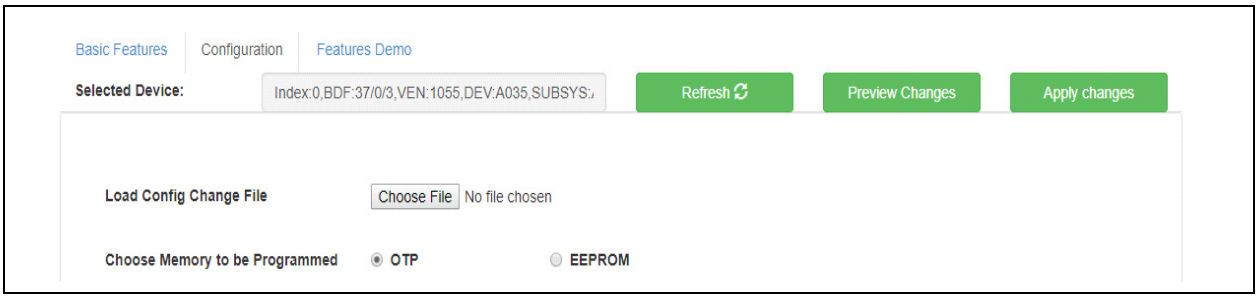


FIGURE 12: DEVICE DETECTED



From there, navigate to the **Configuration** tab. Select **Choose File** under Load Config Change File and navigate to the previously generated OTP `.bin` file. Select OTP under Choose Memory to be Programmed and then click on **Apply changes** to program the OTP. See Figure 13.

FIGURE 13: LOAD OTP FILE



7.2 Linux

In Linux, the OTP can be programmed using the `dd` command. As mentioned earlier, the OTP enumerates either in `/dev/PCI1xxxxOTP#` or `/sys/bus/nvmem/devices/PCI1xxxx_OTP#/nvmem`, where # is a hexadecimal number.

Given the OTP location, the OTP itself can then be programmed with `dd if=<PATCHED OTP FILE> of=<OTP LOCATION> status=progress`. Note that the command may need to be run with elevated privileges. See Figure 14.

FIGURE 14: PROGRAMMING OTP FROM FILE

```
lab@lab-MS-7C56:~$ sudo dd if=/dev/PCI1xxxxOTP0 of=OTP.bin status=progress
16+0 records in
16+0 records out
8192 bytes (8.2 kB, 8.0 KiB) copied, 0.0701408 s, 117 kB/s
```

APPENDIX A: REVISION HISTORY

TABLE A-1: REVISION HISTORY

Revision Level & Date	Section/Figure/Entry	Correction
DS00005898A (04-14-25)	Initial release	

NOTES:

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-1032-5

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.