RENESAS DIGITAL MULTIPHASE

GENERATION 2 PROGRAMMING GUIDE

JULY 2022

BIG IDEAS
FOR EVERY SPACE

OVERVIEW

- **This guide specifies the algorithm for programming Renesas generation 2 digital multiphase products via PMBus communication.**

- **Unless noted otherwise, timing and voltage requirements are outlined in the PMBus specification version 1.3.**

- **The following sections are shown in this guide:**
  - Section 1: Device packages and pinouts
  - Section 2: Programming devices
  - Section 3: HEX file CRC information
  - Appendix: DMA Command Format Reference

Supported Devices (ISL Part Numbers)

Device Name	Package	IC_DEVICE_ID Byte ID[1]
ISL68220	32 LD QFN	0x63
ISL68221	40 LD QFN	0x62
ISL68222	40 LD QFN	0x61
ISL68223	40 LD QFN	0x53
ISL68224	52 LD QFN	0x52
ISL68225	52 LD QFN	0x51
ISL68226	60 LD QFN	0x50
ISL68227	60 LD QFN	0x4F
ISL68229	68 LD QFN	0x4E
ISL68233	40 LD QFN	0x6B
ISL68236	52 LD QFN	0x4D
ISL68239	68 LD QFN	0x4B
ISL69222	32 LD QFN	0x3E
ISL69223	40 LD QFN	0x3D
ISL69224	40 LD QFN	0x3C
ISL69225	40 LD QFN	0x3B
ISL69227	52 LD QFN	0x3A
ISL69228	52 LD QFN	0x39

Device Name	Package	IC_DEVICE_ID Byte ID[1]
ISL69234	40 LD QFN	0x43
ISL69236	52 LD QFN	0x42
ISL69237	52 LD QFN	0x66
ISL69239	60 LD QFN	0x41
ISL69242	32 LD QFN	0x58
ISL69243	32 LD QFN	0x59
ISL69247	52 LD QFN	0x48
ISL69248	60 LD QFN	0x47
ISL69249	68 LD QFN	0x6D
ISL69254	40 LD QFN	0x67
ISL69255	52 LD QFN	0x38
ISL69256	52 LD QFN	0x37
ISL69259	40 LD QFN	0x46
ISL69260	40 LD QFN	0x6E
ISL69268	60 LD QFN	0x3F
ISL69269	68 LD QFN	0x55

This programming guide supports the devices shown in the Device Table above.

Supported Devices (RAA Part Numbers)

Device Name	Package	IC_DEVICE_ID Byte ID[1]
RAA228000	32 LD QFN	0x64
RAA228004	52 LD QFN	0x65
RAA228006	60 LD QFN	0x6C
RAA229001	48 LD QFN	0x69
RAA229004	48 LD QFN	0x6A
RAA229022	32 LD QFN	0x6F
RAA229126	48 LD QFN	0x7E

This programming guide supports the devices shown in the Device Table above.

Prerequisites and Conventions

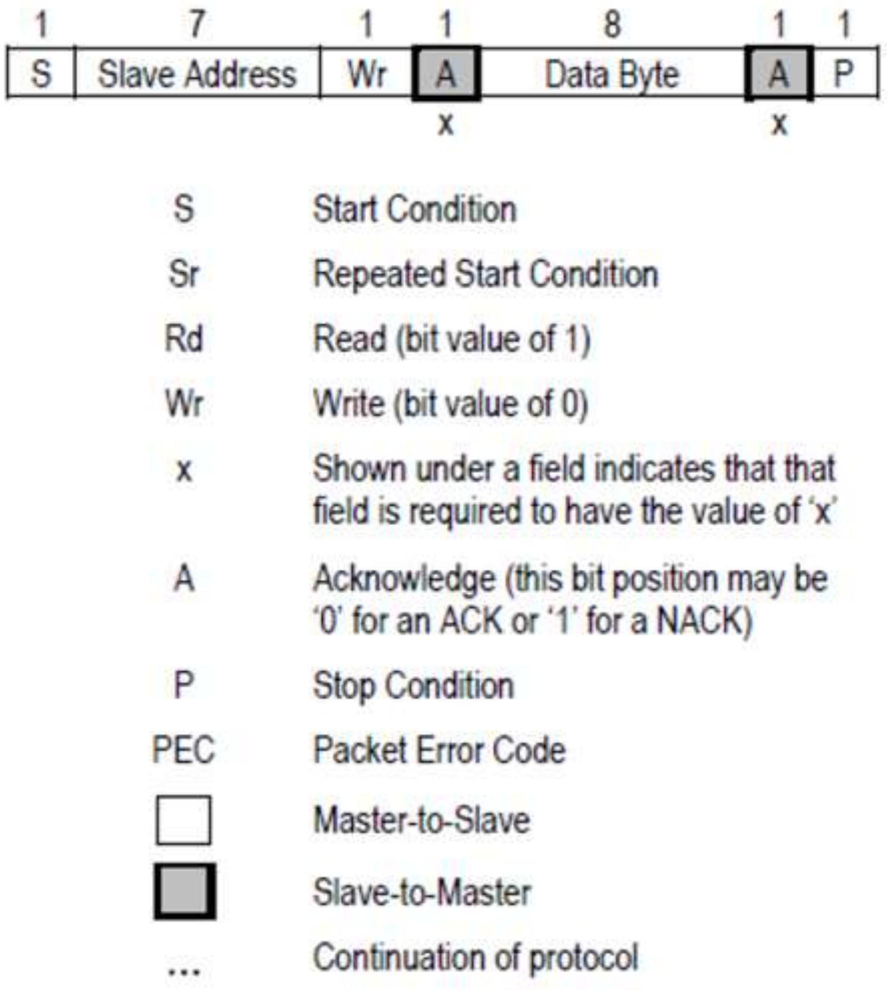
- HEX configuration files must be generated using PowerNavigator.
- In this guide, address 0x60 (7-bit format) is used in all examples.
- Data on the bus may be reversed. Follow examples for correct byte order.

Minimum Pin and Component Requirements

The following pins must be connected when programming a device:

- PMSCL and PMSDA (I^2C clock and data pins). These are open drain and must be pulled to 3.3V via a resistor (1kΩ maximum).
- VCC, provided with an external 3.3V supply. A 1uF decoupling capacitor from this pin to ground is also needed.
- VCCS must be decoupled with 4.7µF or greater MLCC (X5R or better).
- Ground pin must be connected to ground.
- ADDRESS pin must have an address set resistor to ground. Connect directly to ground for address 0x60.
- Other pins may be floated.

PMBus Communication Key



1	7	1	1	8	1	1
S	Slave Address	Wr	A	Data Byte	A	P
			x		x	

S	Start Condition
Sr	Repeated Start Condition
Rd	Read (bit value of 1)
Wr	Write (bit value of 0)
x	Shown under a field indicates that that field is required to have the value of 'x'
A	Acknowledge (this bit position may be '0' for an ACK or '1' for a NACK)
P	Stop Condition
PEC	Packet Error Code
	Master-to-Slave
	Slave-to-Master
...	Continuation of protocol

Note: See PMBus/SMBus spec for additional details and timing requirements.

Direct Memory Access (DMA) Command Codes

Actual device programming is completed through 3 command codes:

- **DMA Address (Command Code 0xC7):** Used to set the register address to use with other DMA commands.
- **DMA Data (Command Code 0xC5):** Used to read from or write to the register selected by the DMA Address command.
- **DMA Sequential (Command Code 0xC6):** Used to read from or write to the register selected by the DMA Address command, then automatically increment the register address by 1.

Important Notes

- Once the first write from the HEX file to the part has occurred, the part is in programming mode. Regardless of whether programming completes or is suspended, controller VCC must be cycled to clear this mode and put the part back in normal processing mode.

- The command that causes the OTP to burn is the last line of the HEX file. Device programming can be aborted at any point before this, and no contents will be burned to OTP.

- The last byte of each line in the HEX file is a CRC8 check for that line in the file only. It does not relate to any hardware value and can be ignored during programming.

SECTION 1: DEVICE PACKAGES AND PINOUTS

Important Notes

2nd Generation Digital Multiphase products span 6 QFN pkgs and 14 distinct pinouts.

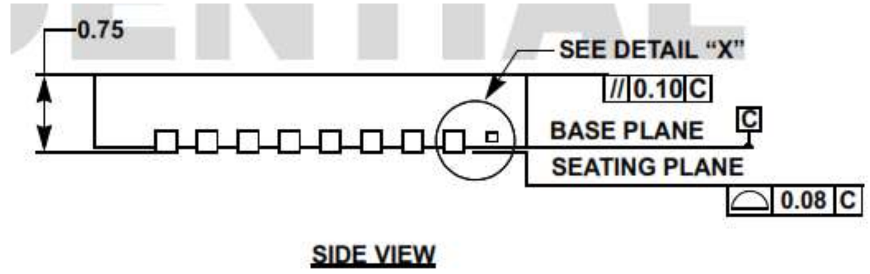
- 4x4
  - 2 pinouts
- 5x5
  - 3 pinouts
- 6x6 (48 LD)
  - 2 pinouts
- 6x6 (52 LD)
  - 2 pinouts
- 7x7
  - 4 pinouts
- 8x8
  - 1 pinout

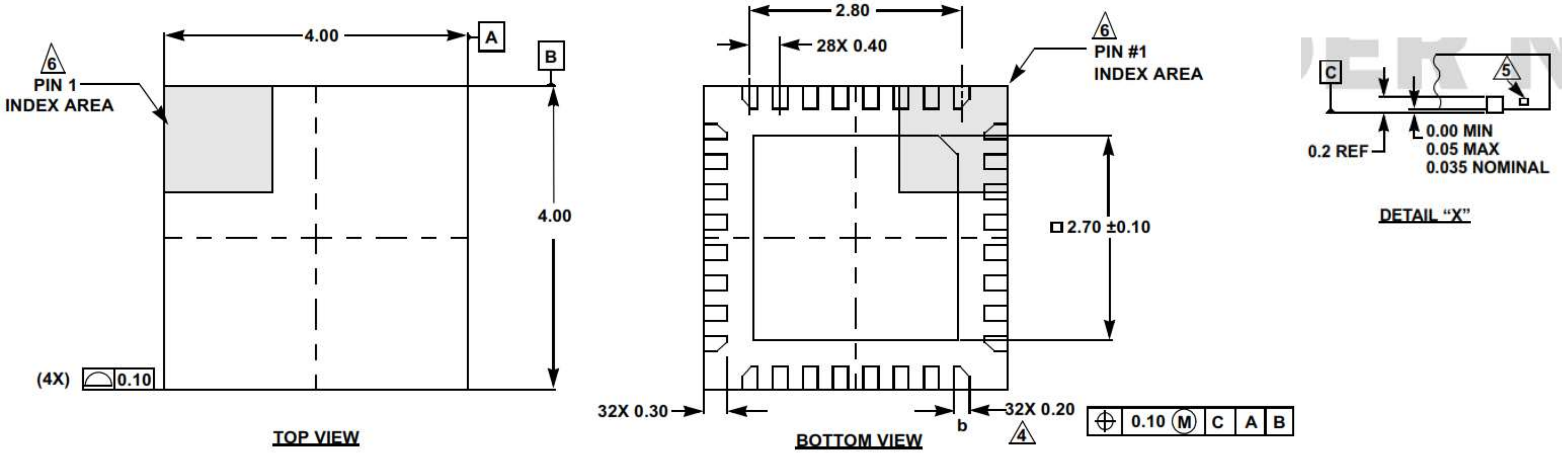
The slides in this section will provide information on each package and pinout.

4x4 Package Characteristics

4x4, 32L QFN package

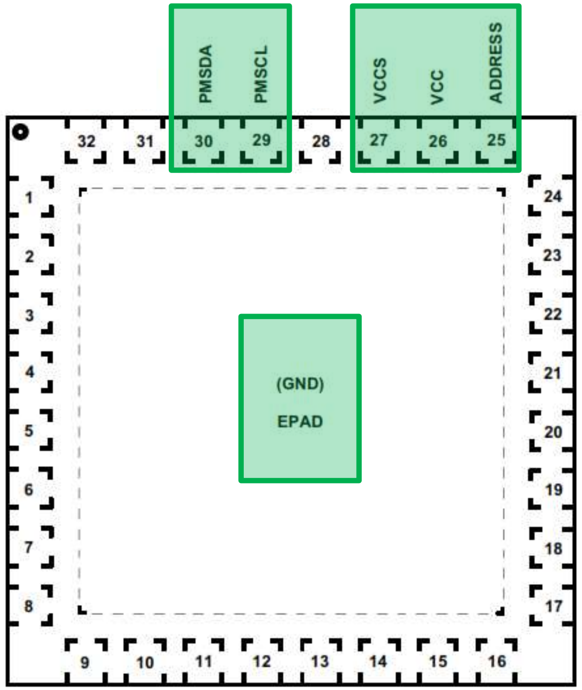
L32.4x4D
32 LEAD THIN QUAD FLAT NO-LEAD PLASTIC PACKAGE
Rev 2, 10/16



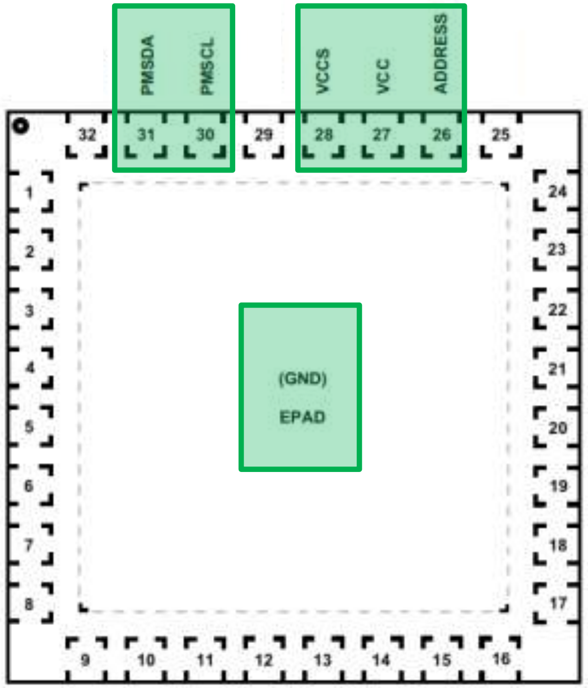


4x4 Pinouts and Devices

4x4, 32L QFN package



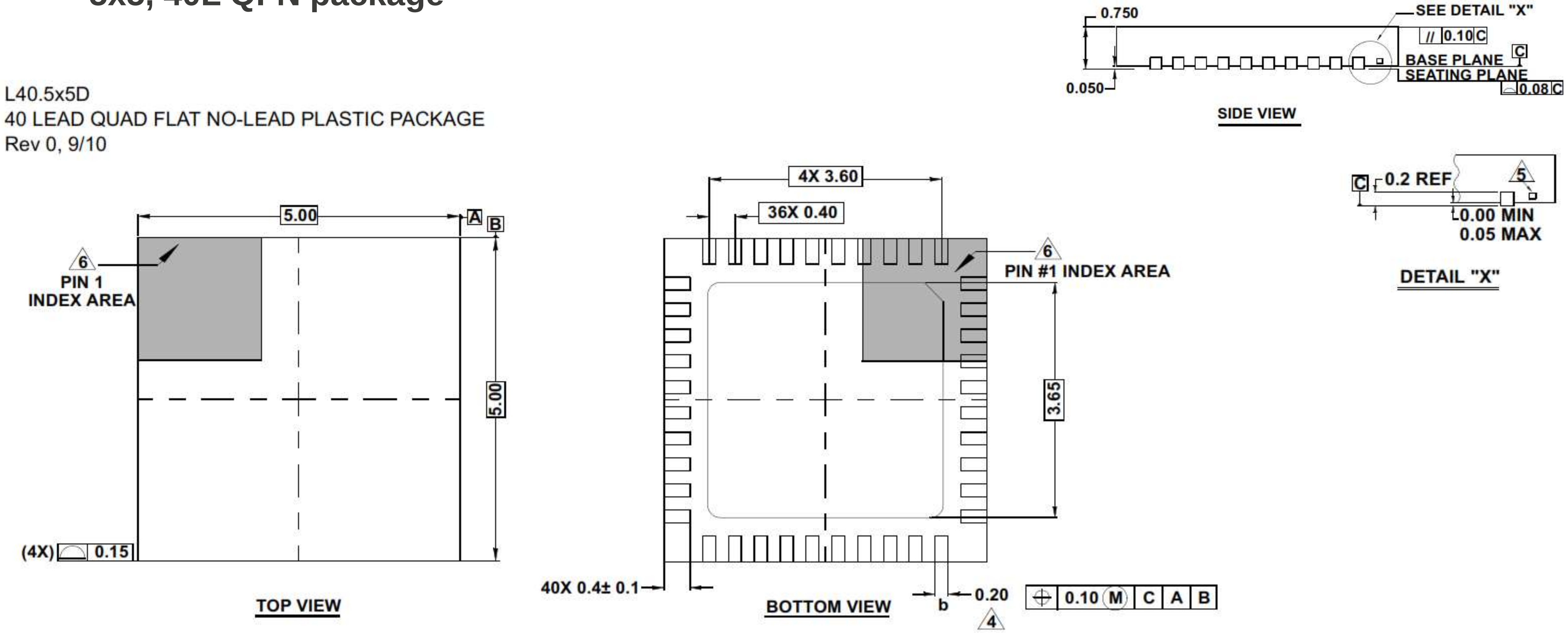
DEVICE TYPE
ISL69243



DEVICE TYPE
ISL68220
ISL69222
ISL69242
RAA228000
RAA229022

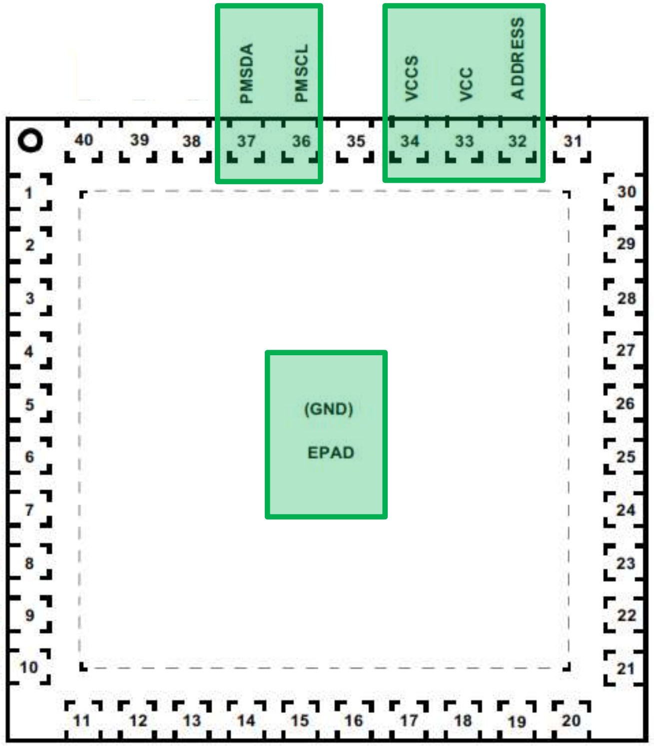
5x5 Package Characteristics

5x5, 40L QFN package

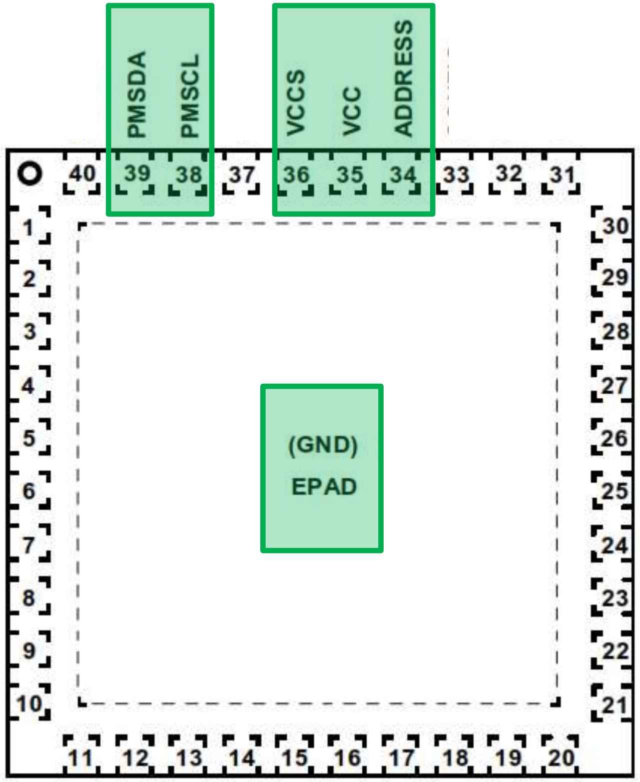


5x5 Pinouts and Devices

5x5, 40L QFN package



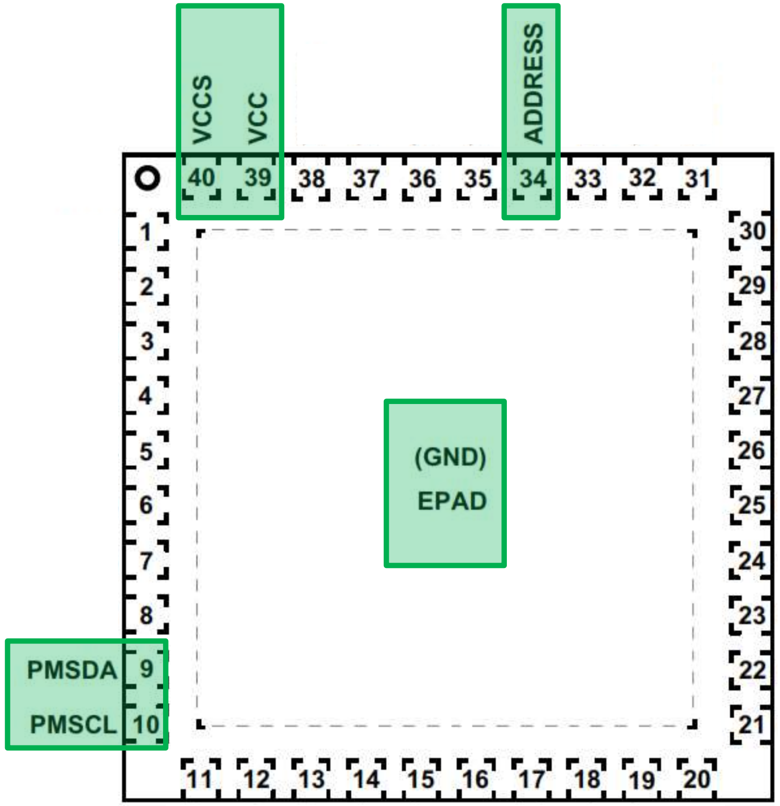
DEVICE TYPE
ISL68222
ISL68223
ISL68233
ISL69224
ISL69225
ISL69234



DEVICE TYPE
ISL68221
ISL69223

5x5 Pinouts and Devices

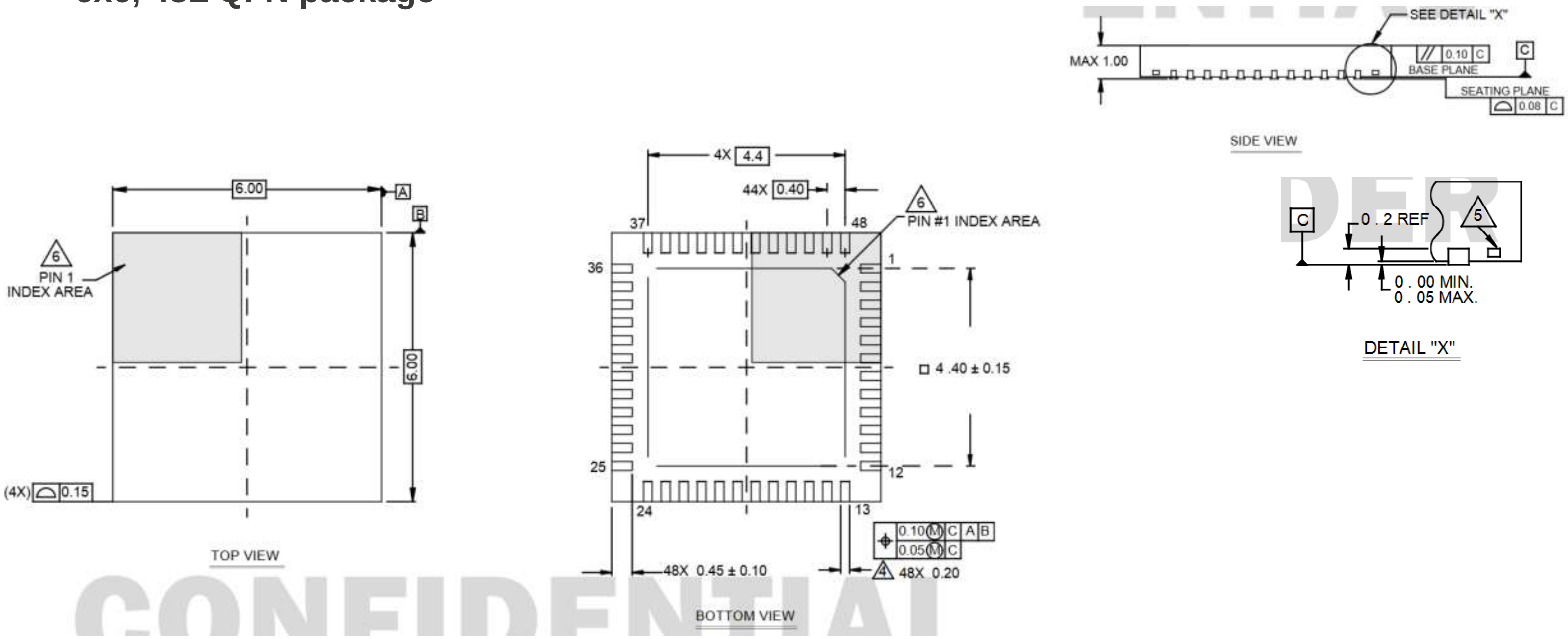
5x5, 40L QFN package



DEVICE TYPE
ISL69254
ISL69259
ISL69260

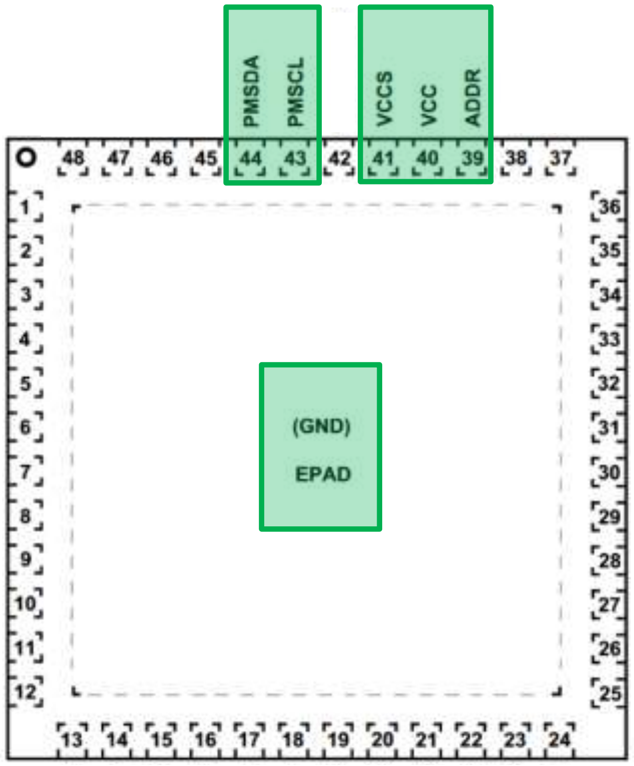
6x6 (48 LD) Package Characteristics

6x6, 48L QFN package

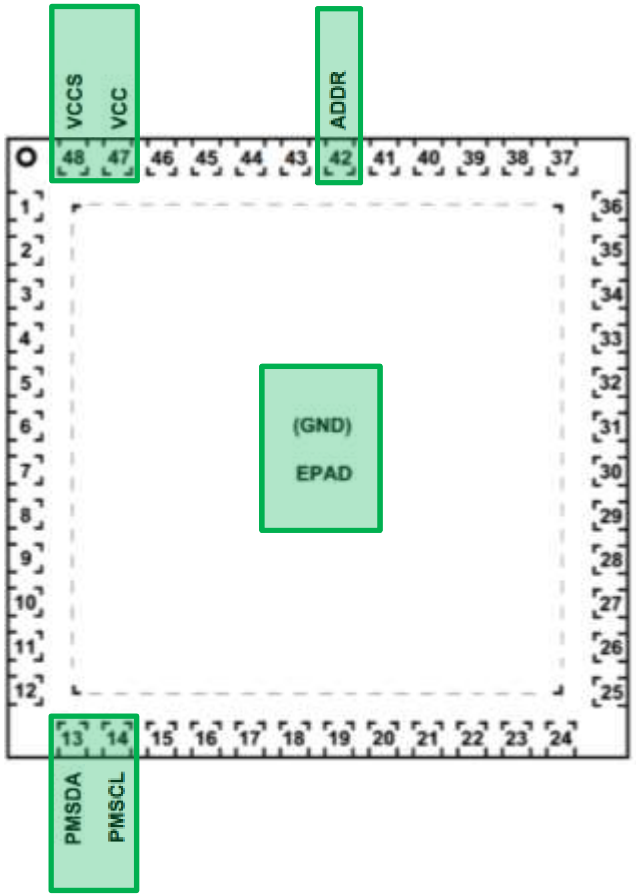


6x6 (48 LD) Pinouts and Devices

6x6, 48L QFN package



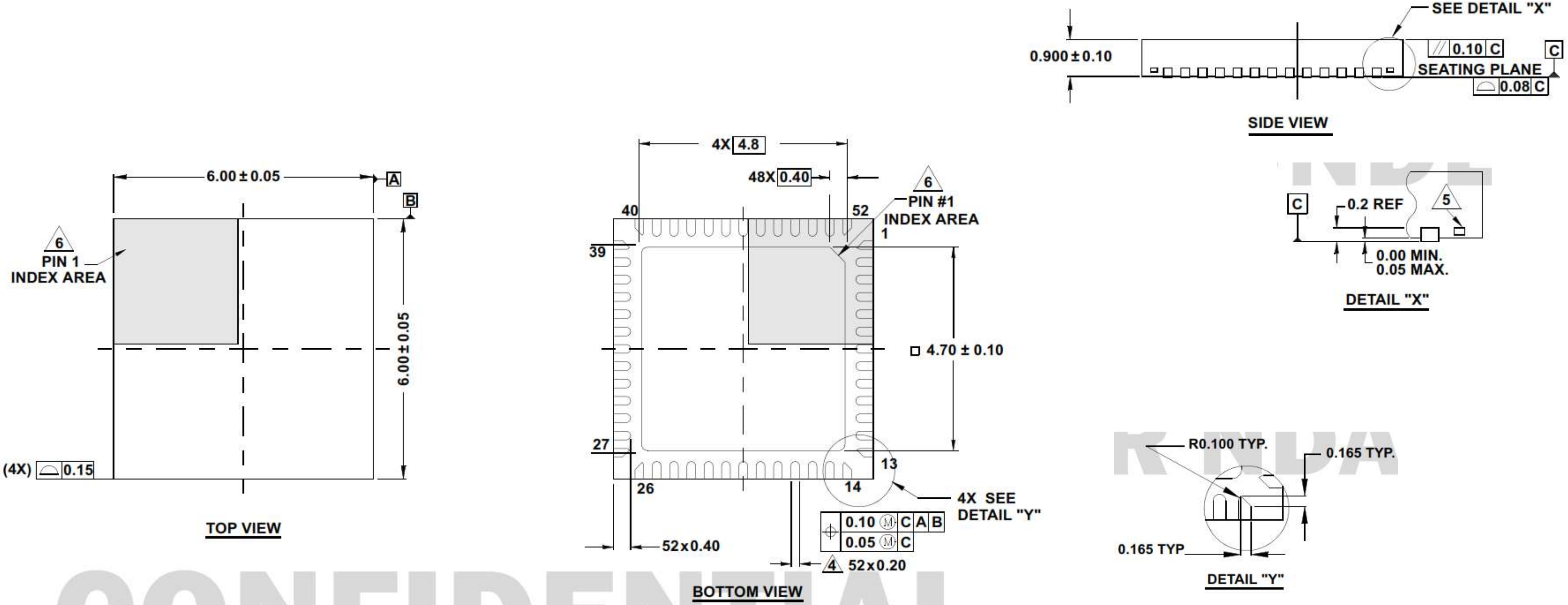
DEVICE TYPE
RAA229001
RAA229004



DEVICE TYPE
RAA229126

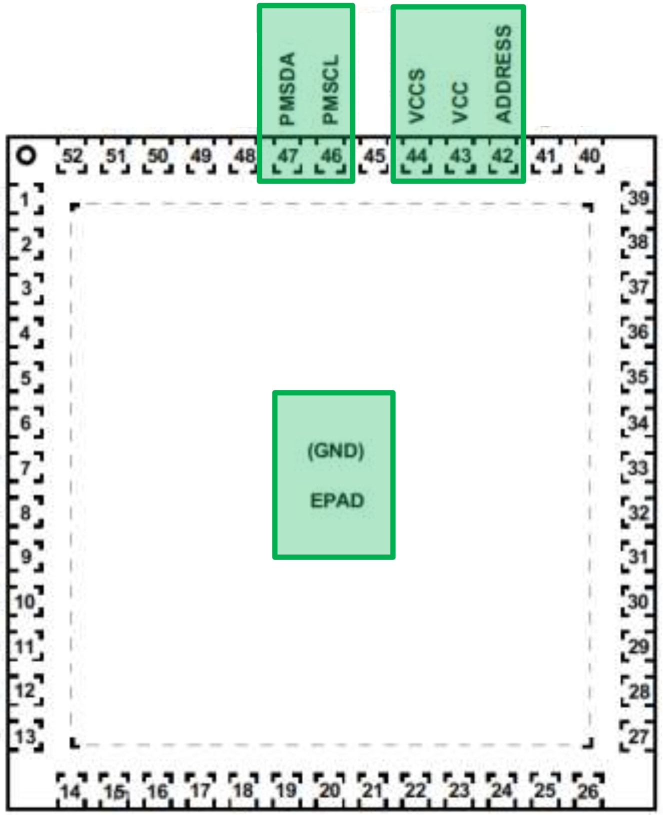
6x6 (52 LD) Package Characteristics

6x6, 52L QFN package

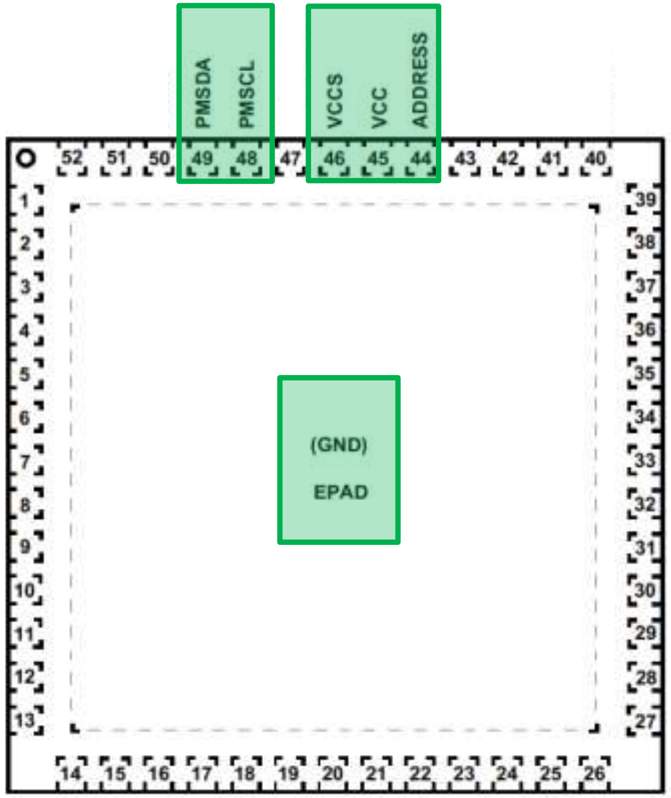


6x6 (52 LD) Pinouts and Devices

6x6, 52L QFN package



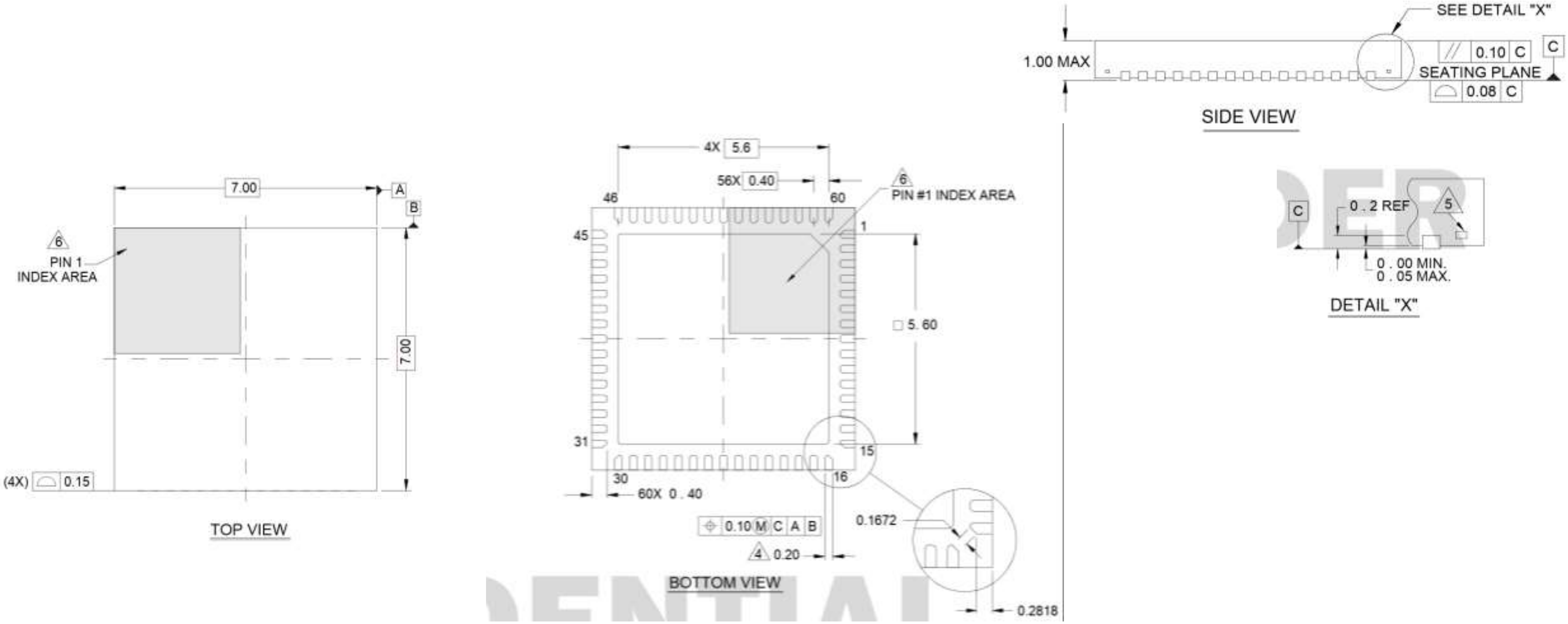
DEVICE TYPE
ISL68225
ISL68236
ISL69236
ISL69238
ISL69247
ISL69255
ISL69256
RAA228004



DEVICE TYPE
ISL68224
ISL69227
ISL69228
ISL69237

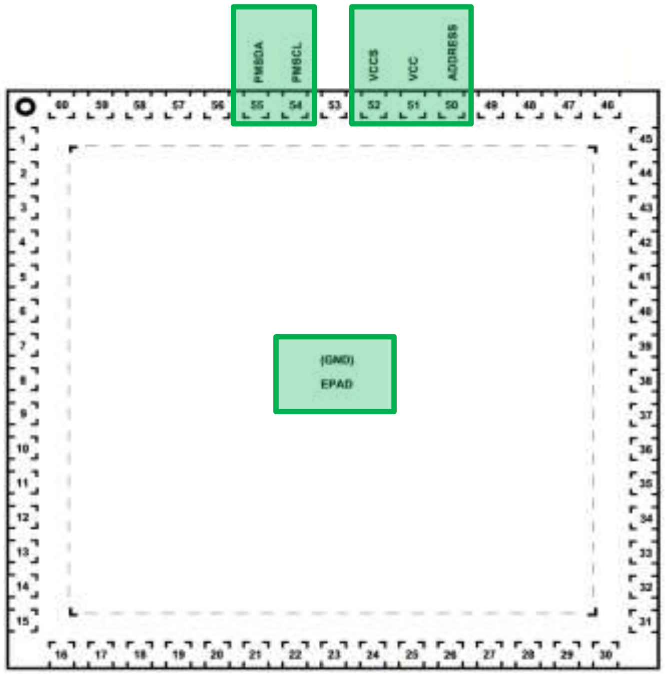
7x7 Package Characteristics

7x7, 60L QFN package

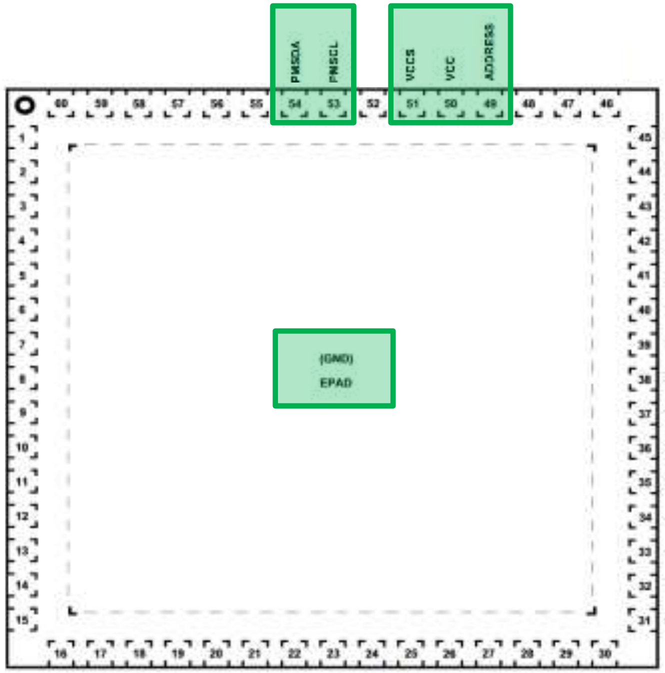


7x7 Pinouts and Devices

7x7, 60L QFN package



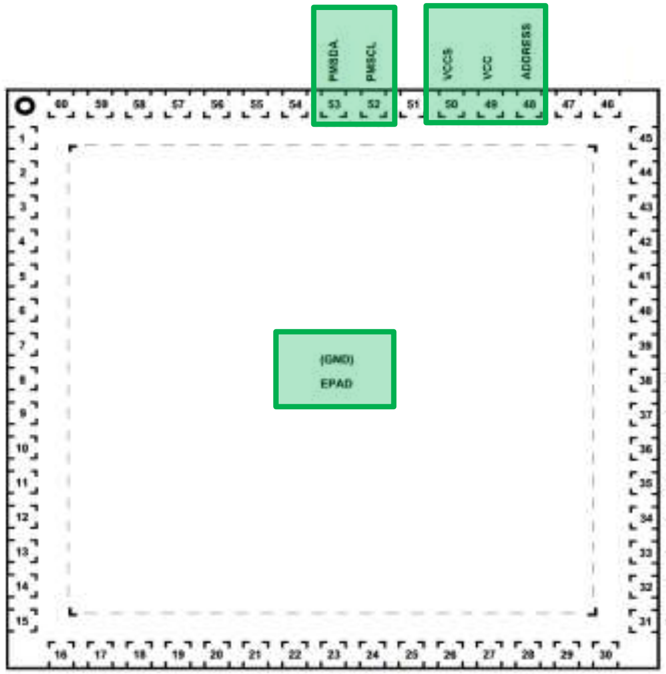
DEVICE TYPE
ISL68226
ISL69239
ISL69268



DEVICE TYPE
ISL68227
RAA228006

7x7 Pinouts and Devices

7x7, 60L QFN package

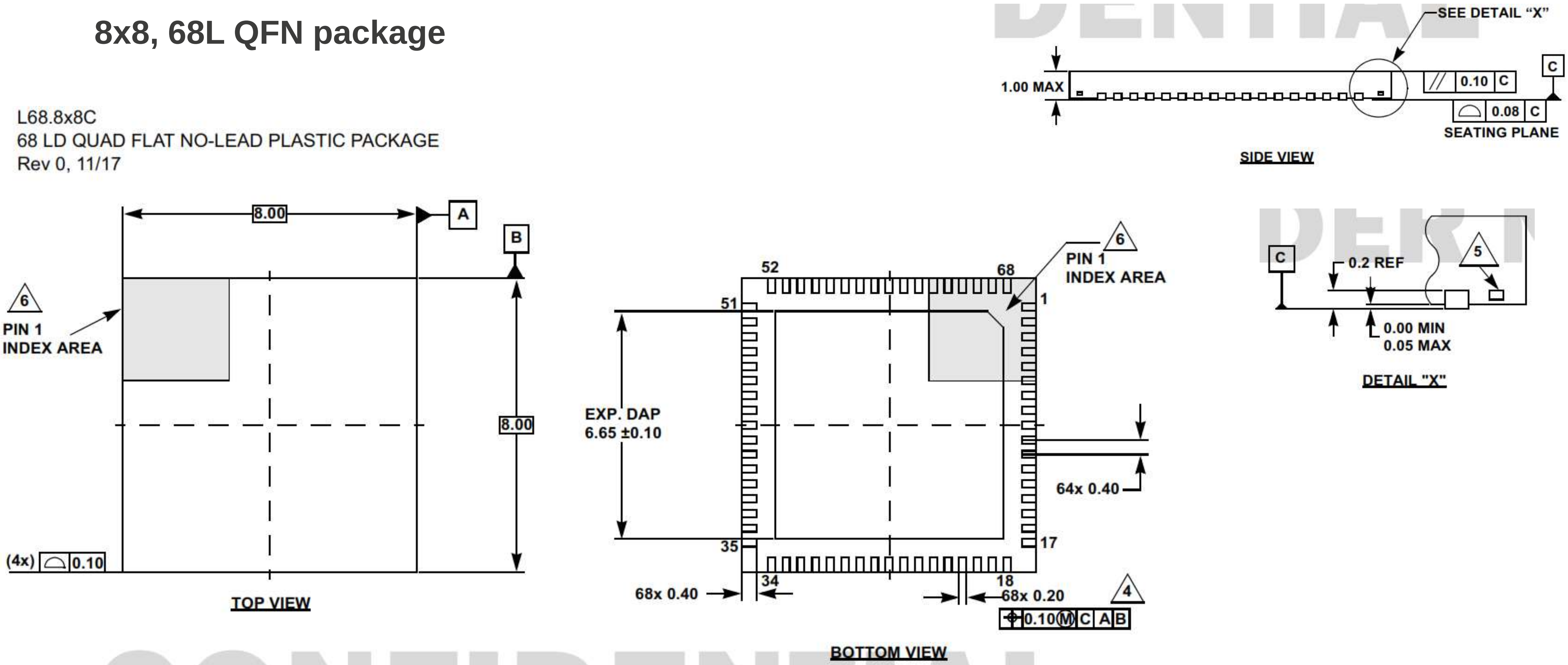


DEVICE TYPE
ISL69248

8x8 Package Characteristics

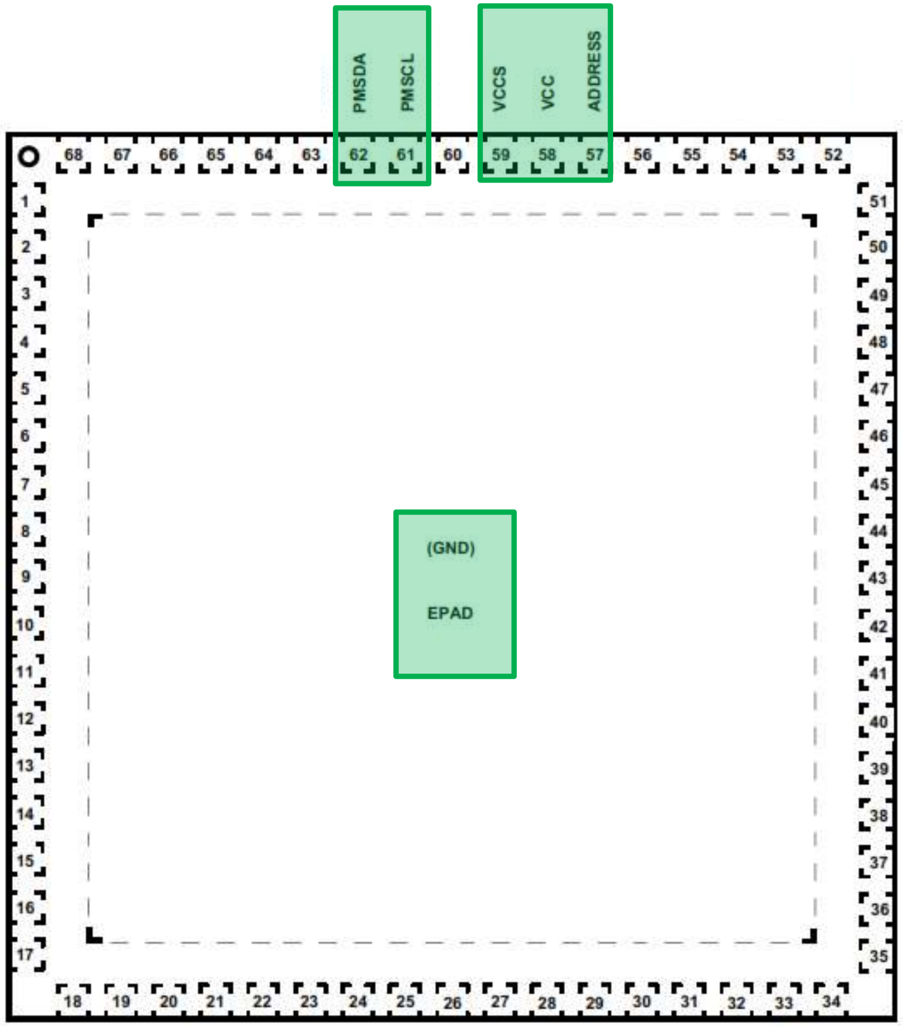
8x8, 68L QFN package

L68.8x8C
68 LD QUAD FLAT NO-LEAD PLASTIC PACKAGE
Rev 0, 11/17



8x8 Pinouts and Devices

8x8, 68L QFN package



DEVICE TYPE
ISL68229
ISL68239
ISL69249
ISL69269

SECTION 2: PROGRAMMING DEVICES

PROGRAMMING ALGORITHM OVERVIEW

1. **Determine number of NVM slots available.**

2. **Verify device and file versions.**
   a. Read and parse header data from HEX file. Go to the Step 2a section for more information.
   b. Read IC_DEVICE_ID from device. Verify the value matches the Device Table.
   c. Read IC_DEVICE_REV from device. Verify the value matches the HEX file.

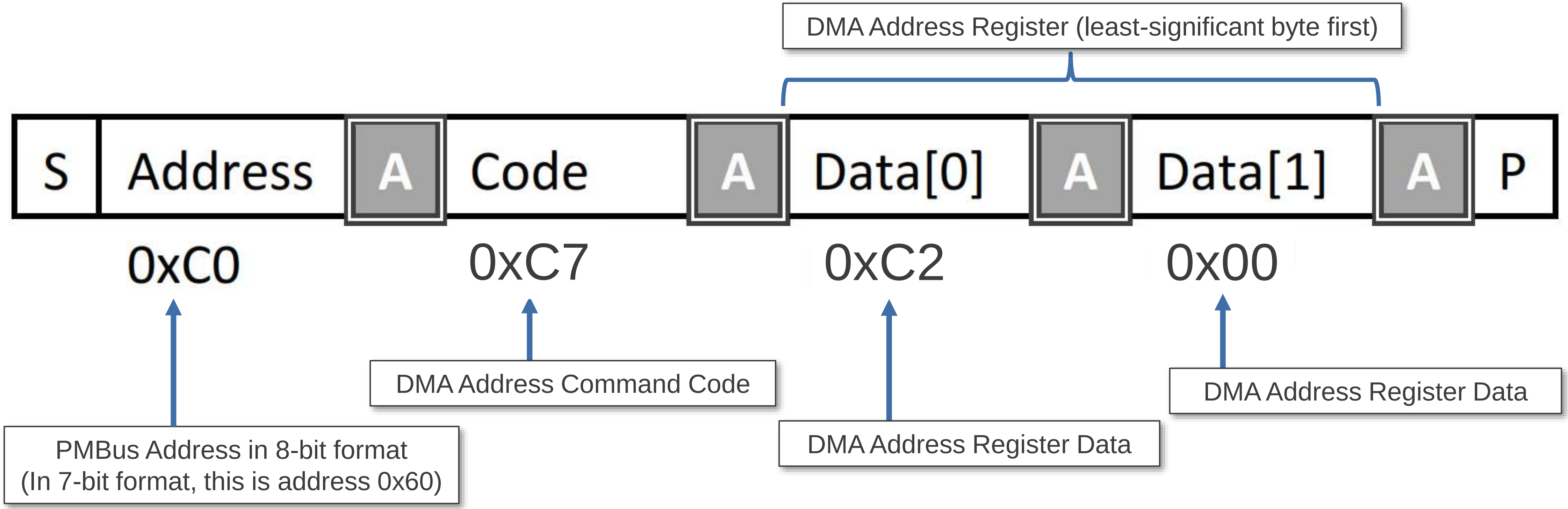
3. **Read and parse one line from HEX file. Write to device.**
   - This step must be repeated for all configuration lines in the HEX file.

4. **Verify programming success.**
   a. Poll PROGRAMMER_STATUS register until programming is complete.
   b. Read the BANK_STATUS registers to confirm all configurations were programmed successfully.

5. **Cycle VCC and verify CRC values.**

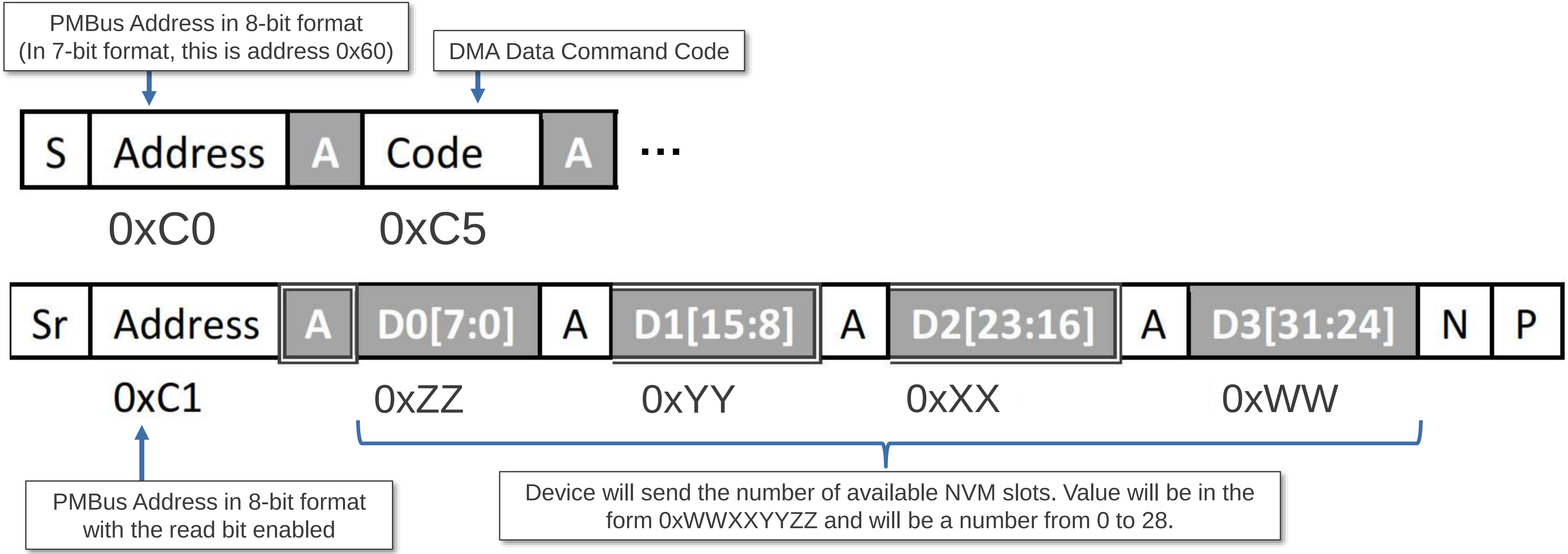
Step 1a – Write to DMA Address Register

To read the number of available NVM slots, first set the DMA address, as shown below.



Step 1b – Read DMA Data Register

Next, Read the content of the register pointed to in step 1.



Step 2a – Parse Header in HEX File

- The first 5 lines in the HEX file (starting with 0x49) are part of the HEX file header and should not be written to the device.

- The HEX header contains IC_DEVICE_ID, IC_DEVICE_REV and HEX_VERSION information.
  - IC_DEVICE_ID in HEX file must match IC_DEVICE_ID read back from device (see step 2b).

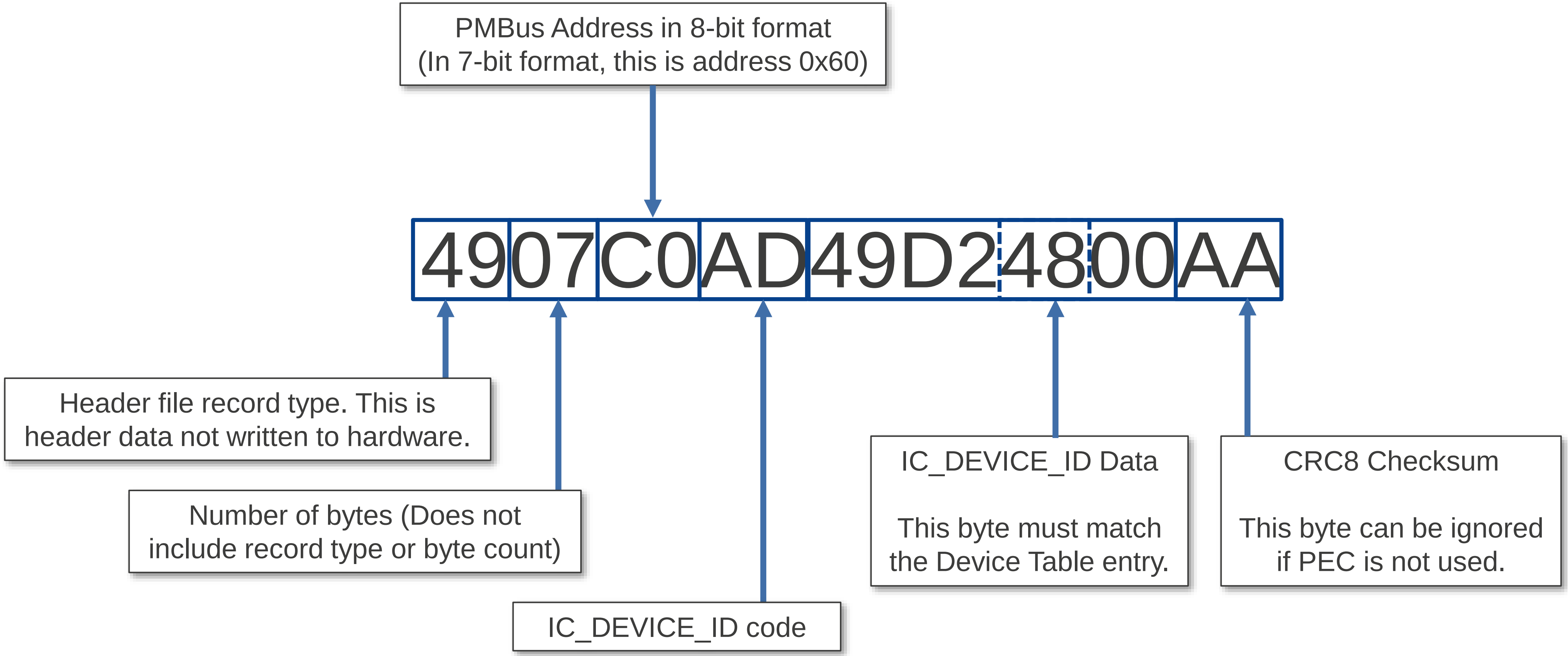
Step 2a – Example HEX File

```
1 4907C0AD49D24800AA
2 4907C0AE0200000017
3 4907C00000000200B0
4 490AC001352E342E31333573
5 490BC00200000168C930F47B6F
6 0005C0E6020033
7 0005C0C7000724
8 0007C0C6E1000000FC
9 0007C0C6001C0000D6
10 0007C0C6840400004C

...

640 0005C0C70C07D8
641 0007C0C60100000098
642 0005C0C721018D
643 0007C0C6AD1D0000BC
644 0005C0C7DB001C
645 0007C0C600000008E
646 0005C0C7DD0062
647 0007C0C600000008E
648 0005C0E6060067
649
```

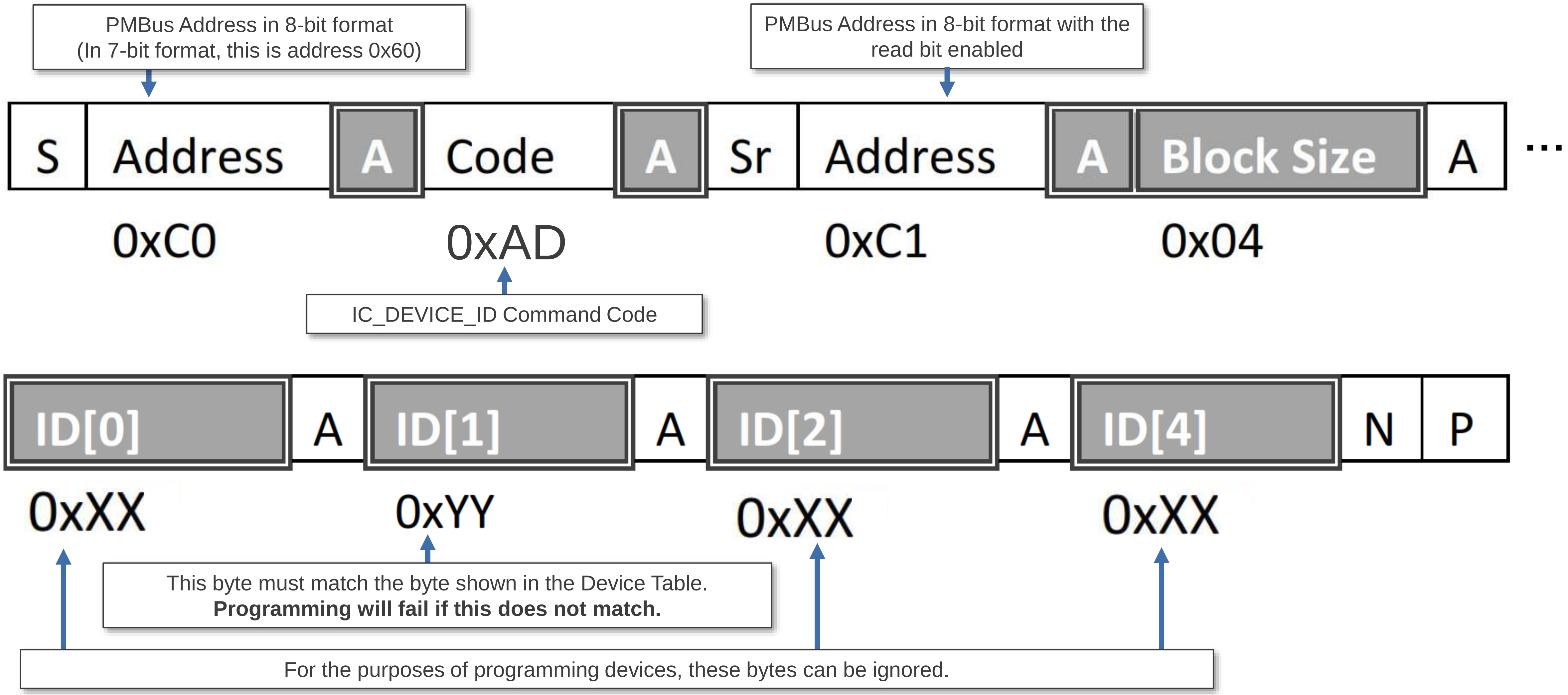
Parse first two lines and verify the HEX file was generated for the device being programmed (see steps 2b and 2c).
- Line 1 = IC_DEVICE_ID
- Line 2 = IC_DEVICE_REV
- Line 3 = HEX_VERSION

Value contained in file	Record Type	Command Code
IC_DEVICE_ID	0x49	0xAD
IC_DEVICE_REV	0x49	0xAE
HEX_VERSION	0x49	0x00

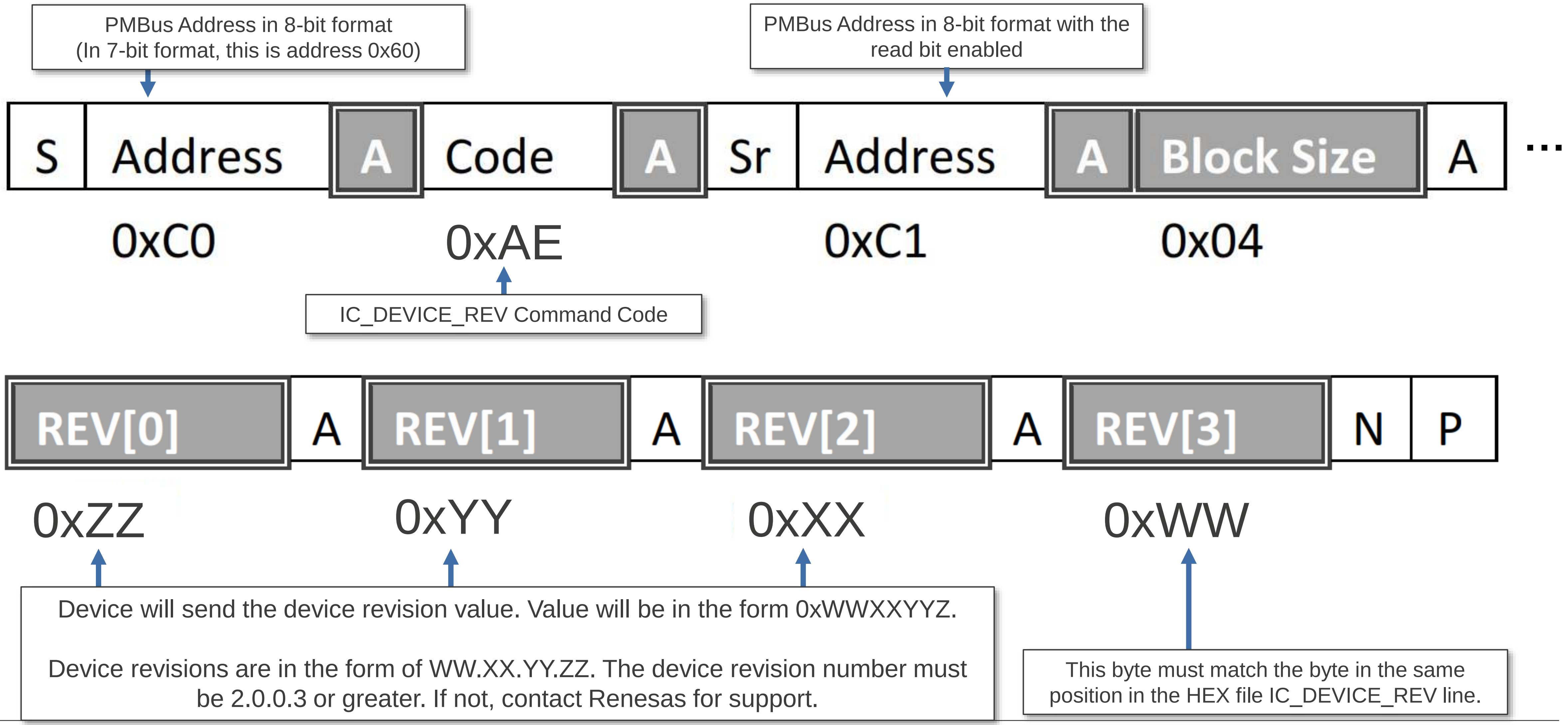
Step 2a – Example HEX File Header Decode



Step 2b – Read IC_DEVICE_ID from Device



Step 2c – Read IC_DEVICE_REV from Device



Step 2d – Compare data to HEX file

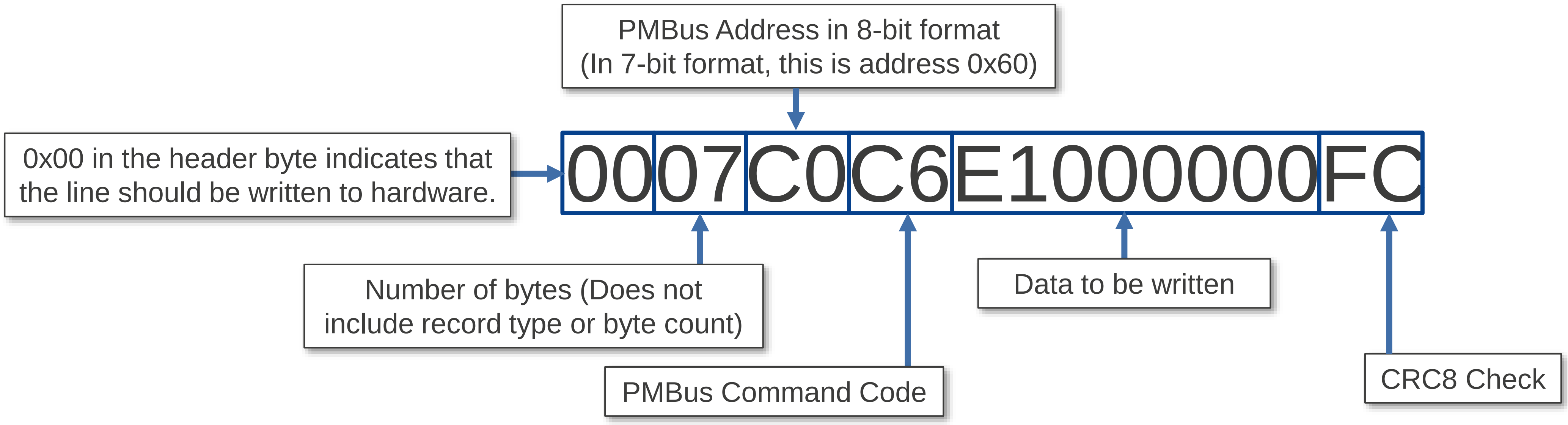
- If the IC_DEVICE_ID and IC_DEVICE_REV bytes in the HEX file match the bytes read back from the device, proceed to step 3.

- If it does not match, the HEX file was generated for a different device and device programming should be halted.

Step 3 – Parse HEX File and Write to Hardware

Parse remaining lines from HEX file (all lines not starting with 0x49), and write to device.

Example HEX File Line:



Step 3 – Example HEX File

```
4907C0AD49D24800AA
4907C0AE0200000017
4907C000000000200B0
490AC001352E342E31333573
490BC00200000168C930F47B6F
0005C0E6020033
0005C0C7000724
0007C0C6E1000000FC
0007C0C6001C0000D6
0007C0C6840400004C

...

0005C0C70C07D8
0007C0C60100000098
0005C0C721018D
0007C0C6AD1D0000BC
0005C0C7DB001C
0007C0C600000008E
0005C0C7DD0062
0007C0C600000008E
0005C0E6060067
```

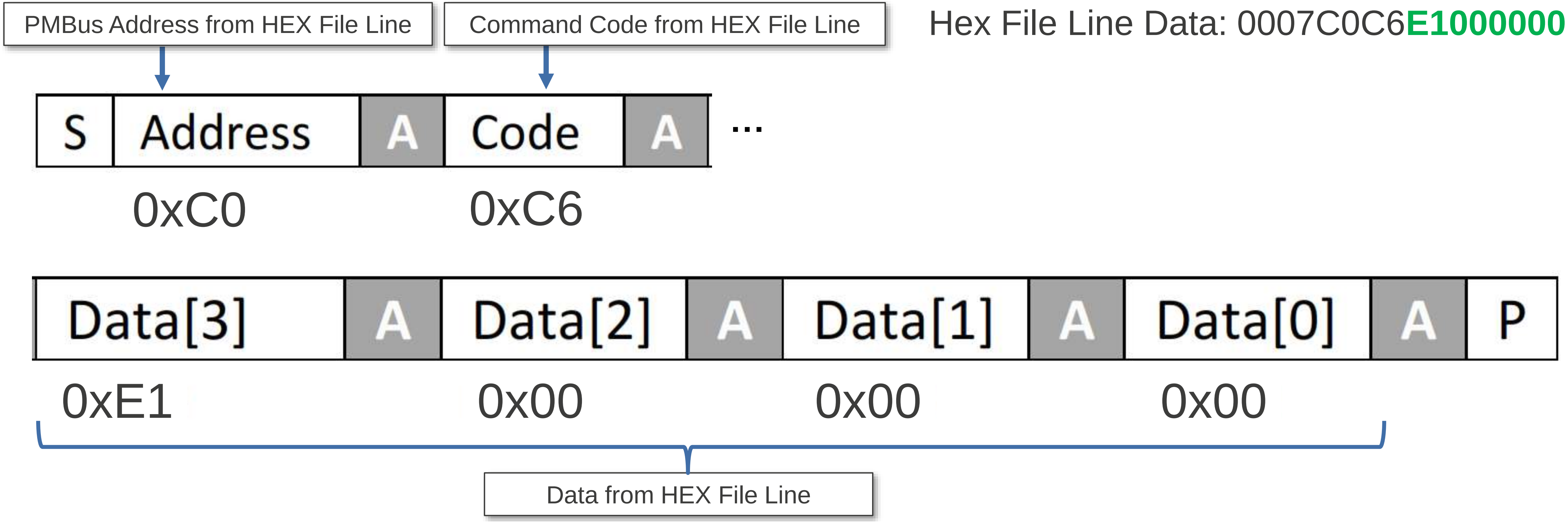
Step 3: Parse and write to device all lines in HEX file that do not start with 0x49

When last line has been written to device, Step 3 is complete.

Step 3 – Example Write of HEX File Data Line

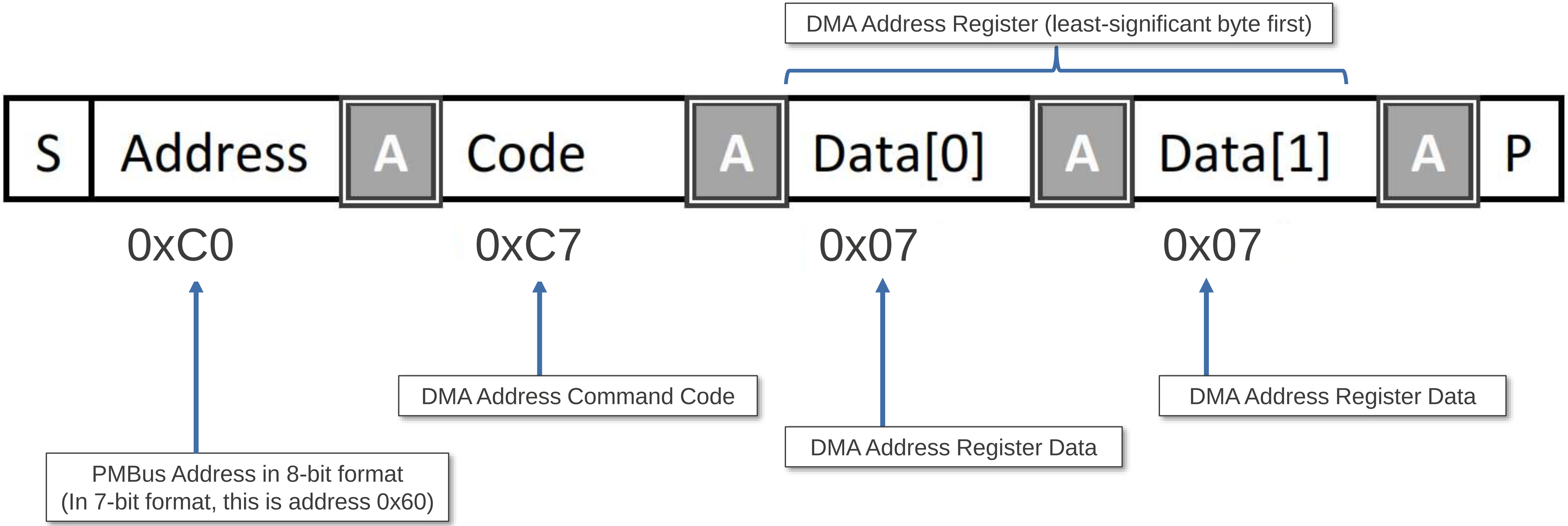
Example PMBus Command:

Hex File Line Data: 0007C0C6**E1000000**FC



Step 4a – Poll PROGRAMMER_STATUS Register

To check for the completion of device programming, first write to the DMA address register as shown below then read the DMA data register until bit 0 is set to 1.



Step 4b – Poll PROGRAMMER_STATUS Register

PMBus Address in 8-bit format
(In 7-bit format, this is address 0x60)

DMA Data Command Code

S	Address	A	Code	A	…
	0xC0		0xC5		

After completing step 4a, use the DMA read command to poll register until bit 0 is set to 1.

If after 2s timeout, bit 0 has not been set to 1, the part has failed programming. See step 4c for more details.

Sr	Address	A	D0[7:0]	A	D1[15:8]	A	D2[23:16]	A	D3[31:24]	N	P
	0xC1		0x01		0xXX		0xXX		0xXX		

For the purposes of programming devices, these bytes can be ignored.

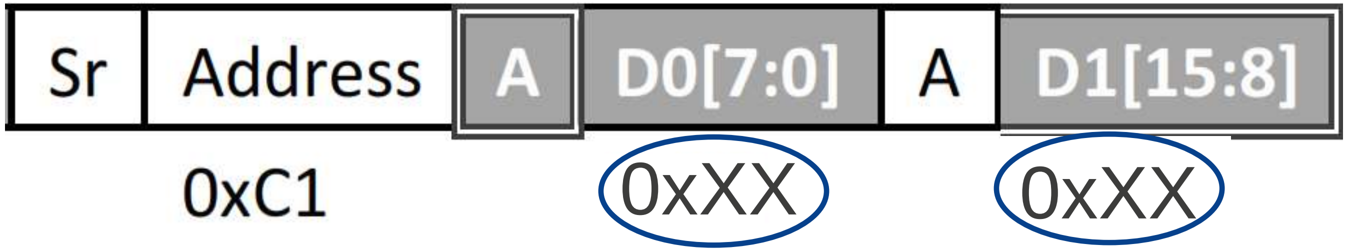
PMBus Address in 8-bit format
with the read bit enabled

Continue to read register until bit 0 is set to 1.

Step 4c – Programming Failure

To determine programming failure, take bits[8:0] from Step 4b and decode using the following…

Bits[8:0] from Step 4b



b0X0X0XXX0

If bit 8 is 1, the HEX file contains more configurations than are available. Programming fails before OTP banks are consumed.

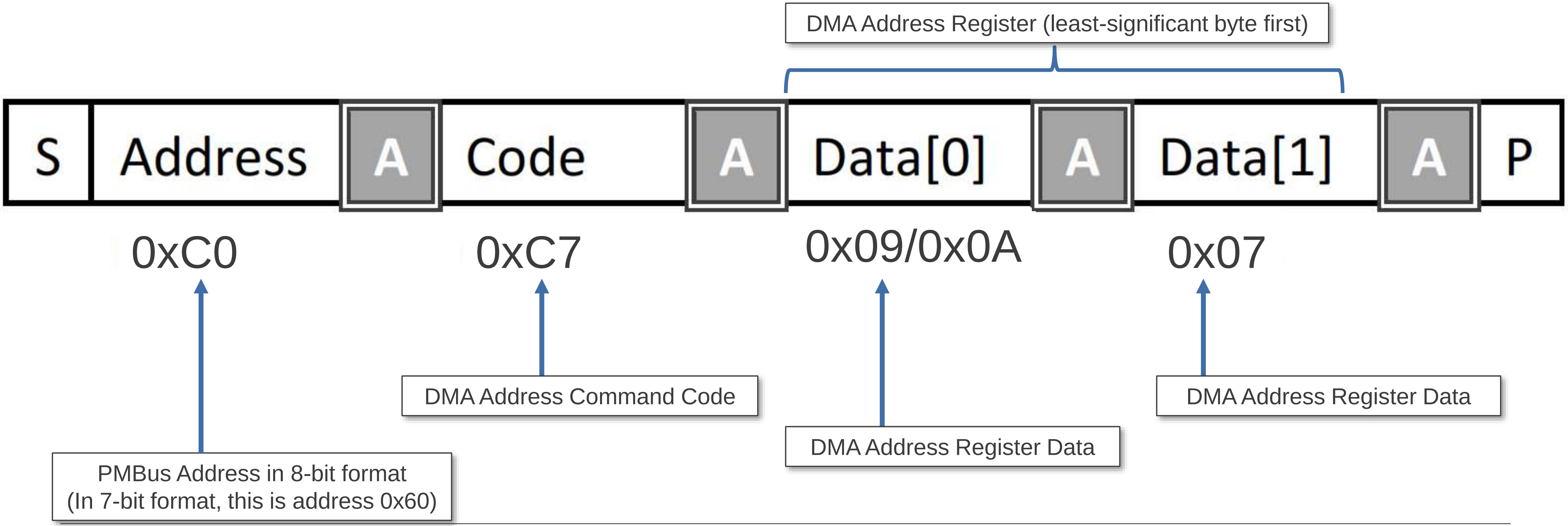
If bit 6 is 1, the CRC check fails on the OTP memory. Programming fails **after** OTP banks are consumed.

If bit 4 is 1, a CRC mismatch exists within the configuration data. Programming fails before OTP banks are consumed.

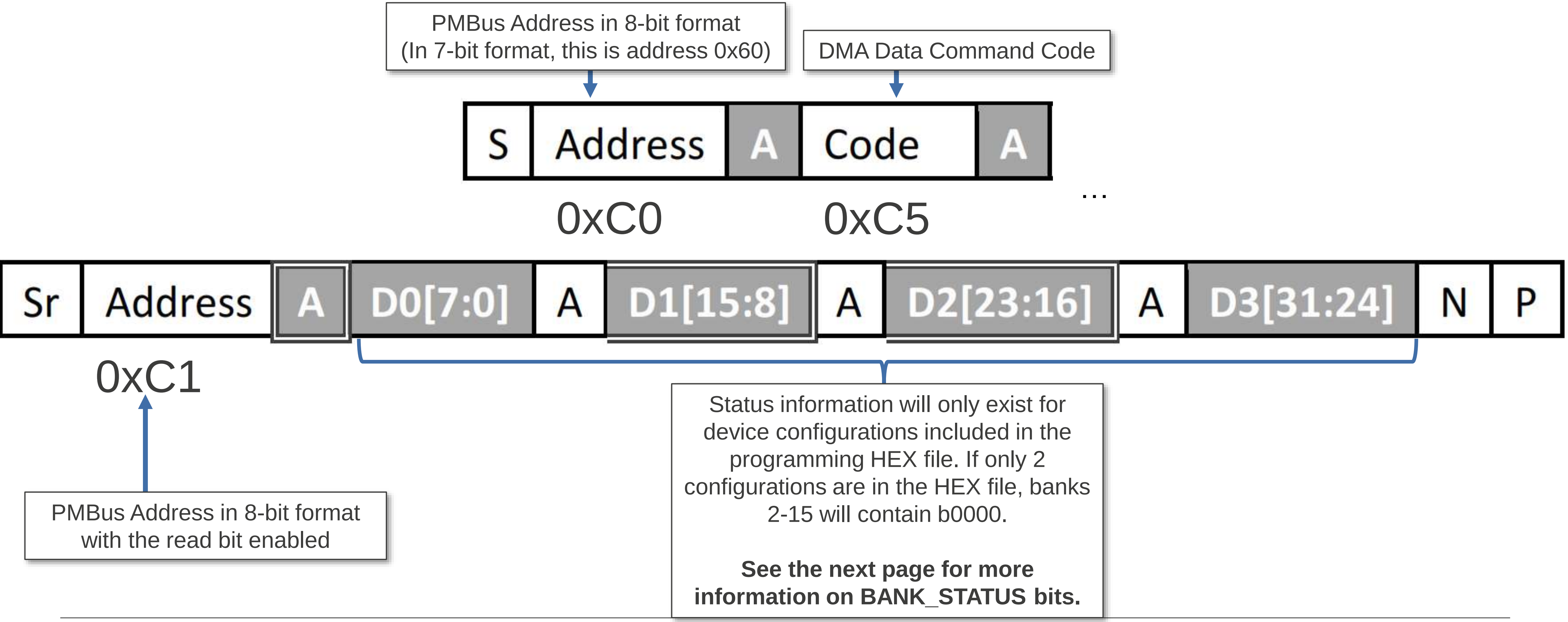
If this bit is 0, programming has failed.

Step 4d – Read BANK_STATUS Registers

To read the BANK_STATUS registers, first write to the DMA address register as shown below then read the DMA data register to retrieve BANK_STATUS data.



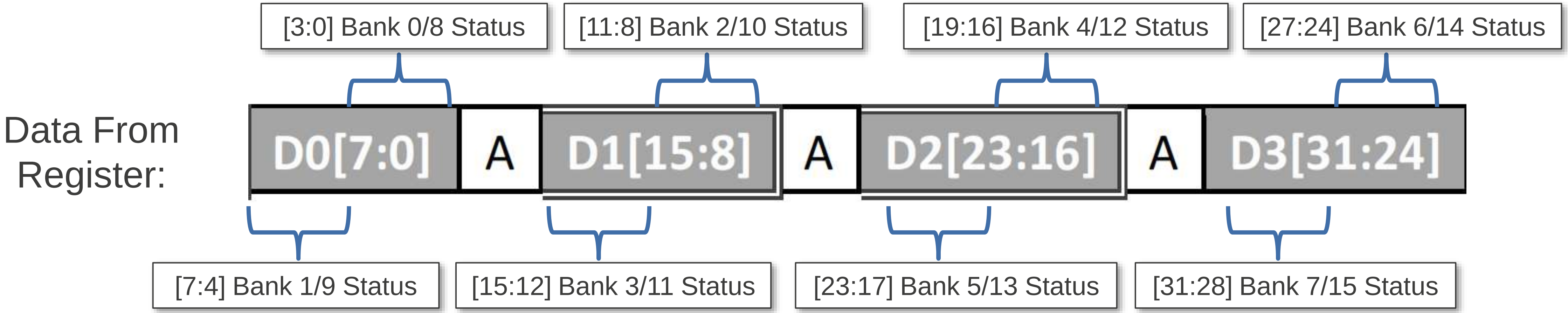
Step 4d – Read BANK_STATUS Registers



Step 4d – Read BANK_STATUS Registers

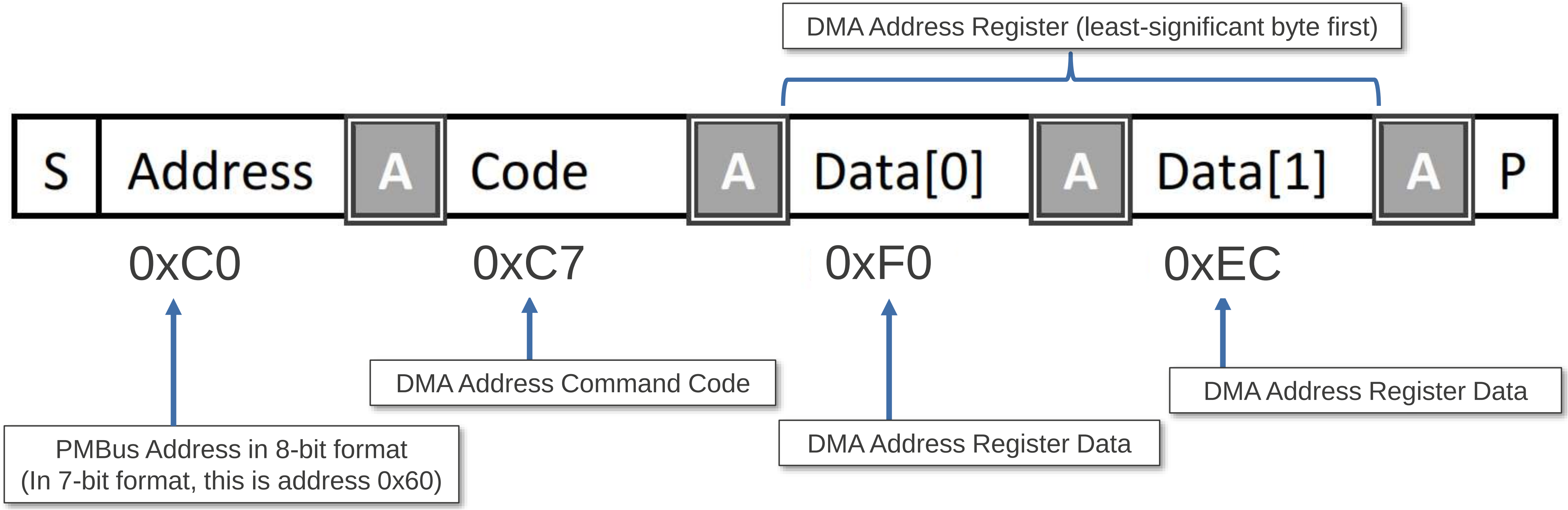
Every set of bank status bits will correspond to the table below:

Bank Status Bits	Description
b1000	Fail: CRC mismatch OTP
b0100	Fail: CRC mismatch RAM
b0010	Reserved
b0001	Bank Written (No Failures)
b0000	Bank Unaffected



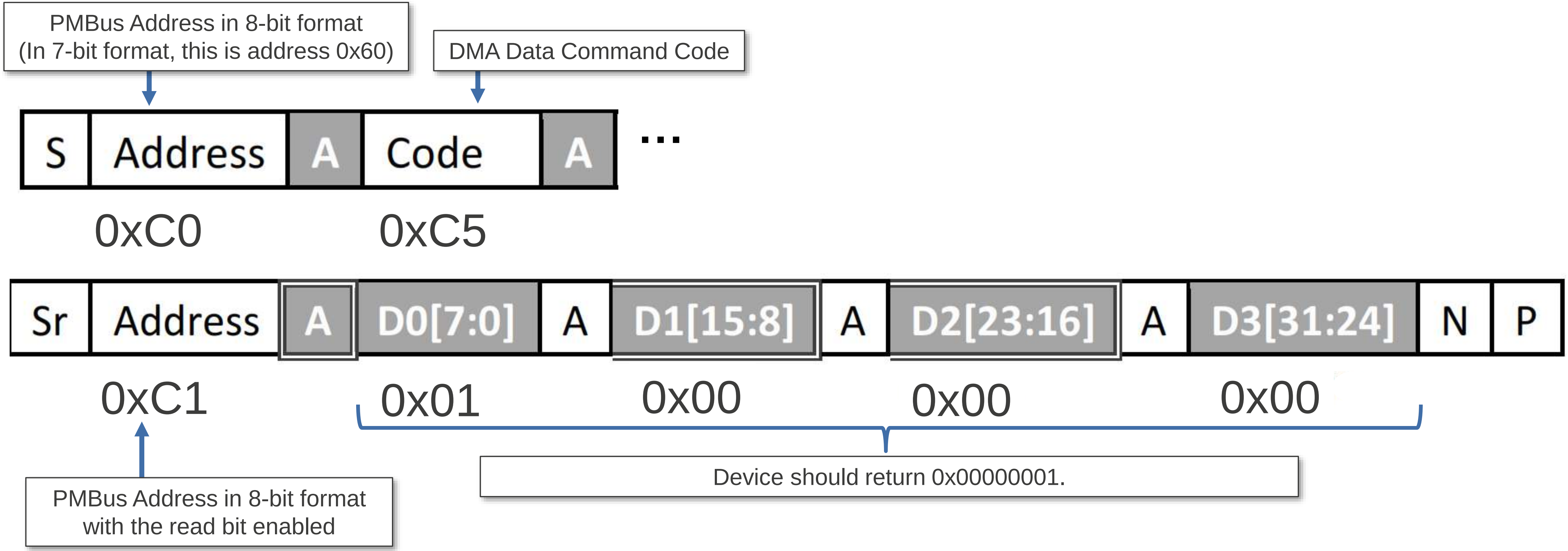
Step 5a – Write to DMA Address Register

After cycling VCC with a 50ms delay, first set the DMA address as shown below.



Step 5b – Read DMA Data Register

Next, Read the content of the register pointed to in Step 5a.



BIG IDEAS FOR EVERY SPACE

Step 5c – Write DMA Data Register

Next, Write 0x00000009 to the register selected in Step 5a.

PMBus Address in 8-bit format
(In 7-bit format, this is address 0x60)

DMA Data Command Code

S	Address	A	Code	A	…
	0xC0		0xC5		

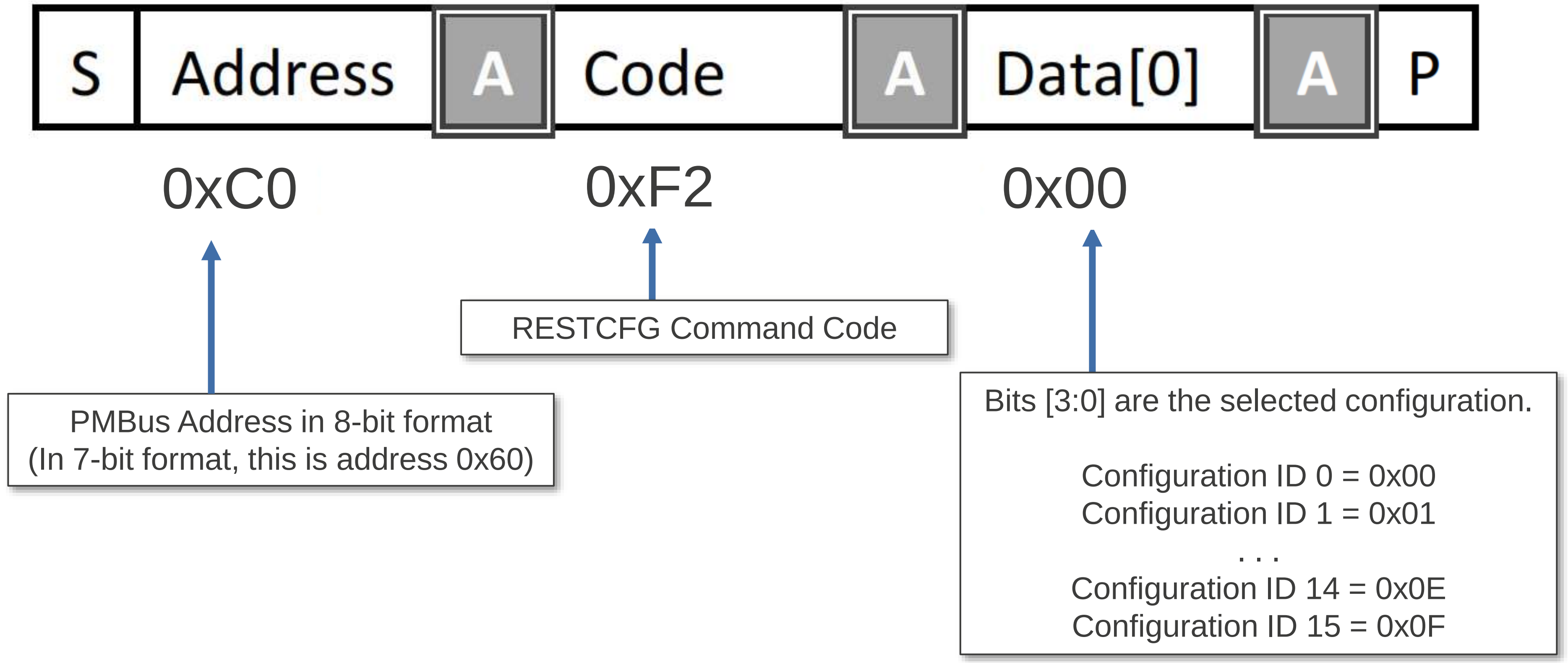
Data[3]	A	Data[2]	A	Data[1]	A	Data[0]	A	P
0x09		0x00		0x00		0x00		

Write 0x00000009 to the device.

Steps 5b and 5c are a read-modify-write to set bits [4:1] to b0100.

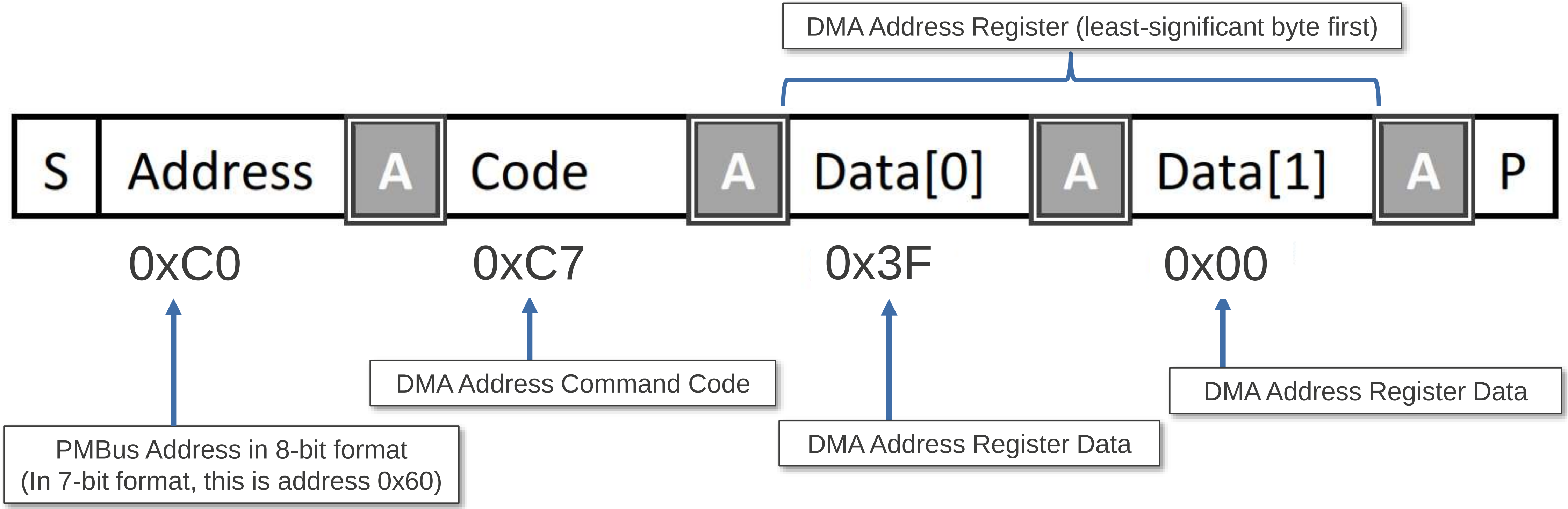
Step 5d – Use RESTCFG Command

Next, use the RESTCFG command code in the format shown below. Do **not** use this command if the device is regulating.



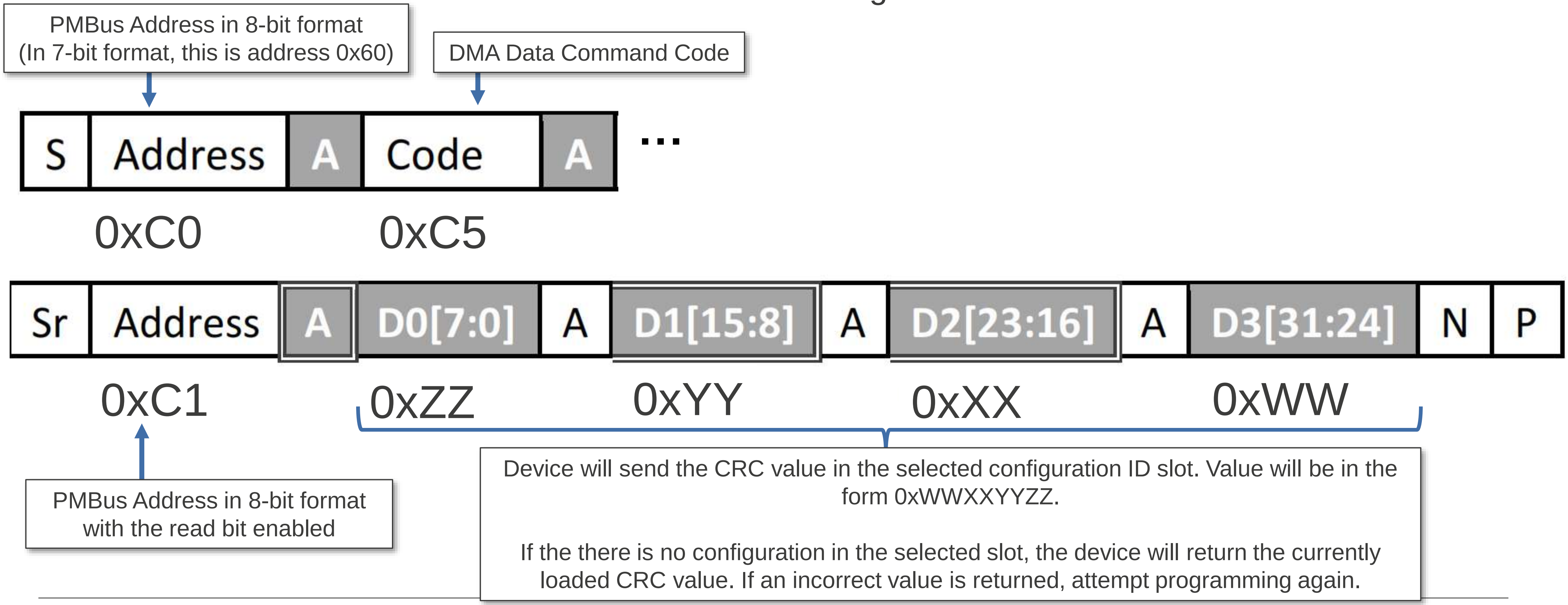
Step 5e – Write to DMA Address Register

To read the CRC value from the Configuration ID selected in Step 5d, first set the DMA address as shown below.



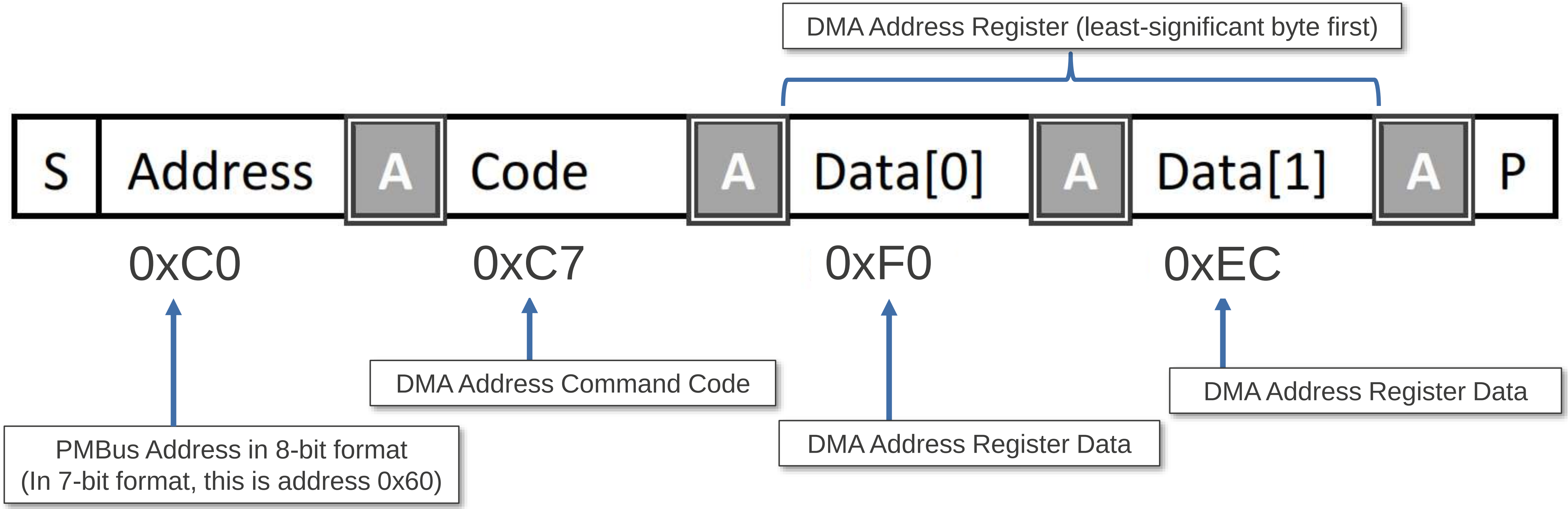
Step 5f – Read DMA Data Register

Next, Read the content of the register pointed to in Step 5e. This is the CRC value in the selected Configuration ID slot.



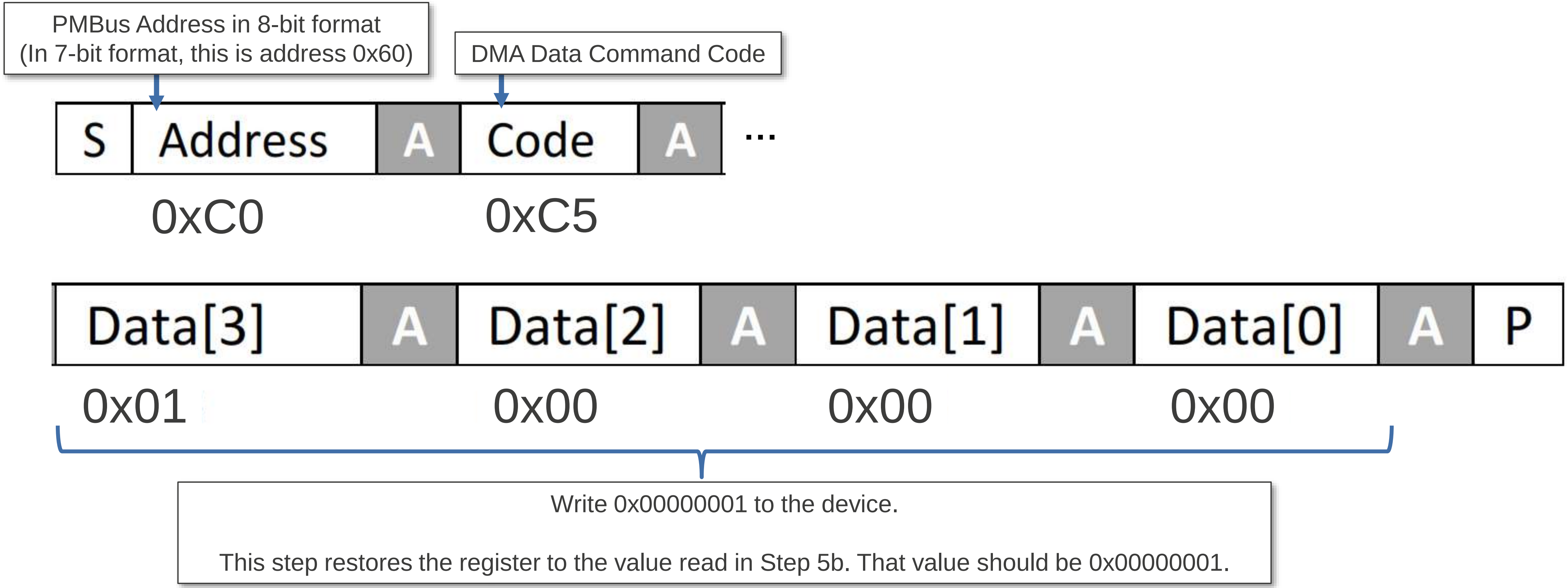
Step 5g – Write to DMA Address Register

To return to normal operation, set the DMA address as shown below.



Step 5h – Write DMA Data Register

Next, Write 0x00000001 to the register selected in Step 5g.



 RENESAS CONFIDENTIAL

BIG IDEAS FOR EVERY SPACE

Algorithm Completion

- After successfully completing programming, 3.3V VCC must be powered down to apply changes if not cycled during Step 5.

SECTION 3: HEX FILE CRC INFORMATION

HEX Files – Total Number of Configurations

$$\text{Total number of configurations} = \frac{\text{\# of lines in HEX file} - 290}{358}$$

```
1    4907C0AD49D24800AA
2    4907C0AE0200000017
3    4907C00000000200B0
4    490AC001352E342E31343003
5    490BC00200000169A21690A97D
6    0005C0E6020033

...

640  0005C0C70C07D8
641  0007C0C60100000098
642  0005C0C721018D
643  0007C0C6AD1D0000BC
644  0005C0C7DB001C
645  0007C0C600000008E
646  0005C0C7DD0062
647  0007C0C600000008E
648  0005C0E6060067
649
```

In the example HEX file, there are 648 lines.

$$\text{Total number of configurations} = \frac{\text{\# of lines in HEX file} - 290}{358}$$

$$\text{Total number of configurations} = \frac{648 - 290}{358}$$

$$\text{Total number of configurations} = \frac{358}{358}$$

$$\text{Total number of configurations} = 1$$

The example HEX file contains **1 configuration**.

HEX Files – Configuration Slot IDs

Slot ID line number = (N ∗ 358) + 282, N = 0, 1, . . ., # of configurations in file – 1

```
1    4907C0AD49D24800AA
2    4907C0AE0200000017
3    4907C000000000200B0
4    490AC001352E342E31343003
5    490BC00200000169A21690A97D
6    0005C0E6020033

          …

279  0007C0C6FFFF000074
280  0007C0C6B407000058
281  0007C0C6C605ABE0B1
282  0007C0C600FFFFFF81
283  0007C0C608160000E1
284  0007C0C600000008E
285  0007C0C60040060076

          …
```

In the example HEX file, there are 648 lines and 1 configuration.

N = 0, 1, . . ., # of configurations in file – 1

Slot ID line number = (N ∗ 358) + 282

For the 1st configuration in the file, N = 0.

Slot ID line number = (0 ∗ 358) + 282

Slot ID line number = 282

The 10th character in the line is the Slot ID.

The example HEX file contains a configuration in **Slot ID 0**.

HEX Files – Configuration CRCs

CRC line number = (N ∗ 358) + 600, N = 0, 1, . . ., # of configurations in file – 1

```
1    4907C0AD49D24800AA
2    4907C0AE0200000017
3    4907C000000000200B0
4    490AC001352E342E31343003
5    490BC00200000169A21690A97D
6    0005C0E6020033

               …

597  0007C0C6000000008E
598  0007C0C6000000008E
599  0007C0C6000000008E
600  0007C0C691EC3C7B99
601  0007C0C624FBA00080
602  0007C0C610000000E9

               …
```

In the example HEX file, there are 648 lines and 1 configuration.

N = 0, 1, . . ., # of configurations in file – 1

CRC line number = (N ∗ 358) + 600

For the 1st configuration in the file, N = 0.

CRC line number = (0 ∗ 358) + 600

CRC line number = 600

The 1st configuration CRC is **0x7B3CEC91**.

DMA COMMAND FORMAT REFERENCE

Direct Memory Access (DMA) Command Codes

This section explains direct memory access (DMA) commands.

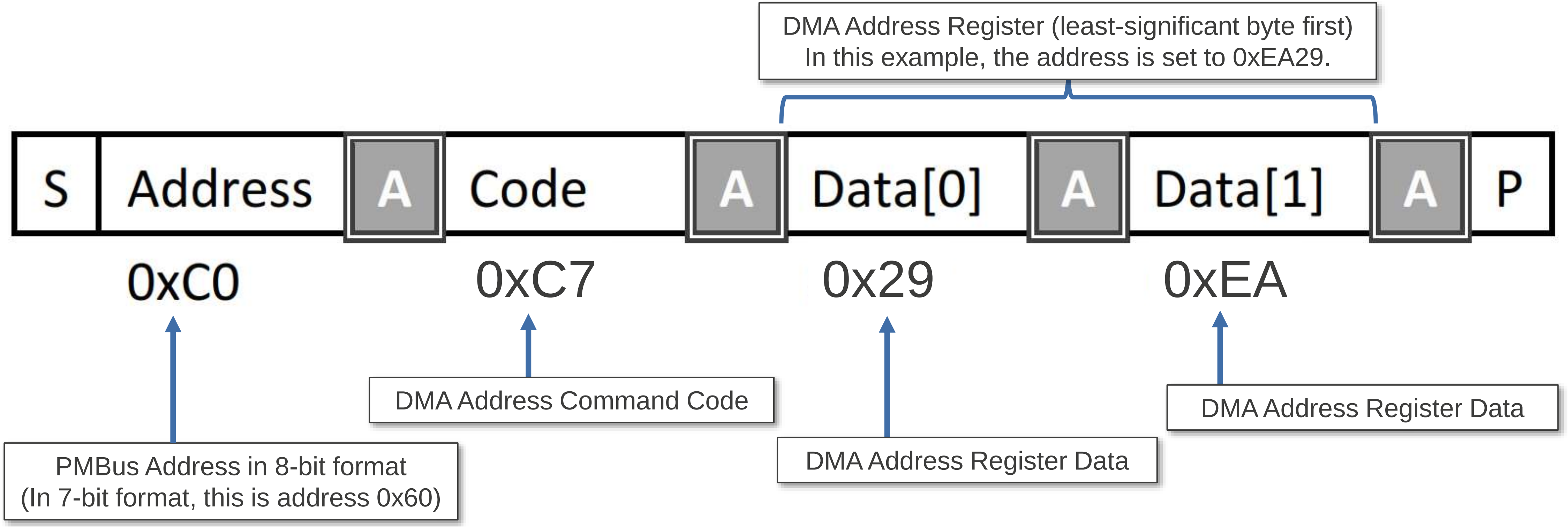
DMA is completed through 3 command codes:

- **DMA Address (Command Code 0xC7):** Used to set the register address to use with other DMA commands.
- **DMA Data (Command Code 0xC5):** Used to read from or write to the register selected by the DMA Address command.
- **DMA Sequential (Command Code 0xC6):** Used to read from or write to the register selected by the DMA Address command, then automatically increment the register address by 1.

DMA ADDRESS (0XC7)

DMA Address (0xC7) – Write

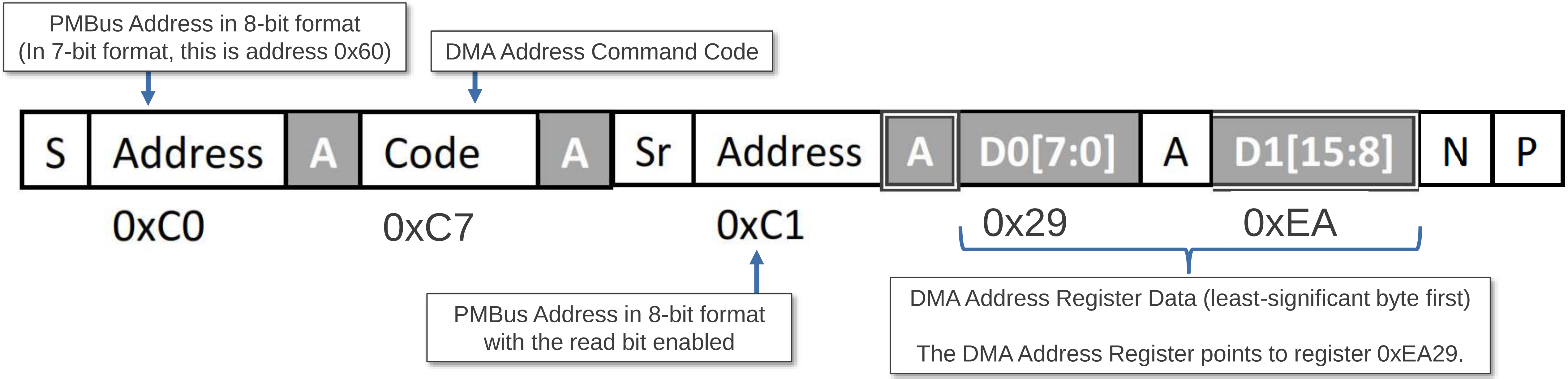
To set a pointer to a register for use with other DMA commands, use the DMA Address command (code 0xC7). This command accepts exactly two bytes of data.



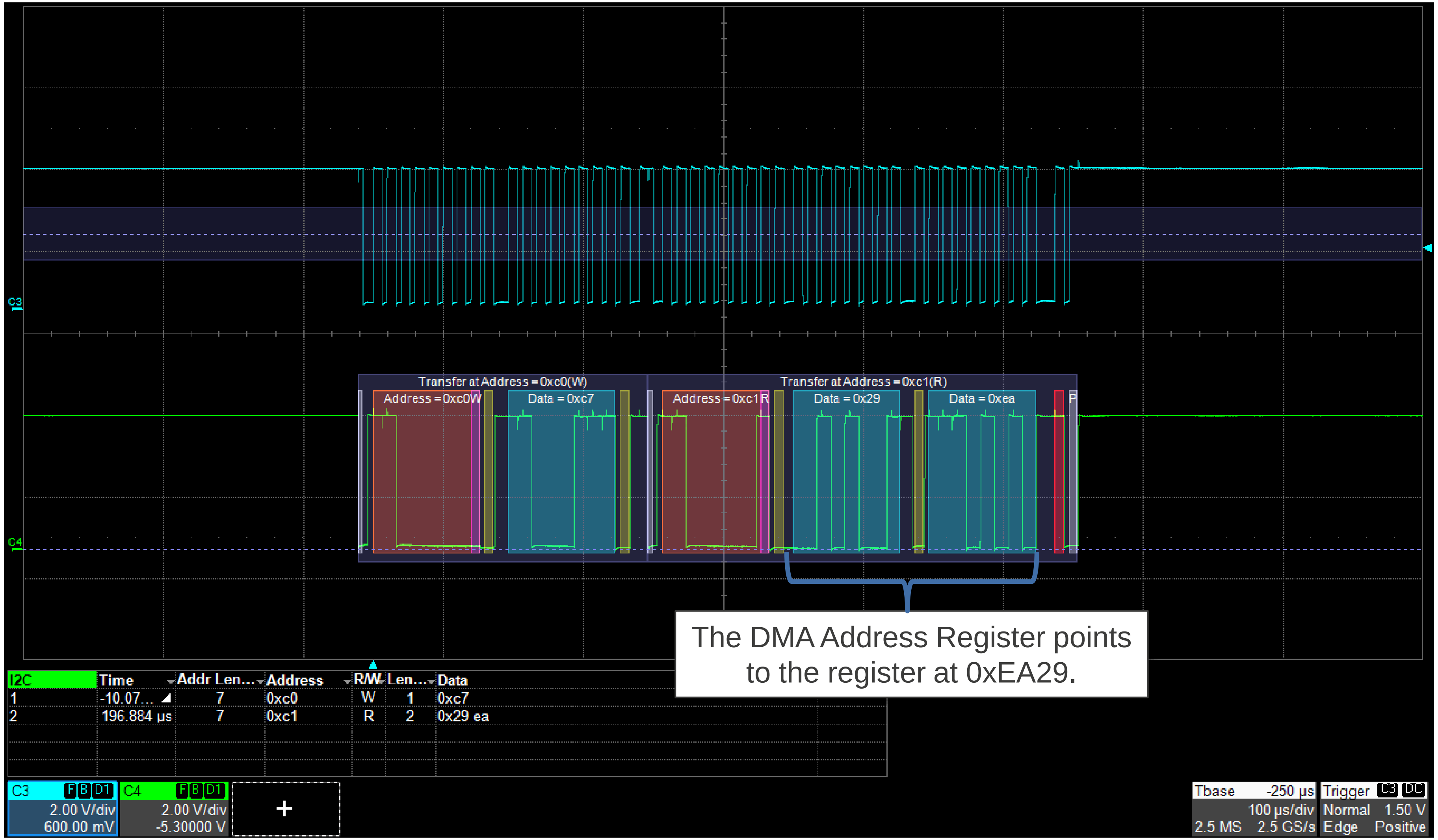
DMA Address (0xC7) – Write Waveform

DMA Address (0xC7) – Read

To read a pointer to a register used with other DMA commands, use the DMA Address command (code 0xC7). This command will return two bytes of data.



DMA Address (0xC7) – Read Waveform



DMA DATA (0XC5)

DMA Data (0xC5) – Write

To set the data at the register selected by the DMA Address Register, use the DMA Data command (code 0xC5). This command accepts exactly four bytes of data.

PMBus Address in 8-bit format
(In 7-bit format, this is address 0x60)

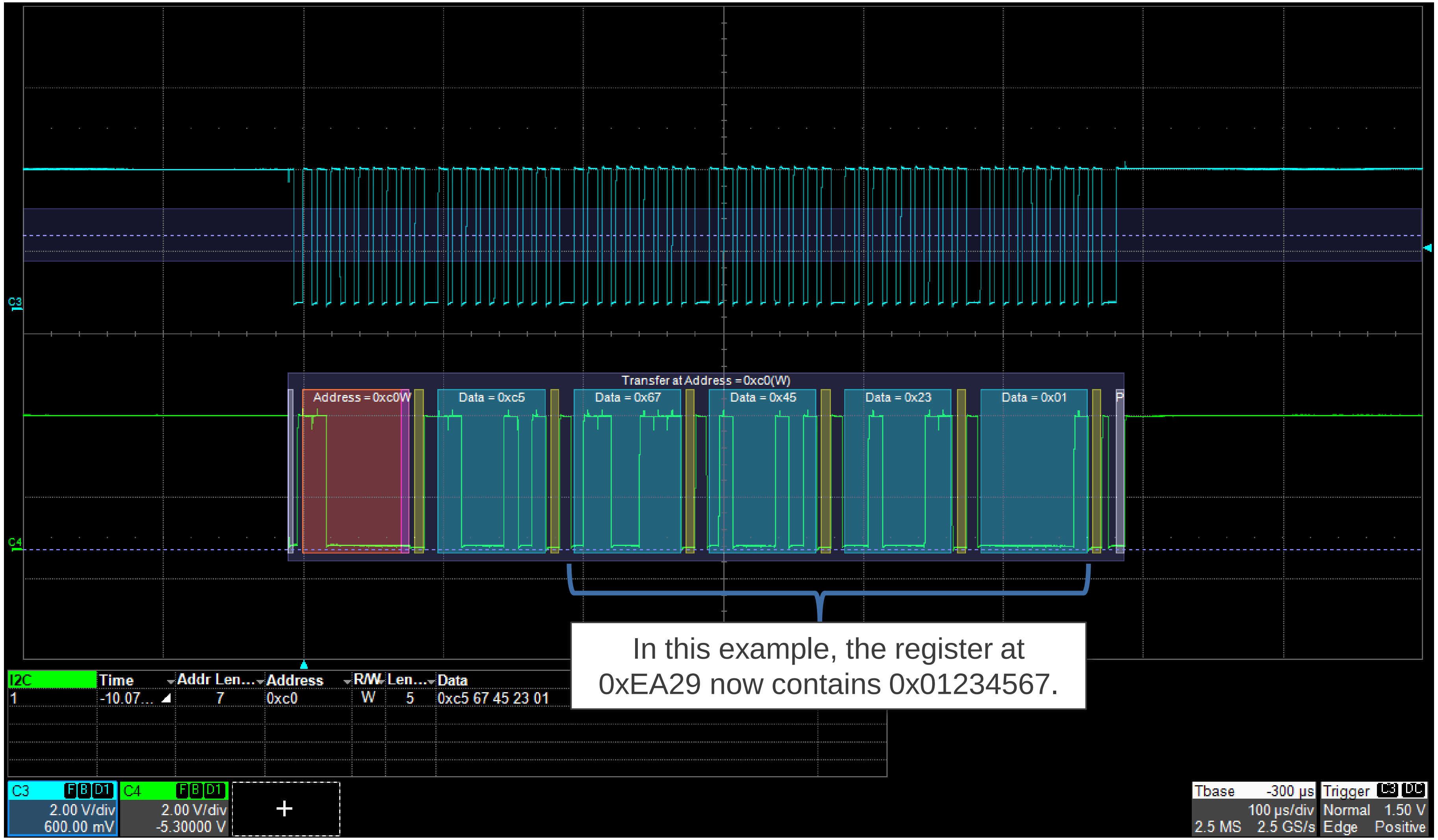
DMA Data Command Code

S	Address	A	Code	A	…
	0xC0		0xC5		

Data[3]	A	Data[2]	A	Data[1]	A	Data[0]	A	P
0x67		0x45		0x23		0x01		

In this example, the register selected by the DMA Address Register will contain 0x01234567.

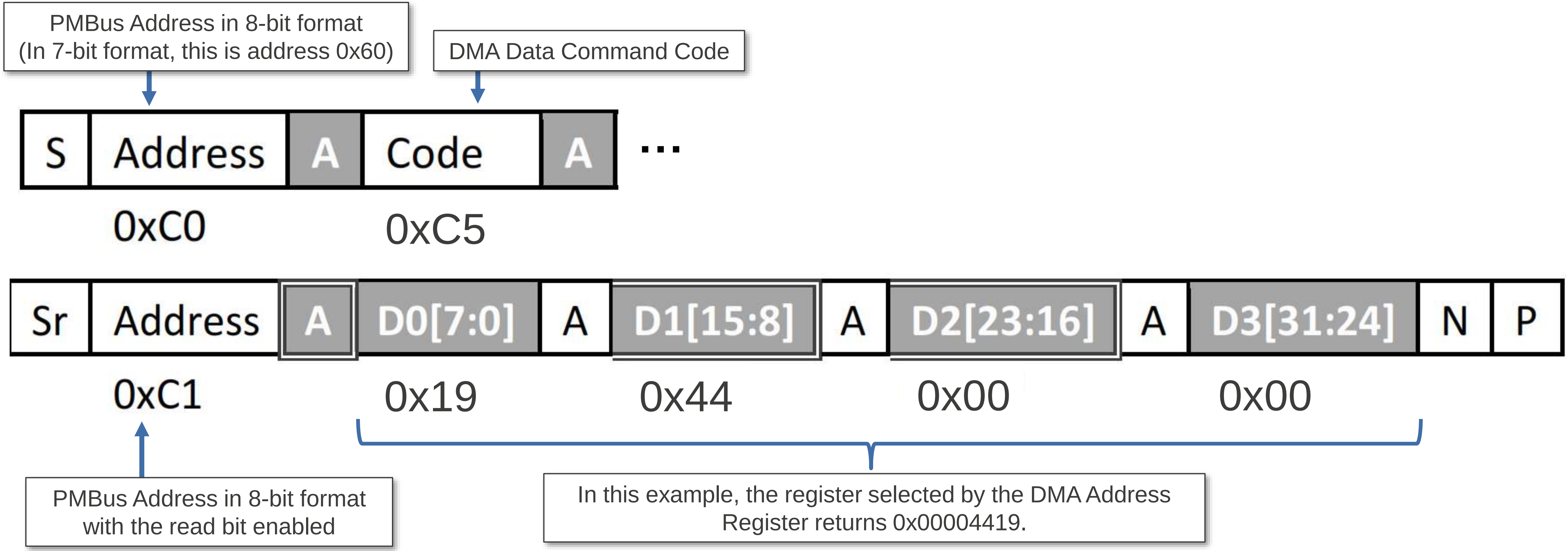
DMA Data (0xC5) – Write Waveform



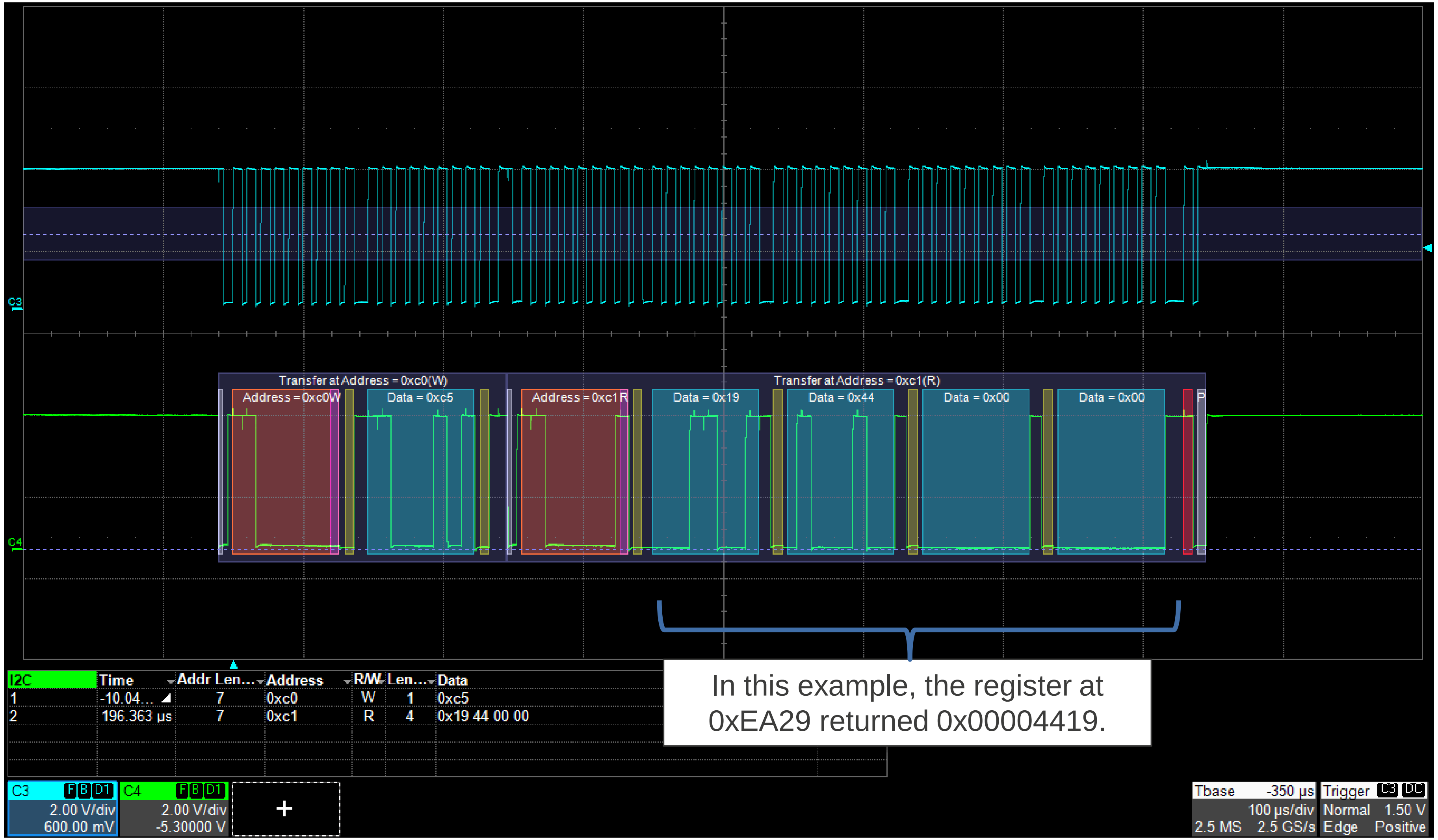
In this example, the register at 0xEA29 now contains 0x01234567.

DMA Data (0xC5) – Read

To read the data at the register selected by the DMA Address Register, use the DMA Data command (code 0xC5). This command returns four bytes of data.



DMA Data (0xC5) – Read Waveform

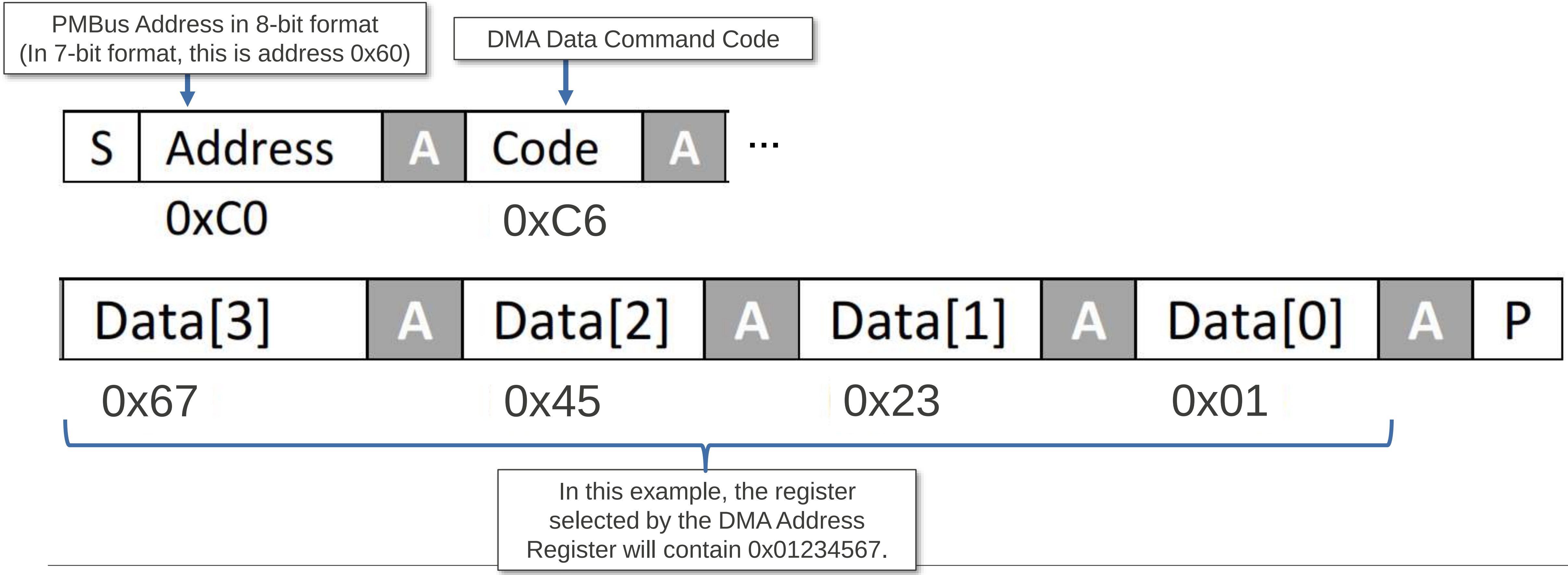


In this example, the register at 0xEA29 returned 0x00004419.

DMA SEQUENTIAL (0XC6)

DMA Sequential (0xC6) – Write

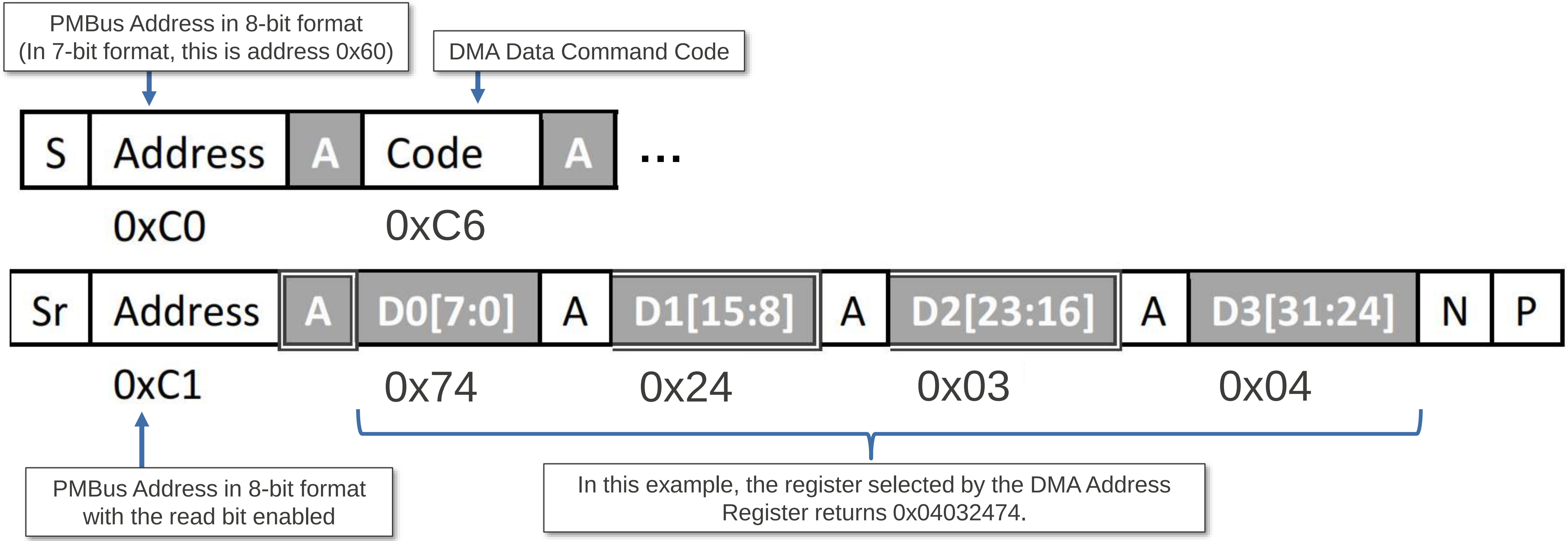
To set the data at the register selected by the DMA Address Register, use the DMA Sequential command (code 0xC6). This command accepts exactly four bytes of data. The DMA Sequential command then increments the DMA Address Register.



DMA Sequential (0xC6) – Write Waveform

DMA Sequential (0xC6) – Read

To read the data at the register selected by the DMA Address Register, use the DMA Sequential command (code 0xC6). This command returns four bytes of data. The DMA Sequential command then increments the DMA Address Register.



DMA Sequential (0xC6) – Read Waveform

In this example, the register at 0xEA29 returned 0x00004419.

The DMA Address Register now points to the register at 0xEA2A.

BIG IDEAS FOR EVERY SPACE

Renesas.com