

# RENESAS DIGITAL MULTIPHASE

## GENERATION 3 PROGRAMMING GUIDE

APRIL 2022

BIG IDEAS  
FOR EVERY SPACE

# OVERVIEW

- **This guide specifies the algorithm for programming Generation 3 Renesas 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
- **This document is intended for production silicon. For preproduction silicon, contact Renesas for support.**

# Supported Devices

Device Name	Package	IC_DEVICE_ID
ISL69260	40 LD QFN	0x49D28100
RAA228113	32 LD QFN	0x49D29000
RAA228234	40 LD QFN	0x49D2AF00
RAA228236	48 LD QFN	0x49D2AE00
RAA228924	48 LD QFN	0x49D28F00
RAA228926	68 LD QFN	0x49D28E00
RAA228928	56 LD QFN	0x49D2B400
RAA228929	48 LD QFN	0x49D2B200
RAA228930	40 LD QFN	0x49D2B300
RAA228931	40 LD QFN	0x49D2B500
RAA228932	56 LD QFN	0x49D2B600
RAA229126	48 LD QFN	0x49D28200
RAA229130	68 LD QFN	0x49D29700
RAA229131	68 LD QFN	0x49D29800
RAA229132	40 LD QFN	0x49D29600

Device Name	Package	IC_DEVICE_ID
RAA229325	48 LD QFN	0x49D28800
RAA229613	32 LD QFN	0x49D28A00
RAA229620	48 LD QFN	0x49D29B00
RAA229621	40 LD QFN	0x49D29C00
RAA229625	48 LD QFN	0x49D28B00
RAA229628	68 LD QFN	0x49D28D00
RAA229629	56 LD QFN	0x49D2A400
RAA229631	40 LD QFN	0x49D2A200
RAA229638	68 LD QFN	0x49D29D00
RAA229724	40 LD QFN	0x49D27F00
RAA229725	40 LD QFN	0x49D2B100
RAA229726	48 LD QFN	0x49D28000
RAA229817	52 LD QFN	0x49D28600
RAA229828	68 LD QFN	0x49D28700

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

# Prerequisites and Conventions

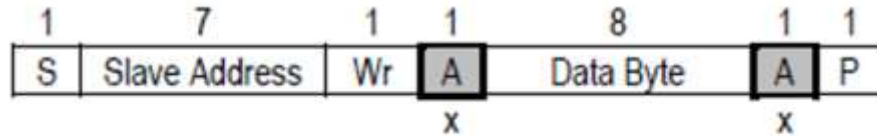
- HEX configuration files must be generated using PowerNavigator.
- In this guide, address 0x60 (7-bit format) is used in all examples.
- Data on the bus may be reversed. Follow examples for correct byte order.



# Minimum Pin and Component Requirements

**The following pins must be connected when programming a device:**

- PMSCL and PMSDA (I<sup>2</sup>C clock and data pins). These are open drain and must be pulled to 3.3V via a resistor (1k $\Omega$  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 4.7 $\mu$ F or greater MLCC (X5R or better).
- Ground pin must be connected to ground.
- ADDRESS/ADDR\_CFG pin must have an address set resistor to ground. Unless noted otherwise, connect directly to ground for address 0x60.
- Other pins may be floated.

# PMBus Communication Key



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

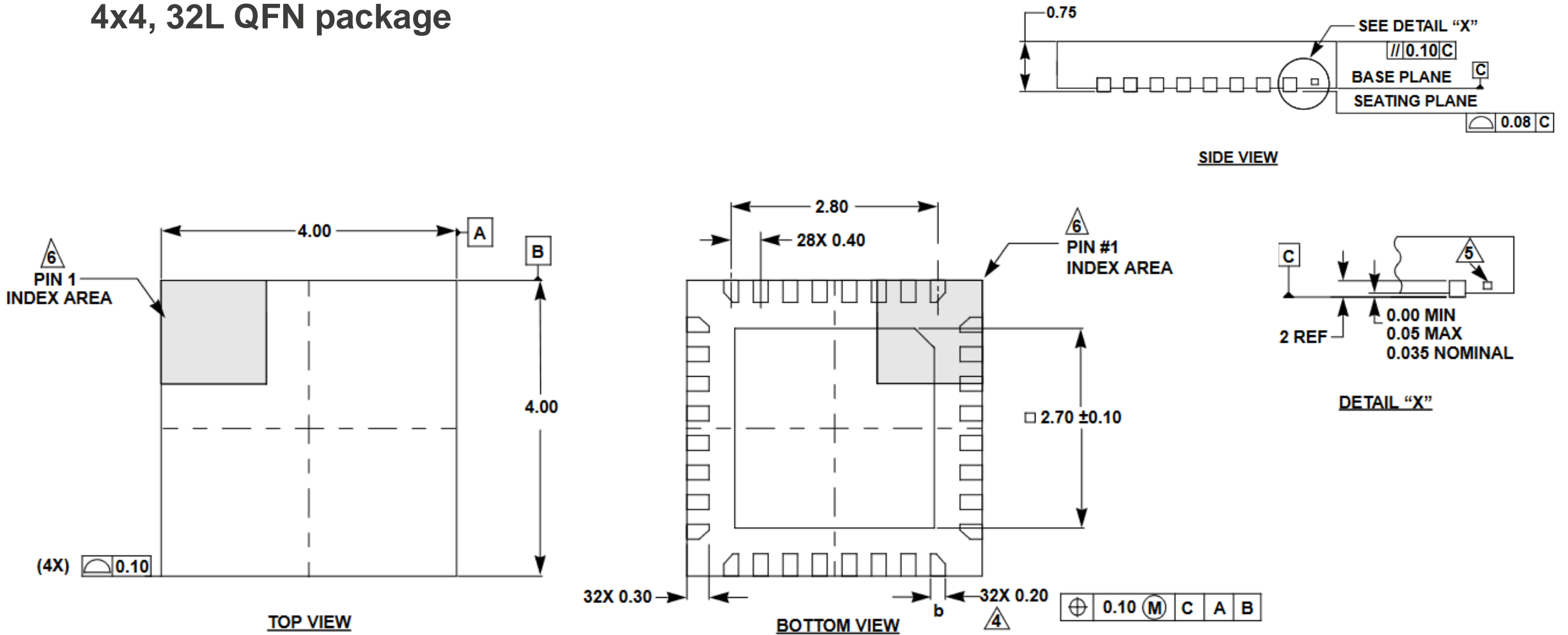
- The command that causes the OTP to burn is the last line of the HEX file. Device programming can be aborted at any point before this, and no contents will be burned to OTP.
- The last byte of each line in the HEX file is a CRC8 check for that line in the file only. It does not relate to any hardware value.



# SECTION 1: DEVICE PACKAGES AND PINOUTS

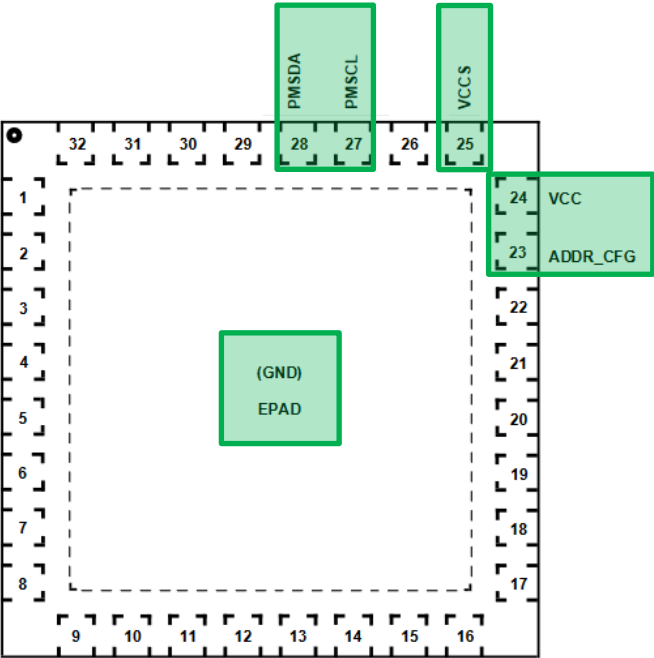
# 4x4 Package Characteristics

## 4x4, 32L QFN package



# 4x4 Pinouts and Devices

4x4, 32L QFN package

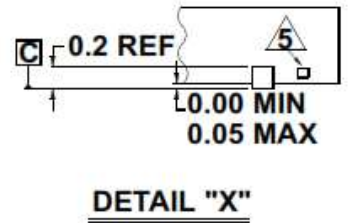
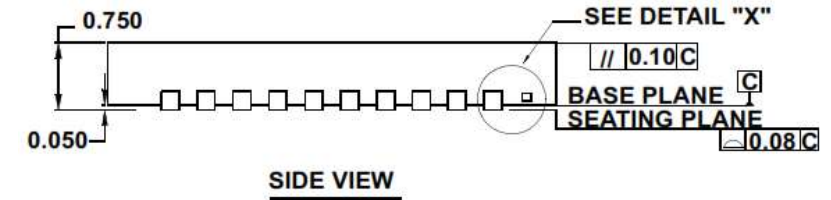
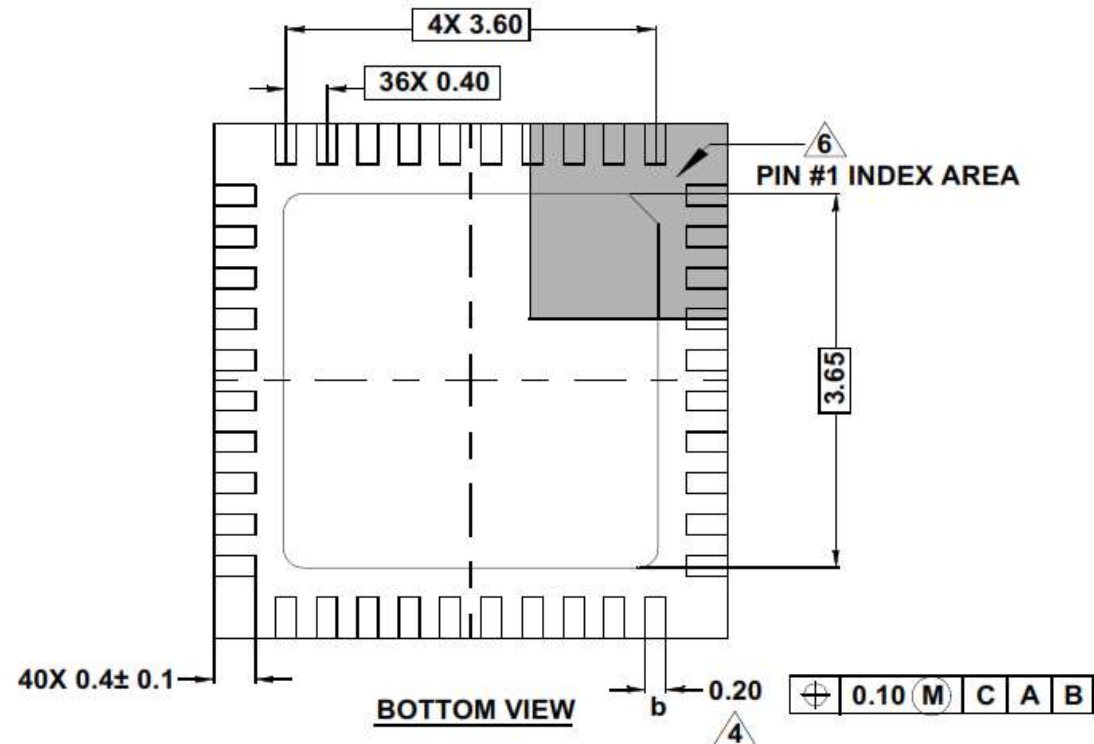
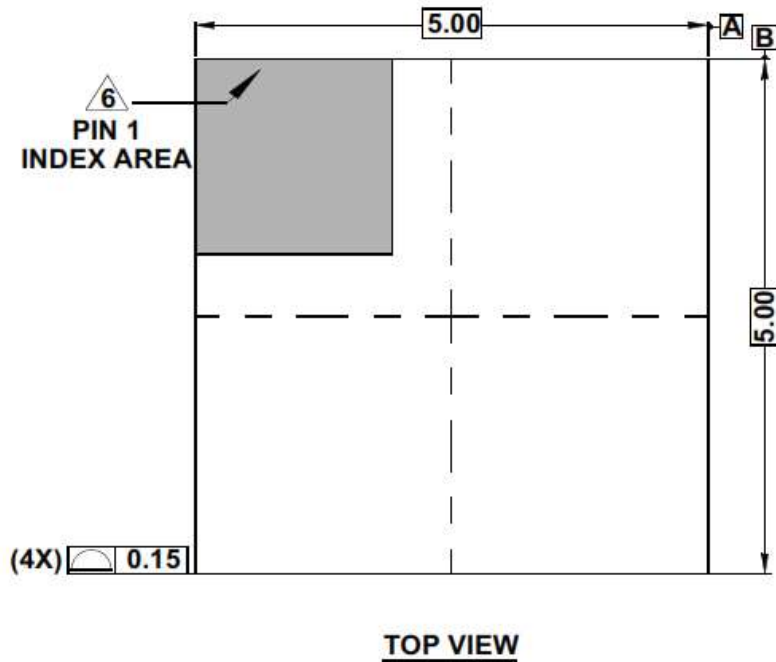


DEVICE TYPE
RAA228113
RAA229613

# 5x5 Package Characteristics

## 5x5, 40L QFN package

L40.5x5D  
40 LEAD QUAD FLAT NO-LEAD PLASTIC PACKAGE  
Rev 0, 9/10



# 5x5 Pinouts and Devices (1)

## 5x5, 40L QFN package

VCCS  
VCC

ADDRESS

(GND)  
EPAD

PMSDA  
PMSCL

DEVICE TYPE
ISL69260
RAA228234
RAA228930
RAA228931*
RAA229621
RAA229724
RAA229725

\*Device uses address 0x20

VCCS  
VCC

PMSDA  
PMSCL

ADDRESS

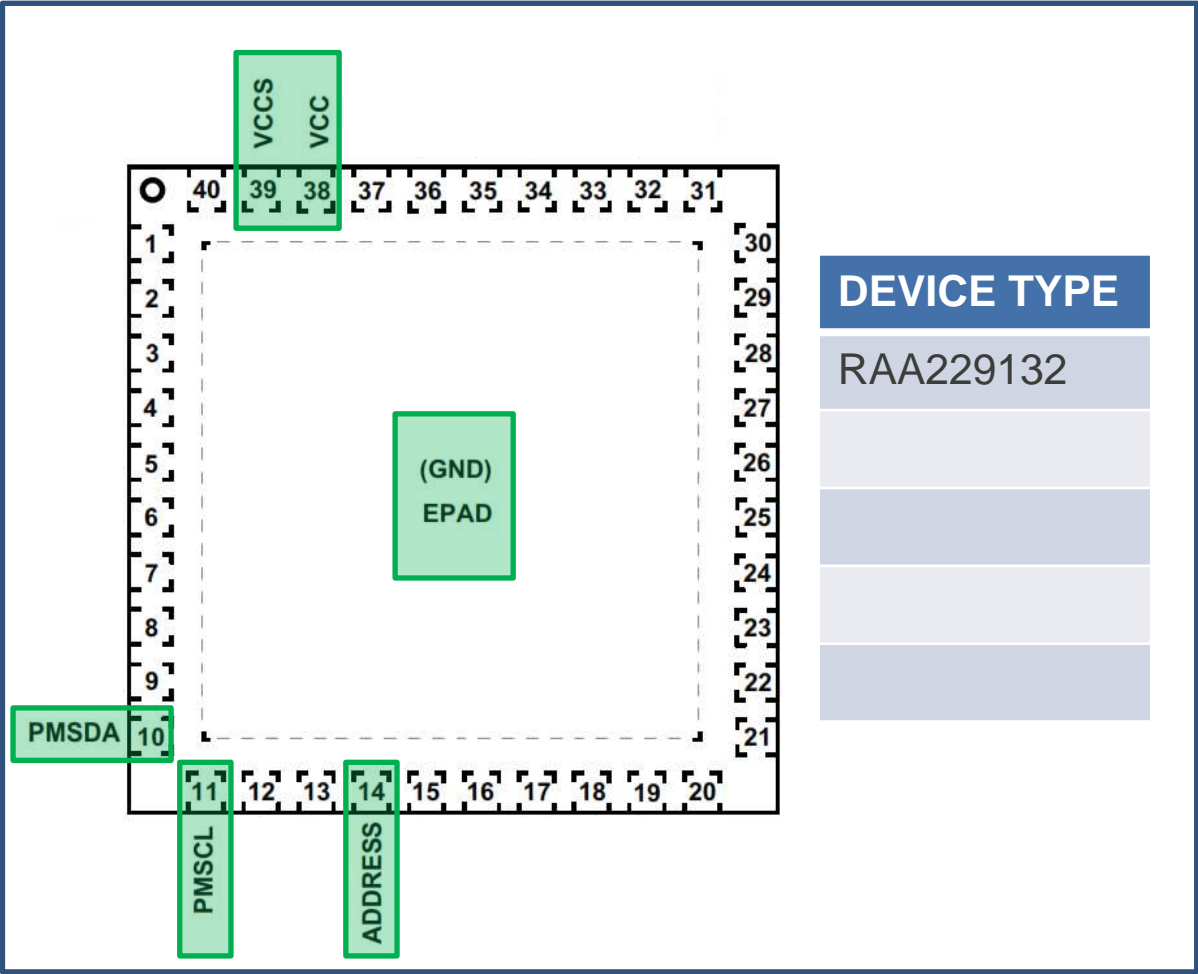
(GND)  
EPAD

DEVICE TYPE
RAA229631*

\*Device uses address 0x74

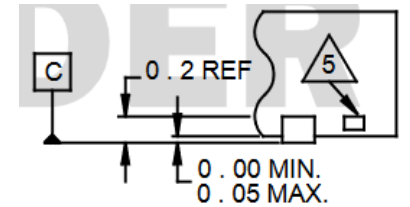
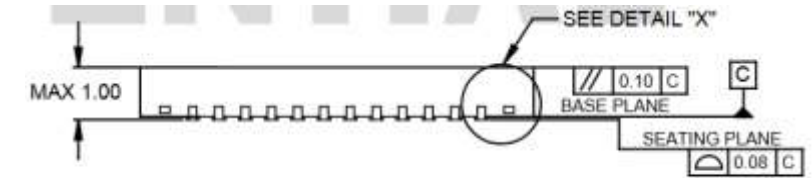
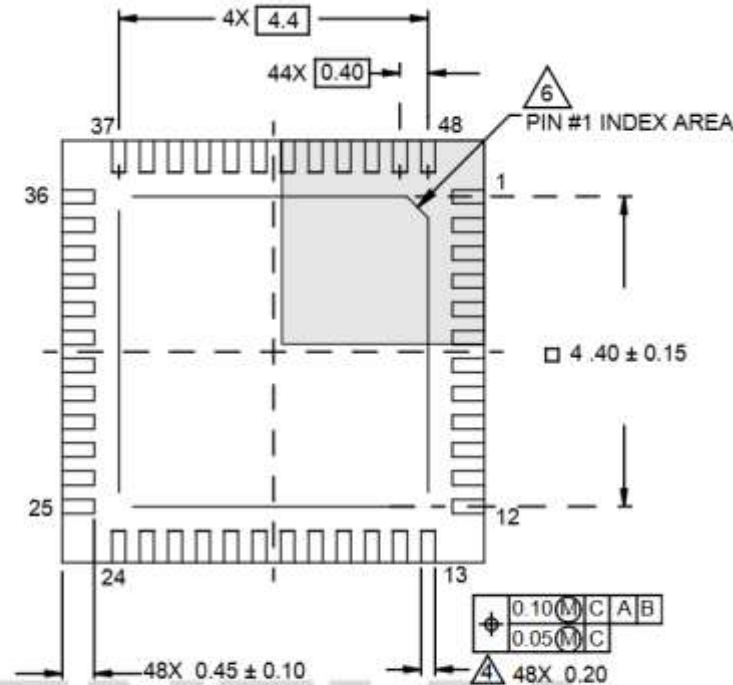
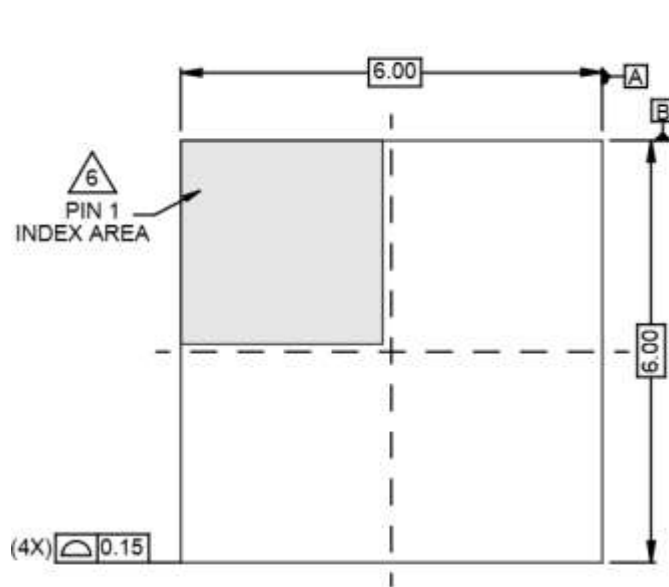
# 5x5 Pinouts and Devices (2)

5x5, 40L QFN package



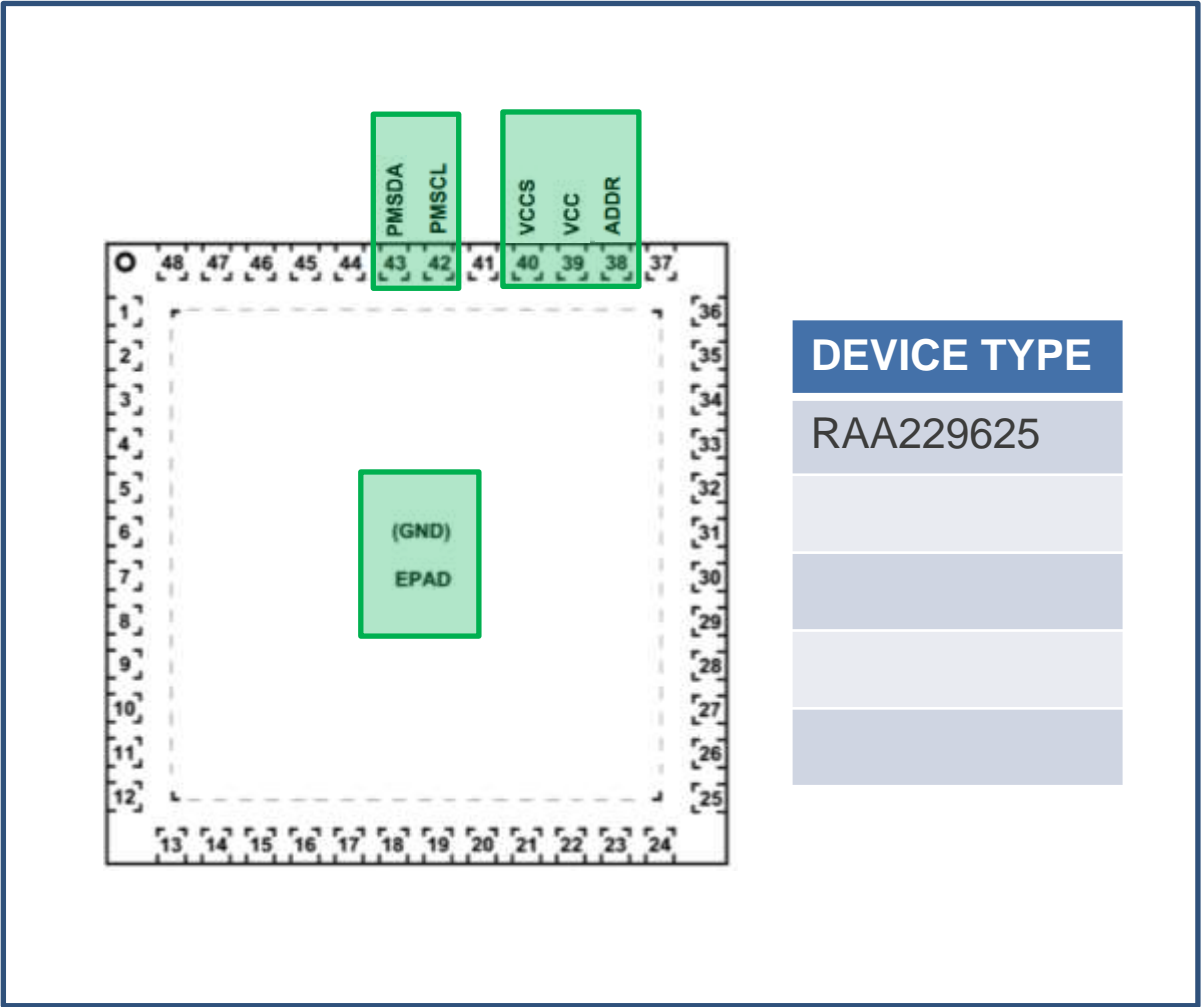
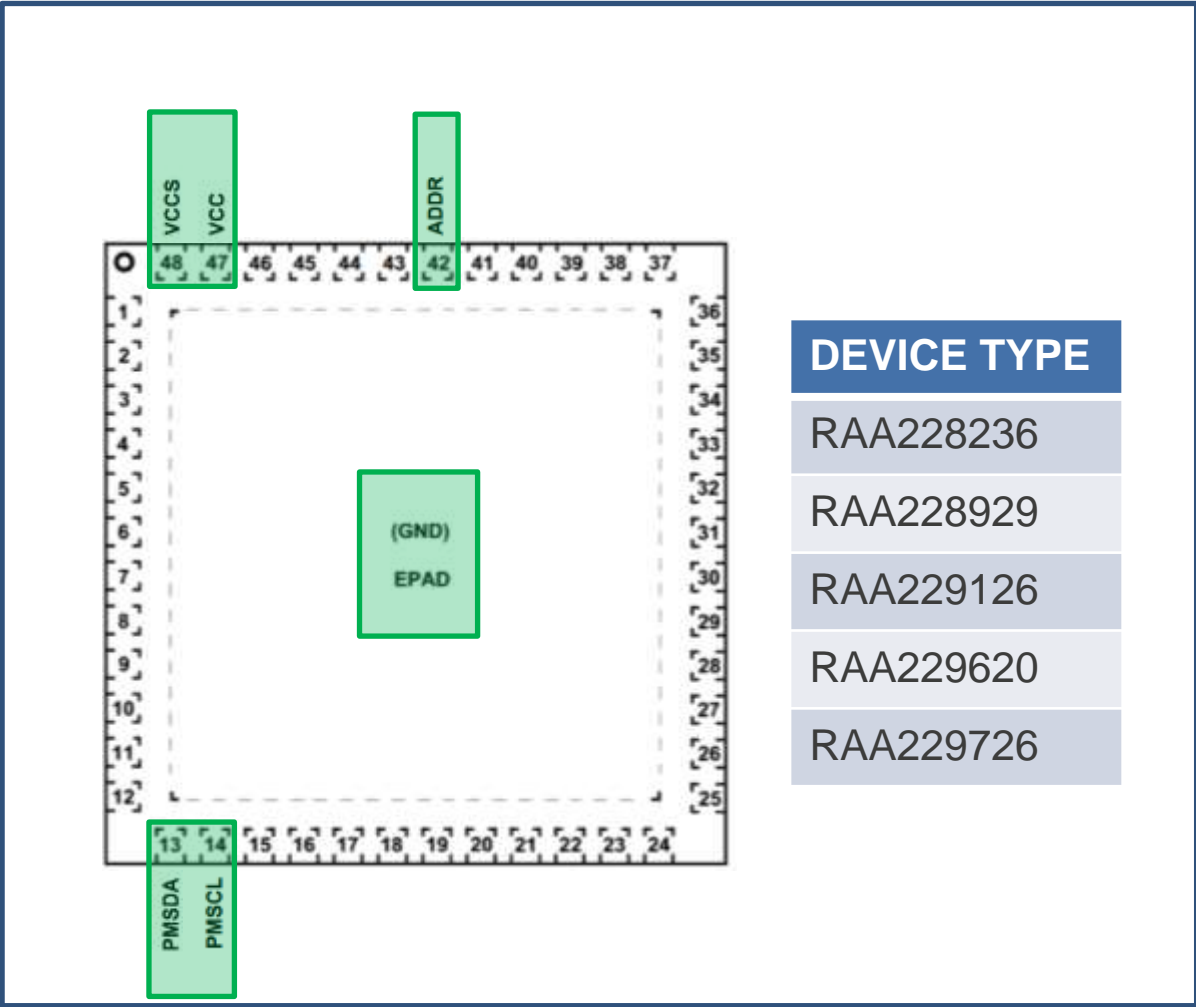
# 6x6 (48 LD) Package Characteristics

## 6x6, 48L QFN package



# 6x6 (48 LD) Pinouts and Devices (1)

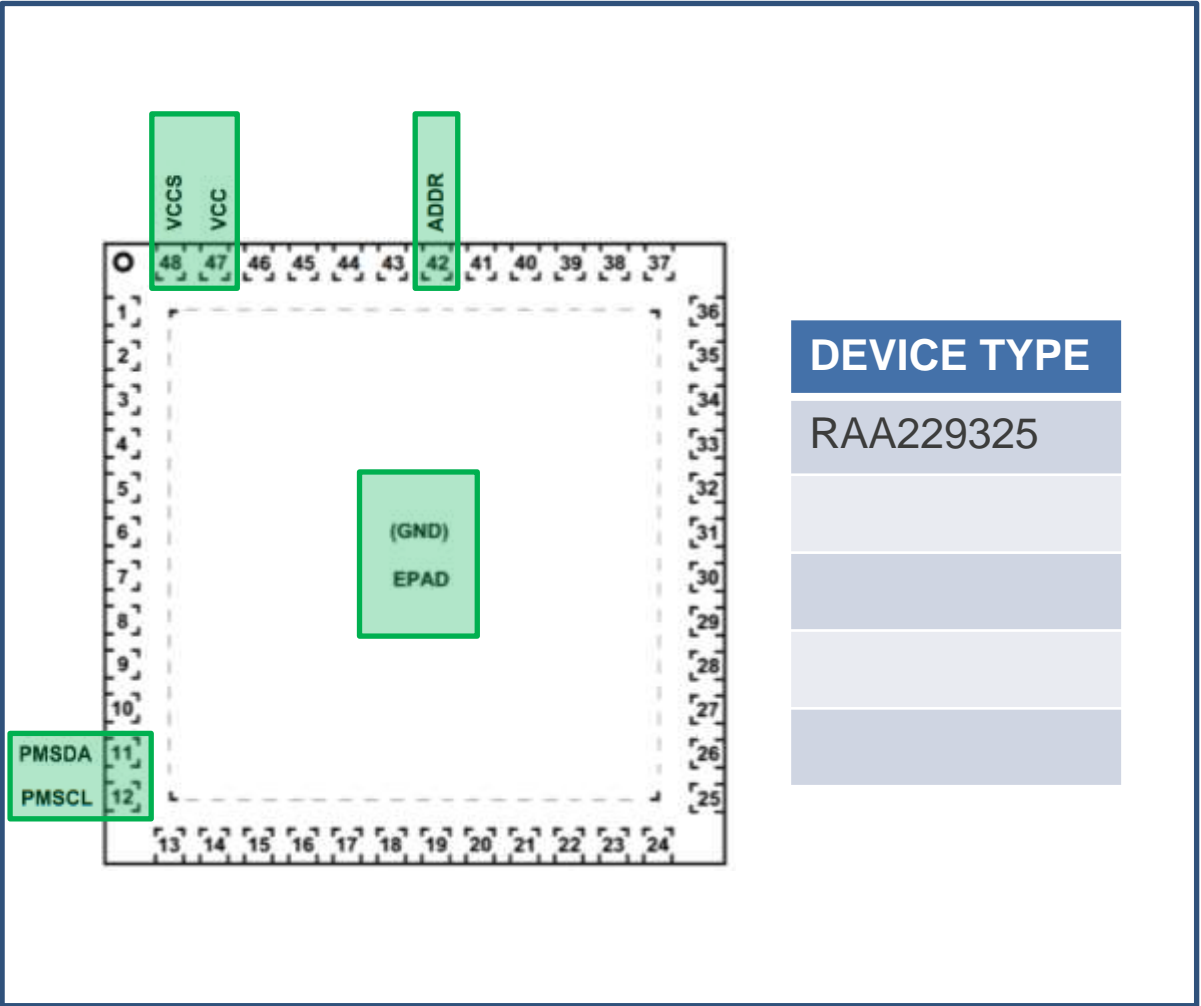
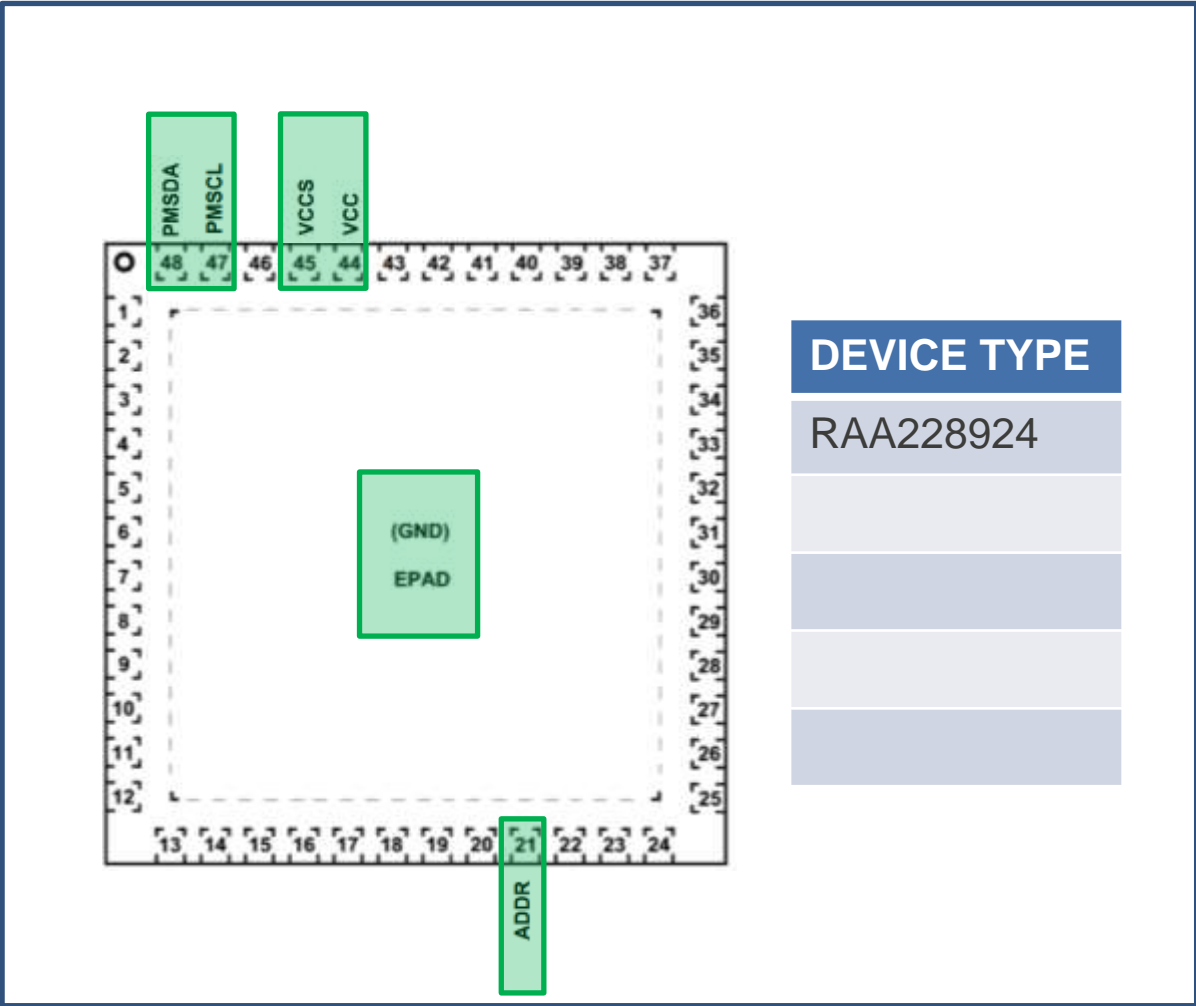
6x6, 48L QFN package





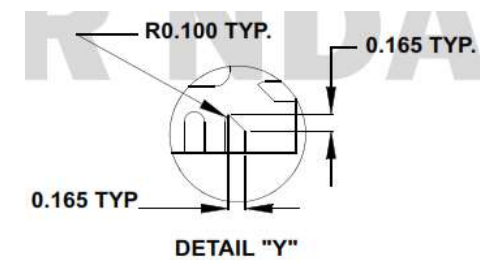
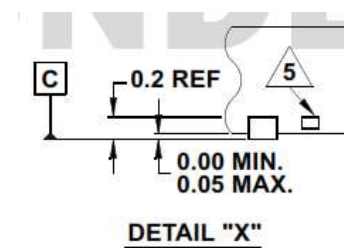
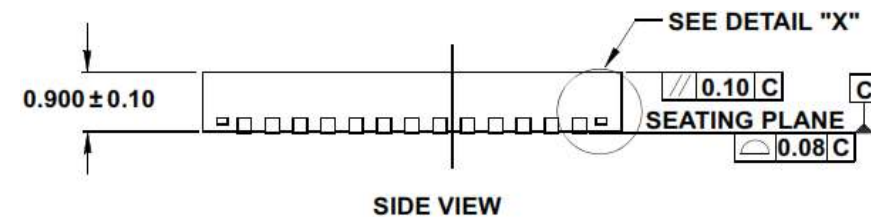
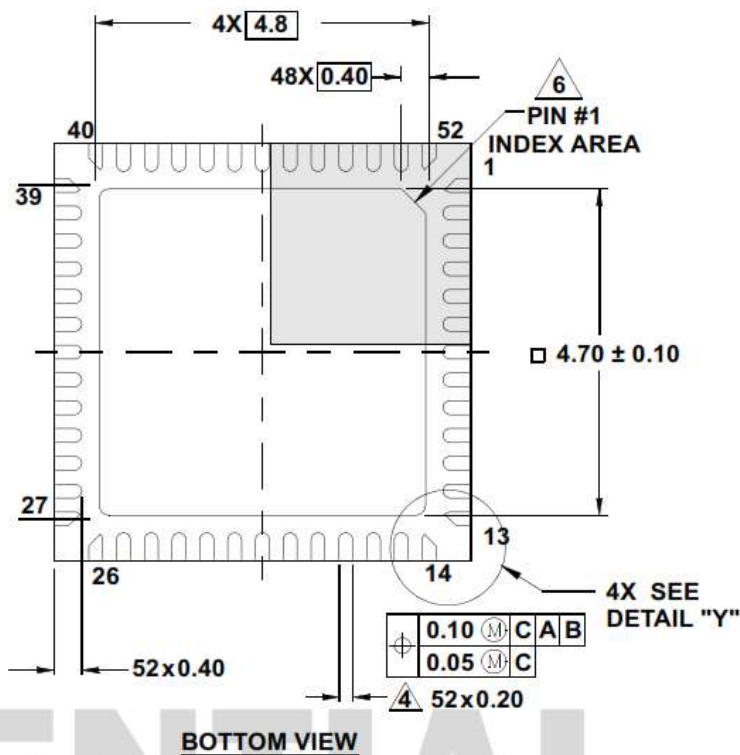
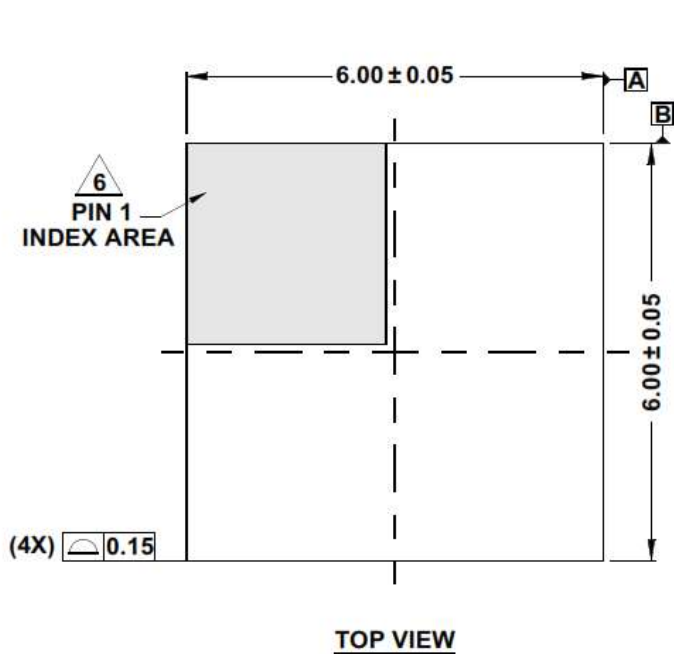
# 6x6 (48 LD) Pinouts and Devices (2)

6x6, 48L QFN package



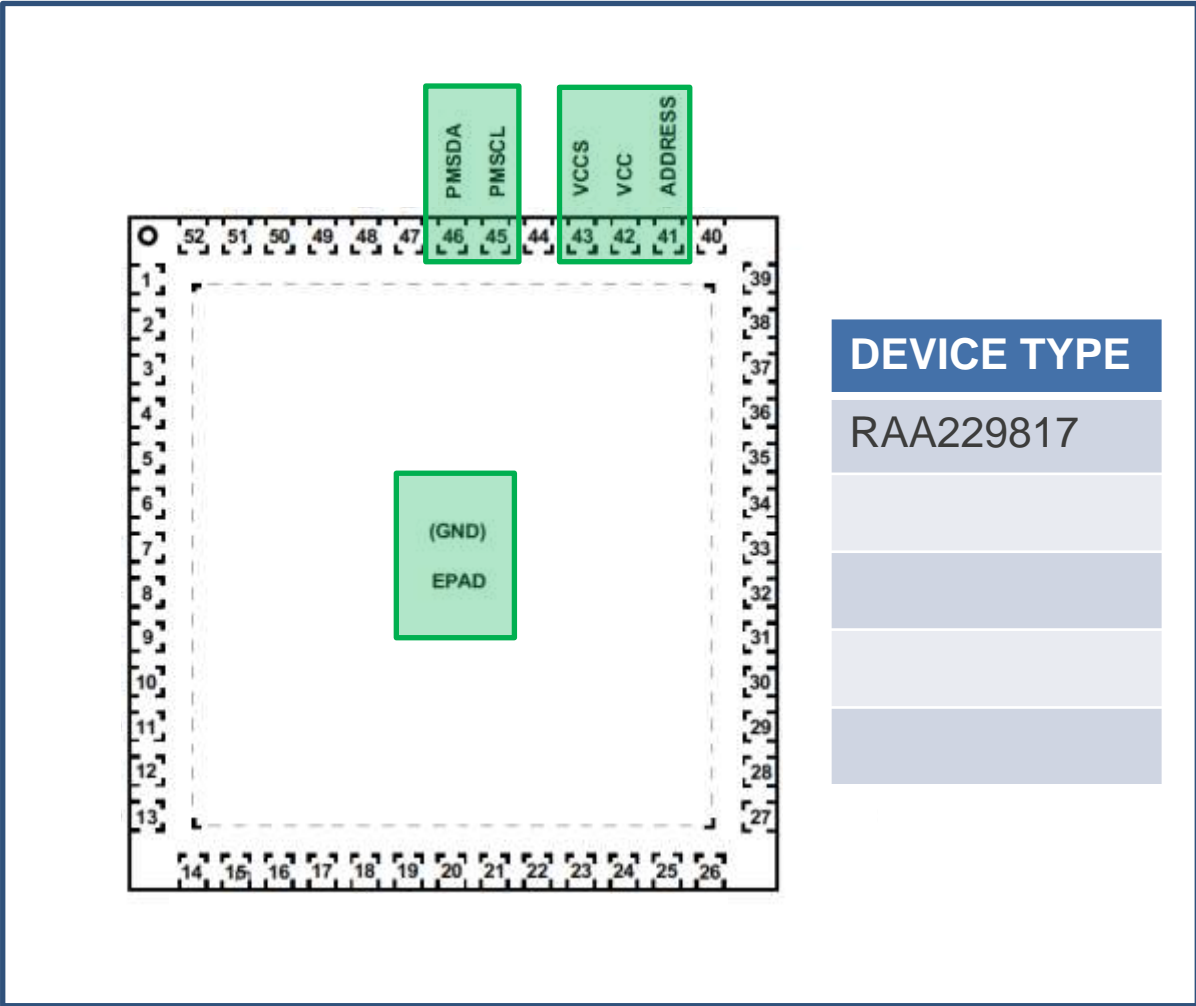
# 6x6 (52 LD) Package Characteristics

## 6x6, 52L QFN package



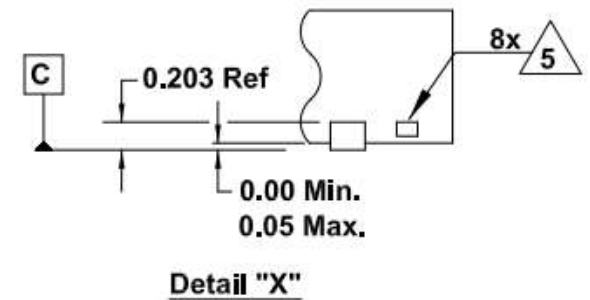
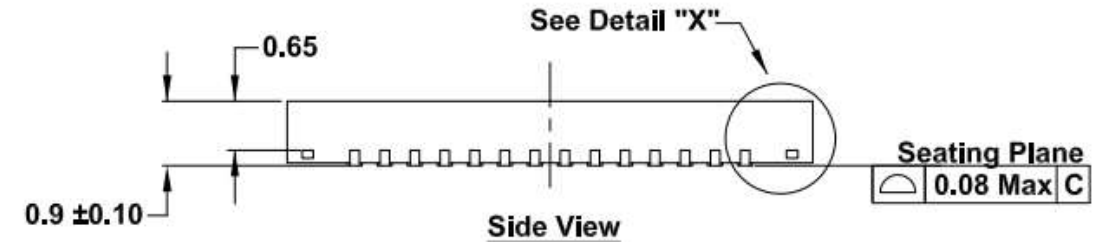
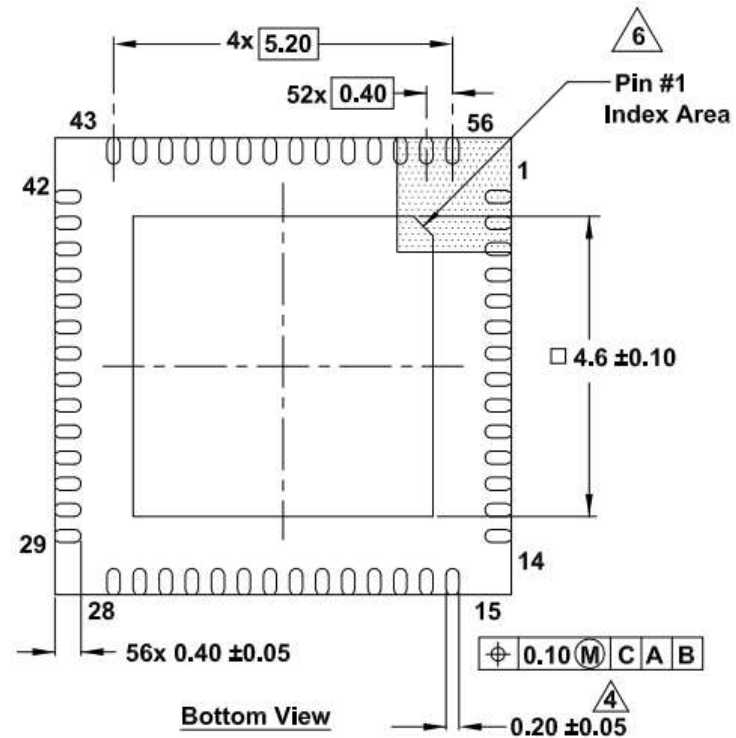
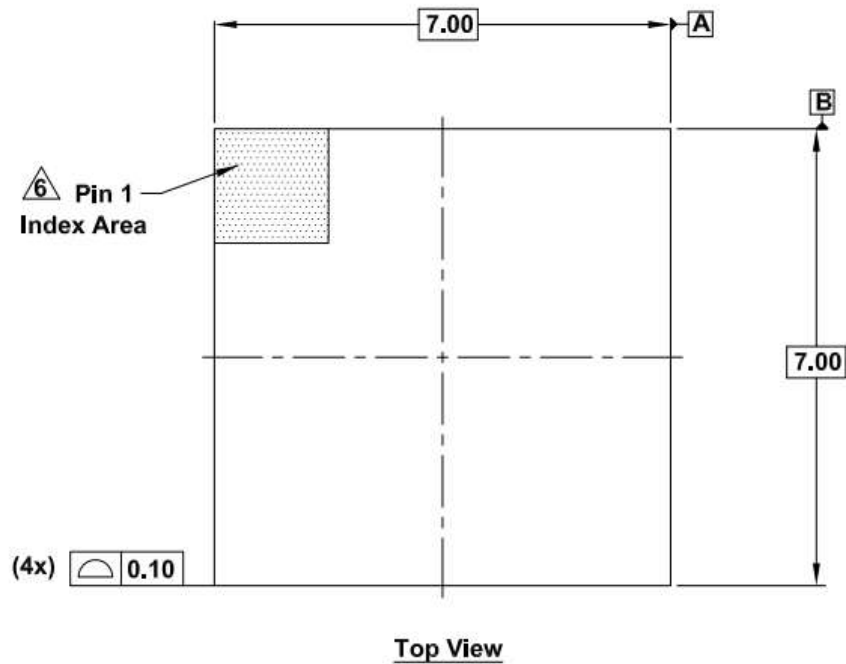
# 6x6 (52 LD) Pinouts and Devices

6x6, 52L QFN package



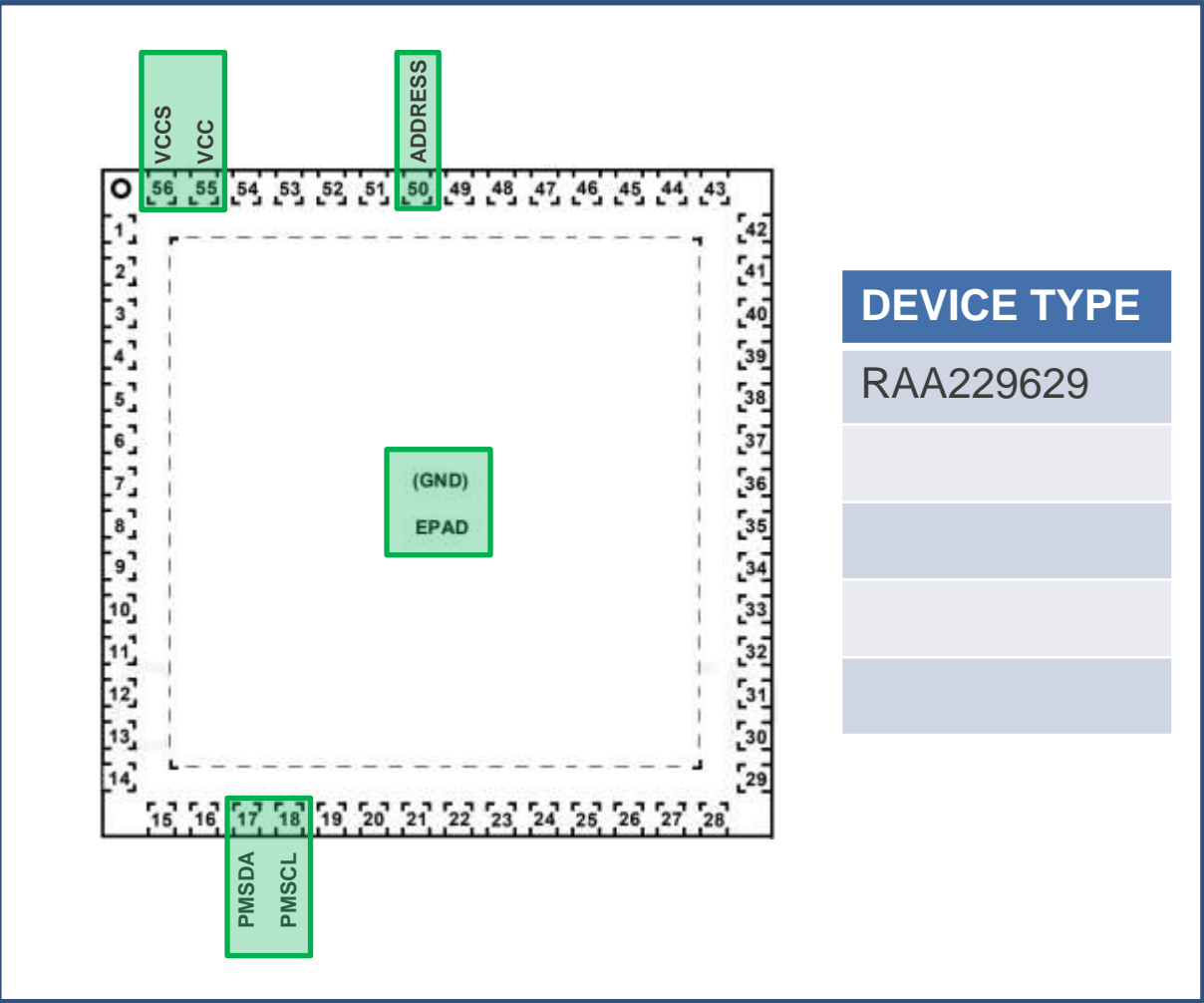
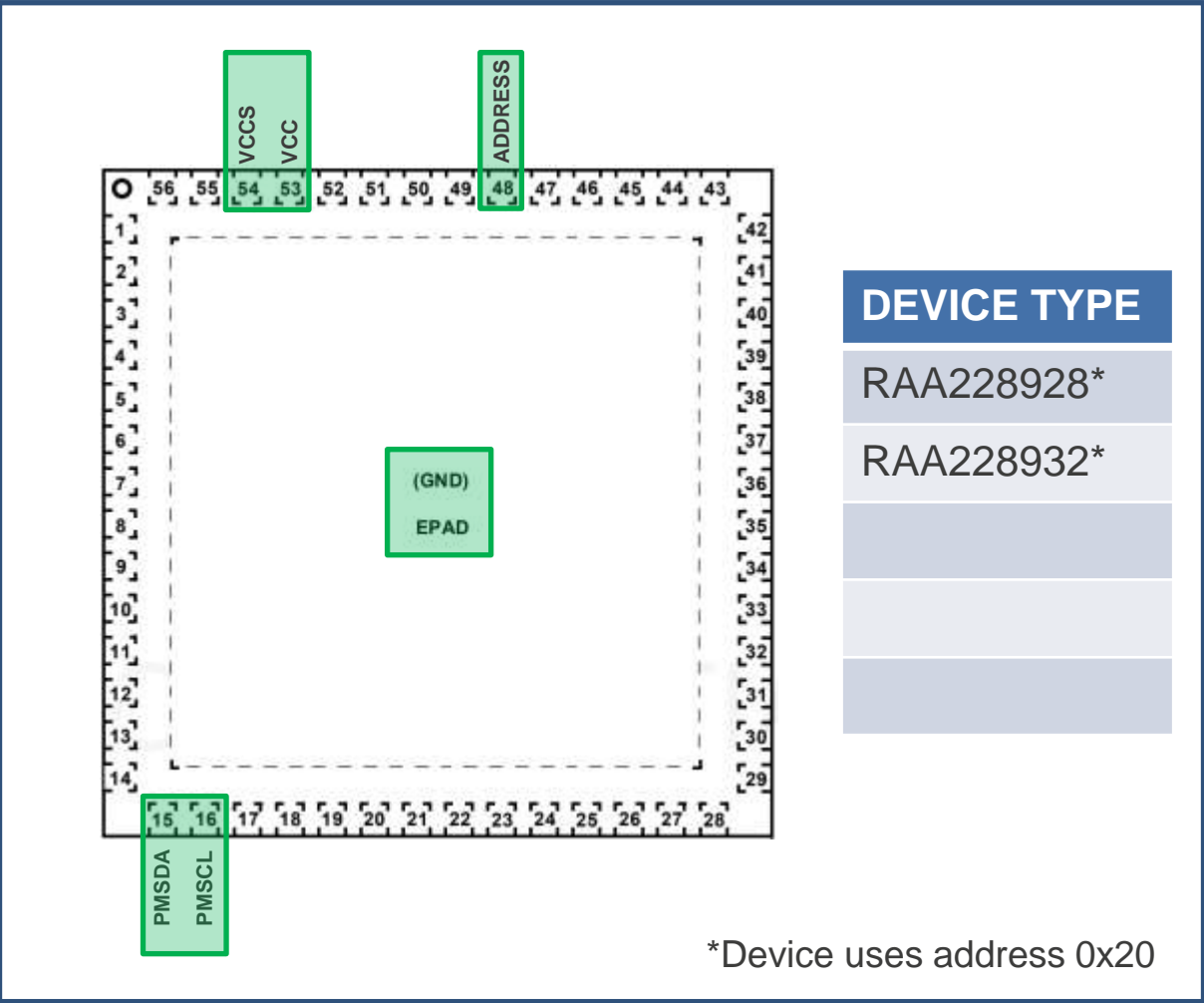
# 7x7 Package Characteristics

## 7x7, 56L QFN package



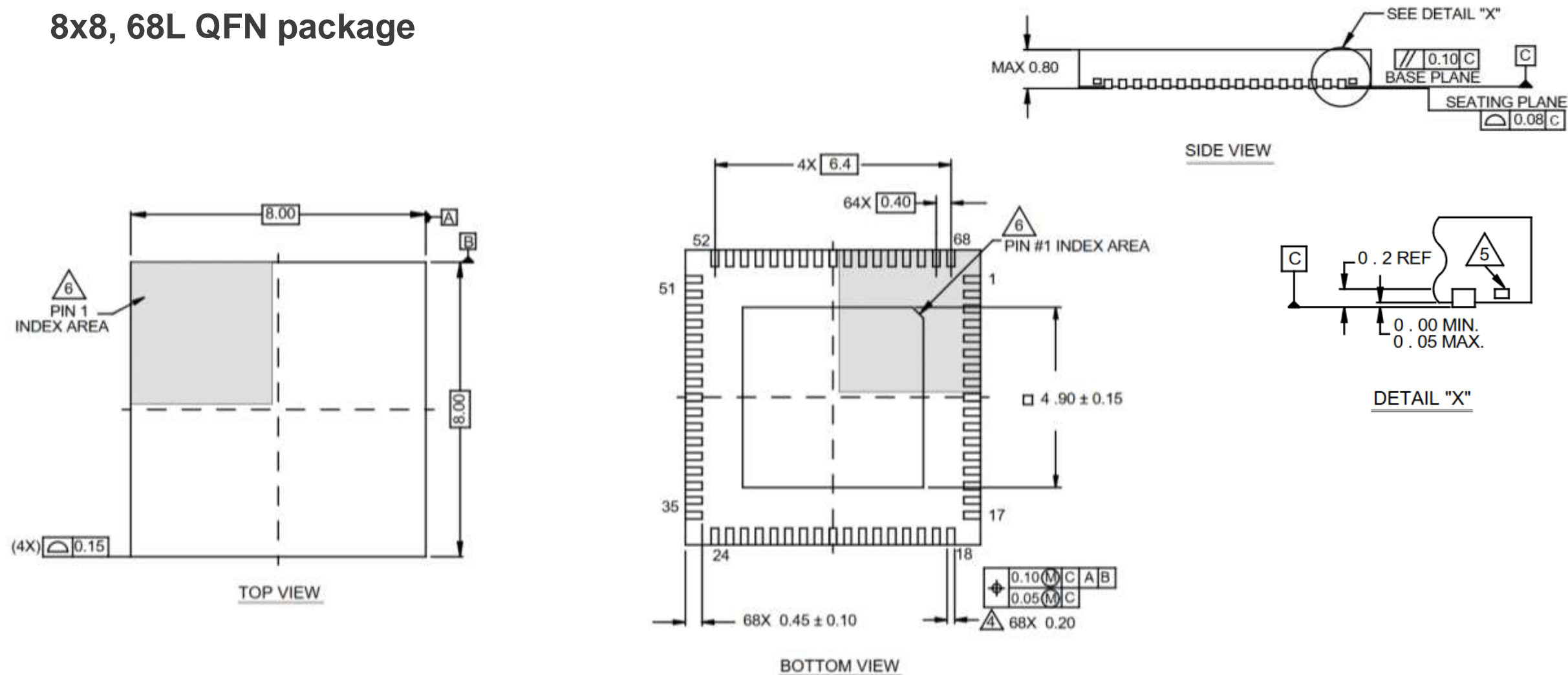
# 7x7 Pinouts and Devices

7x7, 56L QFN package



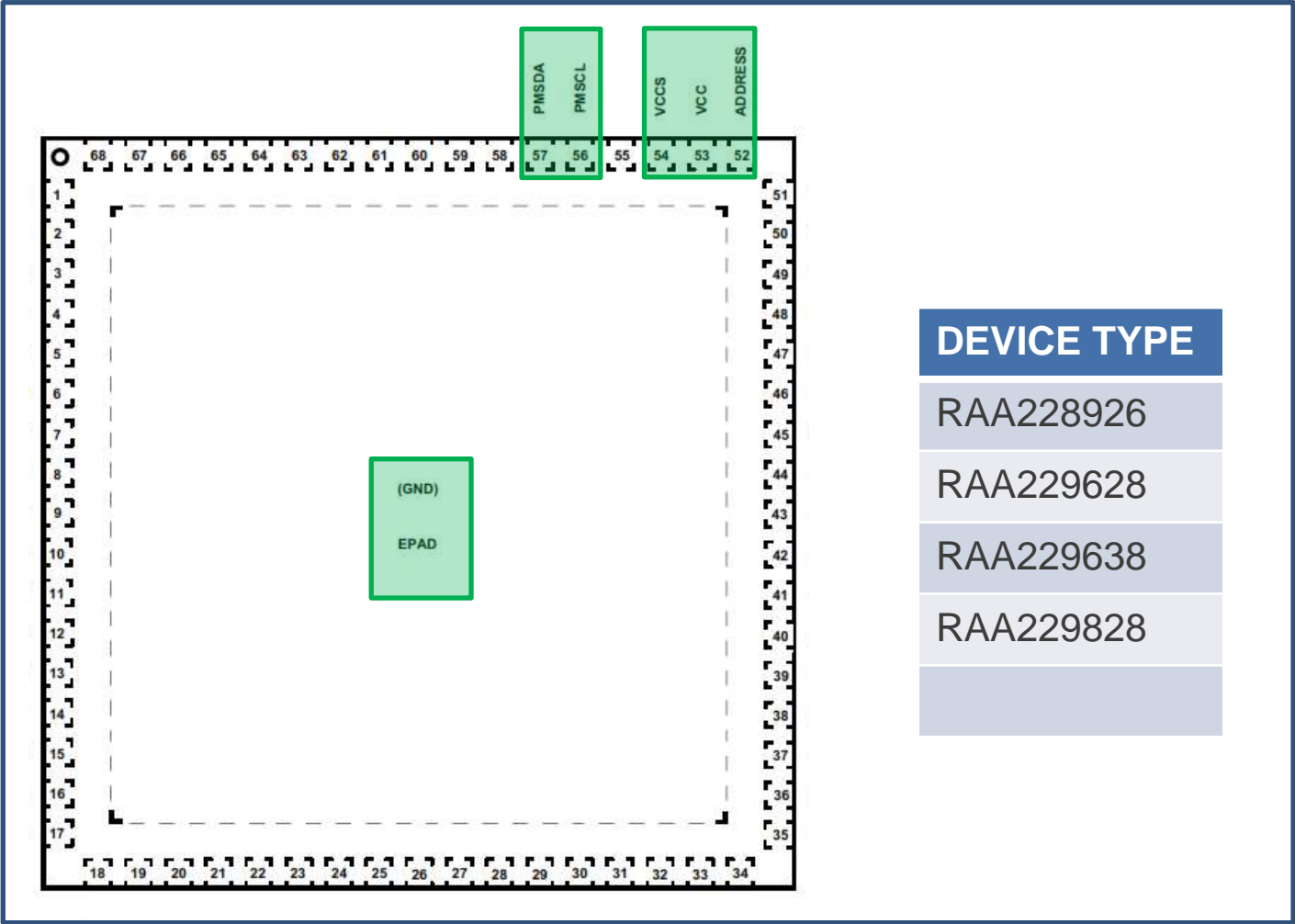
# 8x8 Package Characteristics

## 8x8, 68L QFN package



# 8x8 Pinouts and Devices (1)

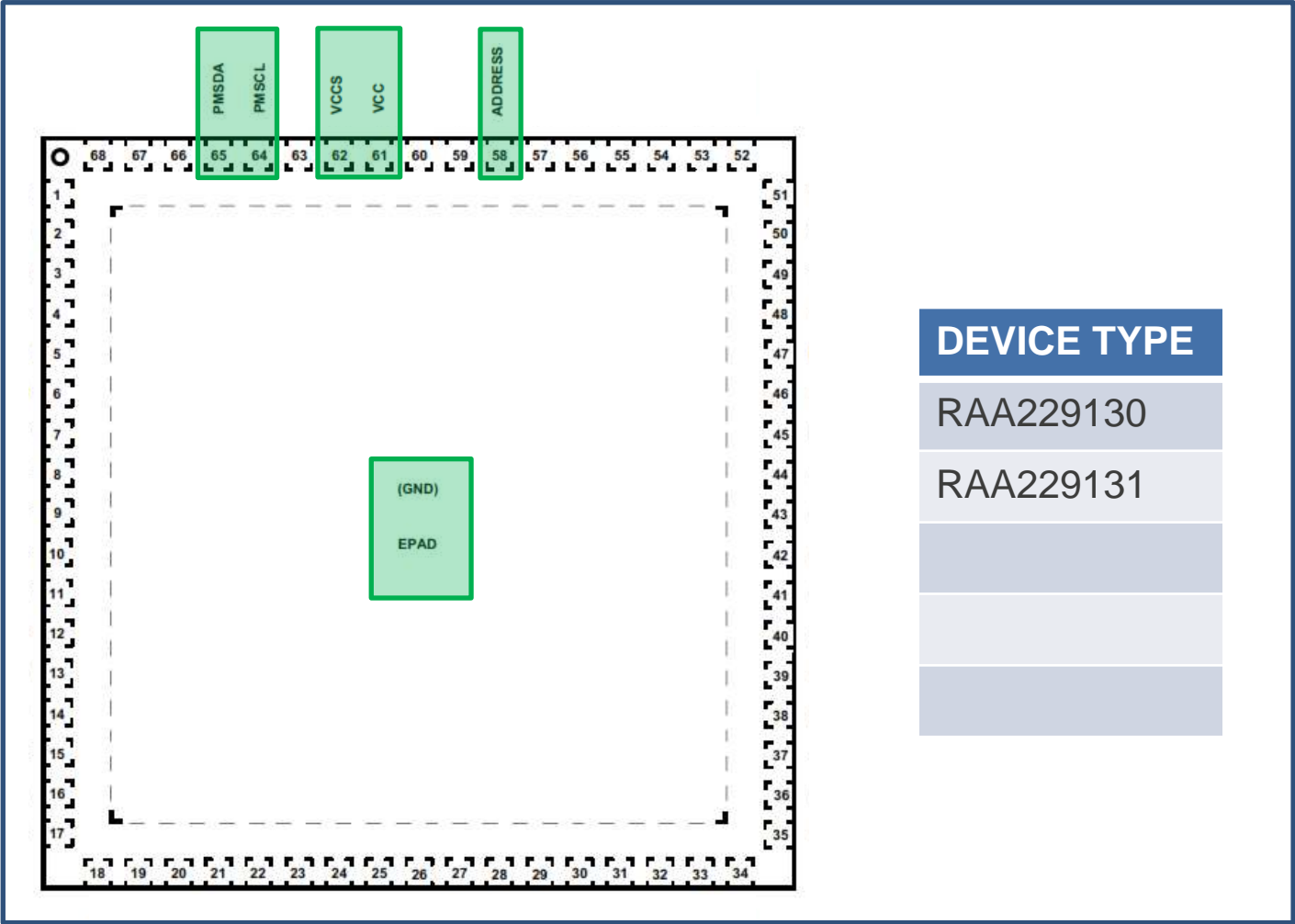
8x8, 68L QFN package



DEVICE TYPE
RAA228926
RAA229628
RAA229638
RAA229828

# 8x8 Pinouts and Devices (2)

8x8, 68L QFN package



DEVICE TYPE
RAA229130
RAA229131



## SECTION 2: PROGRAMMING DEVICES

# PROGRAMMING ALGORITHM OVERVIEW

## 1. (Optional) Disable packet capture.

- This step should be completed for parts with a configuration in RAM or NVM.
- For production programming, Step 1 can be ignored.

## 2. Determine number of NVM slots available.

## 3. Verify device and file versions.

- a. Read and parse header data from HEX file. Go to the Step 3a section for more information.
- b. Read IC\_DEVICE\_ID from device. Verify the value matches the Device Table and HEX file.
- c. Read the Legacy HEX File flag from device. Note compatibility mode for Step 3d.
- d. Read IC\_DEVICE\_REV from device. Verify that the device is compatible with the HEX file.

## 4. Read and parse one line from HEX file. Write to device.

- This step must be repeated for all configuration lines in the HEX file.

# PROGRAMMING ALGORITHM OVERVIEW (2)

## 5. Verify programming success.

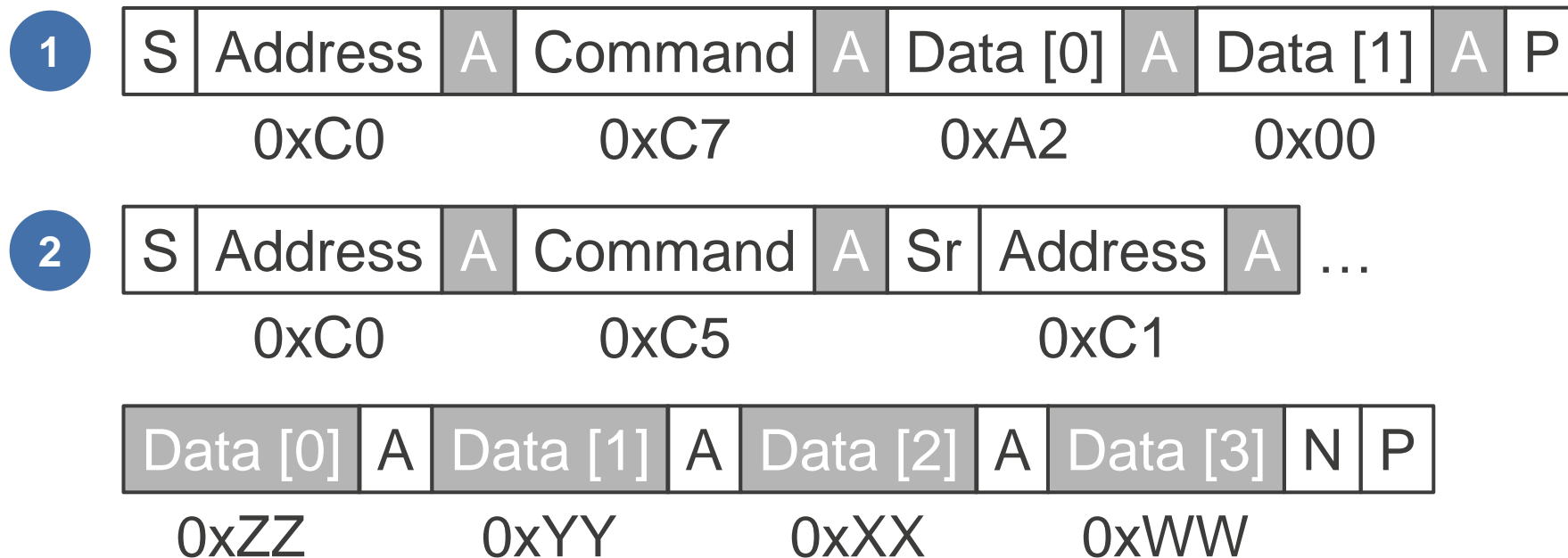
- a. Poll PROGRAMMER\_STATUS register until programming is complete.
- b. Read the BANK\_STATUS register to confirm all configurations were programmed successfully.

## 6. Verify CRC values.

# Step 1a – Retrieve Device Data

Perform Step 1 on parts that have a configuration in RAM or NVM.

Perform a read-modify-write to **clear bit 5** of DMA address **0x00A2** to disable packet capture during programming.



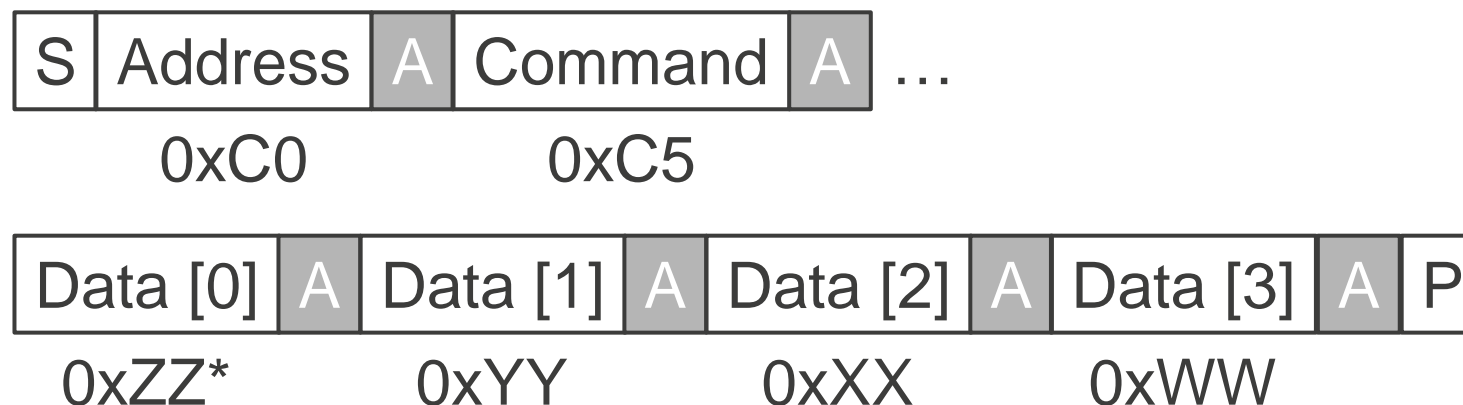
The device data is represented in the format 0xWWXXYYZZ.

Clear bit 5 in byte 0xZZ of this data for use in Step 1b.

# Step 1b – Disable Packet Capture

Perform Step 1 on parts that have a configuration in RAM or NVM.

Write the modified data from Step 1a to disable packet capture during programming.



Byte 0xZZ of this data was modified in Step 1a.

The remaining data is unchanged from the initial read.

# Step 1c – Apply Packet Capture Changes

Perform Step 1 on parts that have a configuration in RAM or NVM.

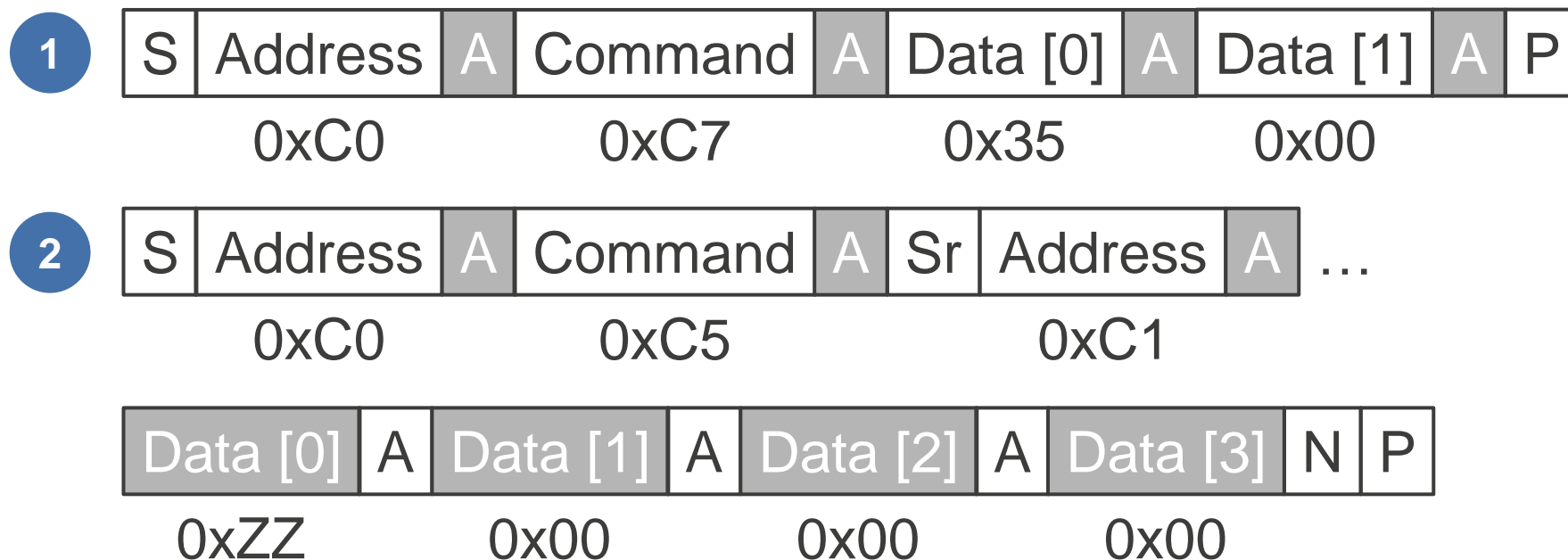
Send the following command to apply the packet capture changes.

S	Address	A	Command	A	Data [0]	A	Data [1]	A	P
	0xC0		0xE7		0x02		0x00		

## Step 2 – Retrieve NVM Counter Data

Read the number of available NVM configuration saves through DMA from address **0x0035**.

**Save this data for use in Step 3.**



The number of available NVM configuration saves takes the format 0xWWXXYYZZ.

This number will be a value from 0 to 28.

# Step 3a – Parse Header in HEX File

- The first 4 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 and IC\_DEVICE\_REV information.
  - IC\_DEVICE\_ID in HEX file must match IC\_DEVICE\_ID read back from device (see Step 3b).



# Step 3a – Example HEX File

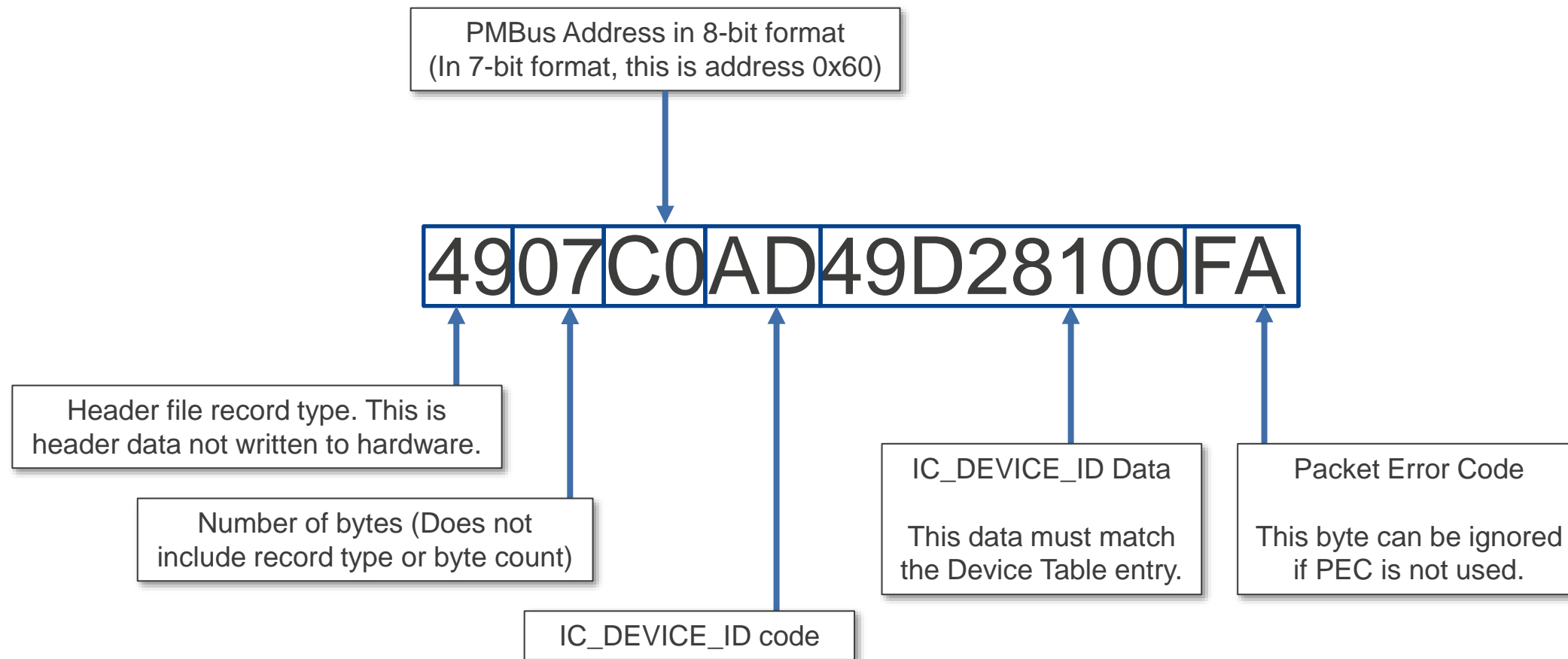
1	4907C0AD49D28100FA
2	4907C0AE0600000004F
3	490AC001352E342E32393742
4	490BC0020000017AEF32FA50F2
5	0005C0C77E0045
6	0007C0C50000000028
7	0005C0C77F0050
8	0007C0C50000000028
9	0005C0C77D007A
10	0007C0C500171100DE
...	
731	0007C0C6000000008E
732	0007C0C6000000008E
733	0007C0C6000000008E
734	0007C0C6000000008E
735	0007C0C6361A040022
736	0007C0C6361A040022
737	0007C0C60F7F338033
738	0005C0E70800DA

Parse first two lines and verify the HEX file was generated for the device being programmed (see steps 3b and 3c).

- Line 1 = IC\_DEVICE\_ID
- Line 2 = IC\_DEVICE\_REV

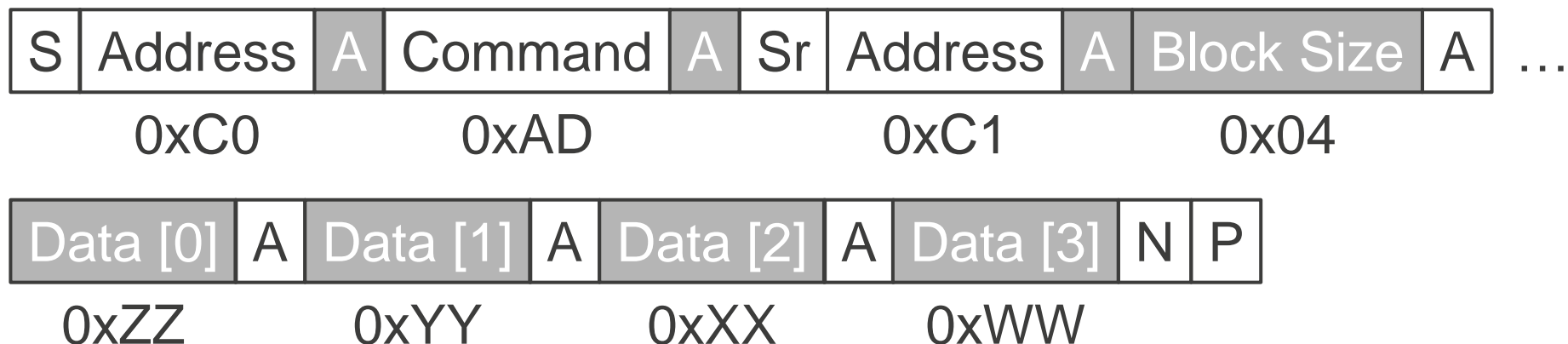
Value contained in file	Record Type	Command Code
IC_DEVICE_ID	0x49	0xAD
IC_DEVICE_REV	0x49	0xAE

# Step 3a – Example HEX File Header Decode



## Step 3b – Read IC\_DEVICE\_ID from Device

Read IC\_DEVICE\_ID information from the device using command code **0xAD**.



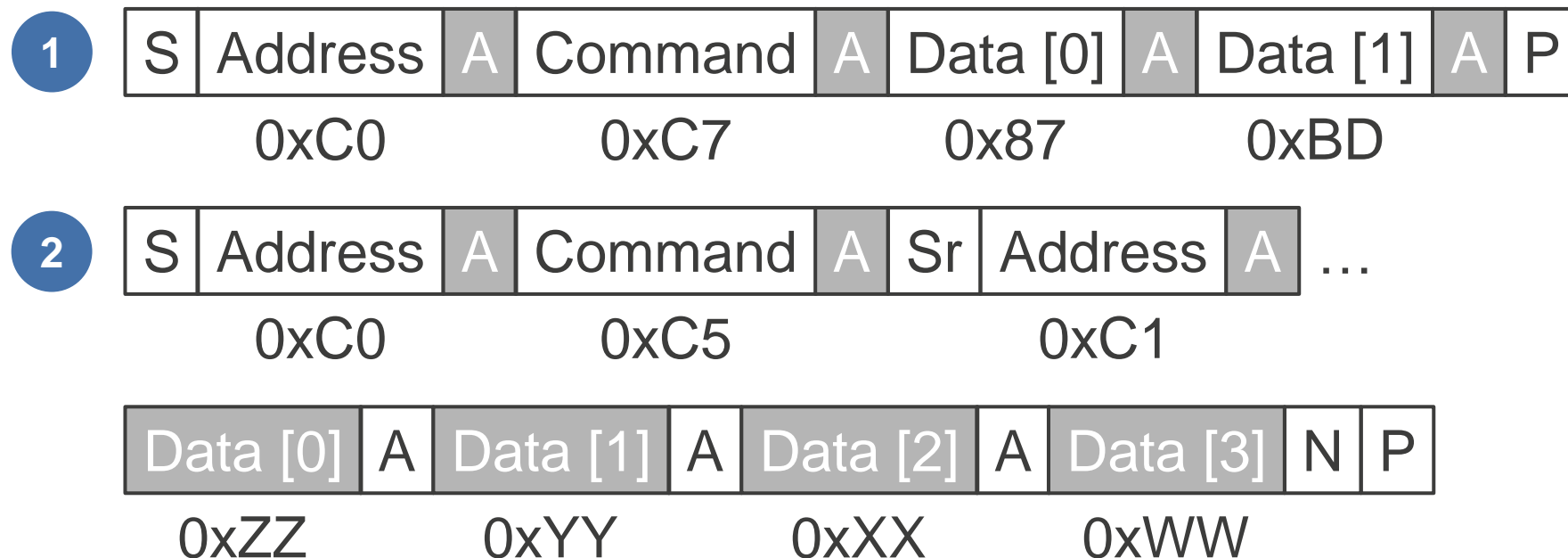
The device returns IC\_DEVICE\_ID, represented in the format 0xWWXXYYZZ.

This value must match the data shown in the Device Table.

**Programming will fail if this data does not match.**

## Step 3c – Retrieve Legacy HEX File Flag

Read the Legacy HEX File compatibility status through DMA from address **0xBD87** to determine the type of HEX file required for programming.



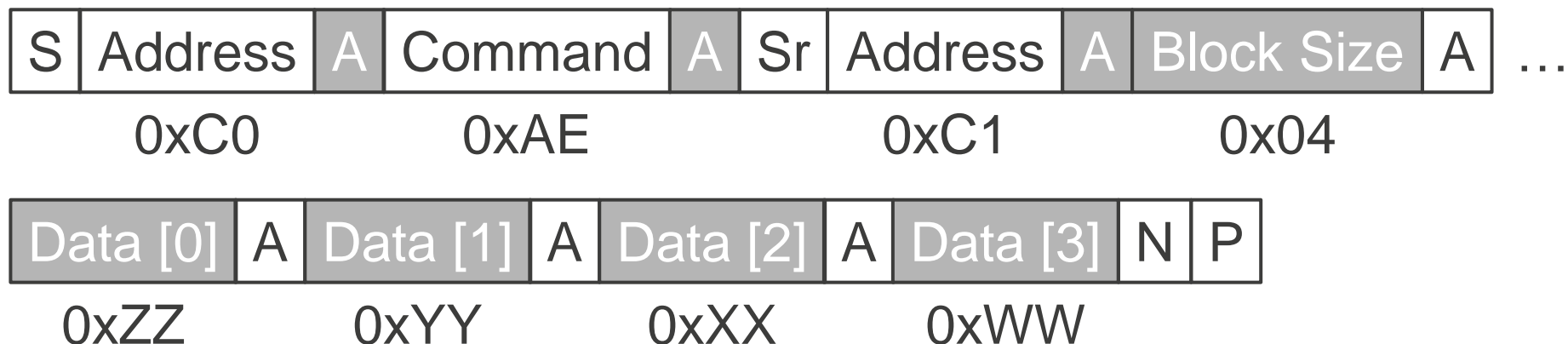
The device data is represented in the format 0xWWXXYYZZ.

If device readback is 0x00000000 **AND** the counter from Step 2 is < 28, Legacy HEX is required.

If device readback is 0xFFFFE0001 **OR** the counter from Step 2 is 28, Production HEX is required.

## Step 3d – Read IC\_DEVICE\_REV from Device

Read IC\_DEVICE\_REV information from the device using command code **0xAE**.



The device returns IC\_DEVICE\_REV, represented in the format WW.XX.YY.ZZ.

**The device revision number must be 6.0.0.0 or greater.**

If not, contact Renesas for support.

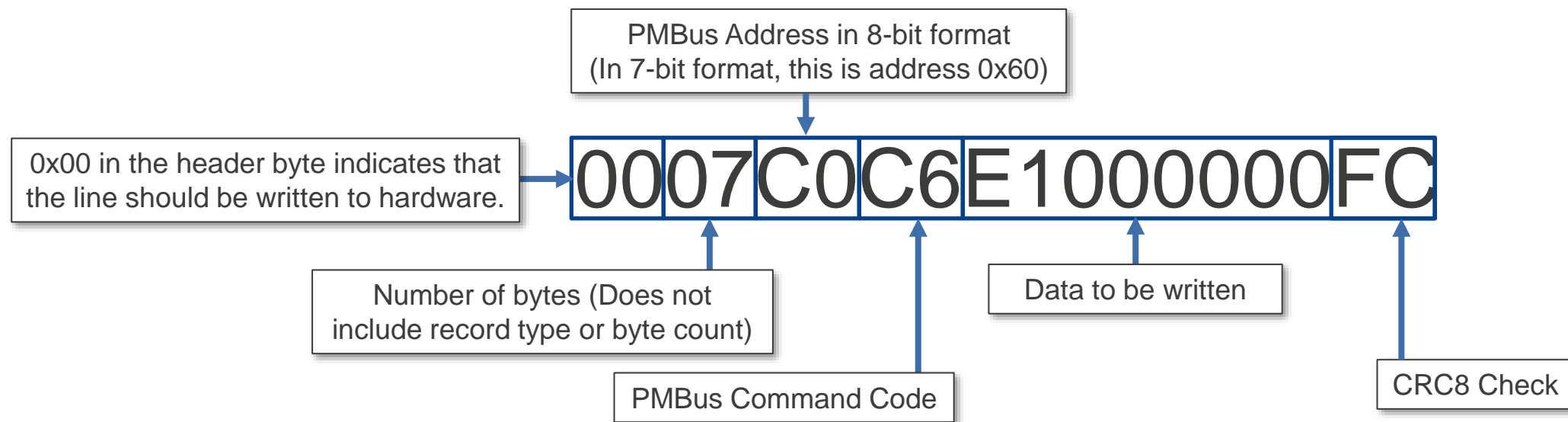
## Step 3e – Compare data to HEX file

- The IC\_DEVICE\_ID value read from the device must match the IC\_DEVICE\_ID value from the HEX file. If it does not match, the HEX file was generated for a different device and programming should be halted.
- The Legacy HEX File flag and IC\_DEVICE\_REV value should be used to determine if the HEX file was generated for the correct version of the device.
  - **If the Legacy HEX File flag readback in Step 3c requires Legacy HEX Files:**
    - IC\_DEVICE\_REV MSB from the device must be 0x06 or greater.
    - IC\_DEVICE\_REV MSB from the HEX file must be 0x00 or 0x01.
  - **If the Legacy HEX File flag readback in Step 3c requires Production HEX Files:**
    - IC\_DEVICE\_REV MSB from the device must be 0x06 or greater.
    - IC\_DEVICE\_REV MSB from the HEX file must be 0x06 or 0x07.

# Step 4 – 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 4 – Example HEX File

1	4907C0AD49D28100FA
2	4907C0AE0600000004F
3	490AC001352E342E32393742
4	490BC0020000017AEF1784C3F5
5	0005C0C77E0045
6	0007C0C500000000028
7	0005C0C77F0050
8	0007C0C500000000028
9	0005C0C77D007A
10	0007C0C500171200E1
...	
731	0007C0C60000000008E
732	0007C0C60000000008E
733	0007C0C60000000008E
734	0007C0C60000000008E
735	0007C0C6361A040022
736	0007C0C6361A040022
737	0007C0C60F7F338033
738	0005C0E70800DA

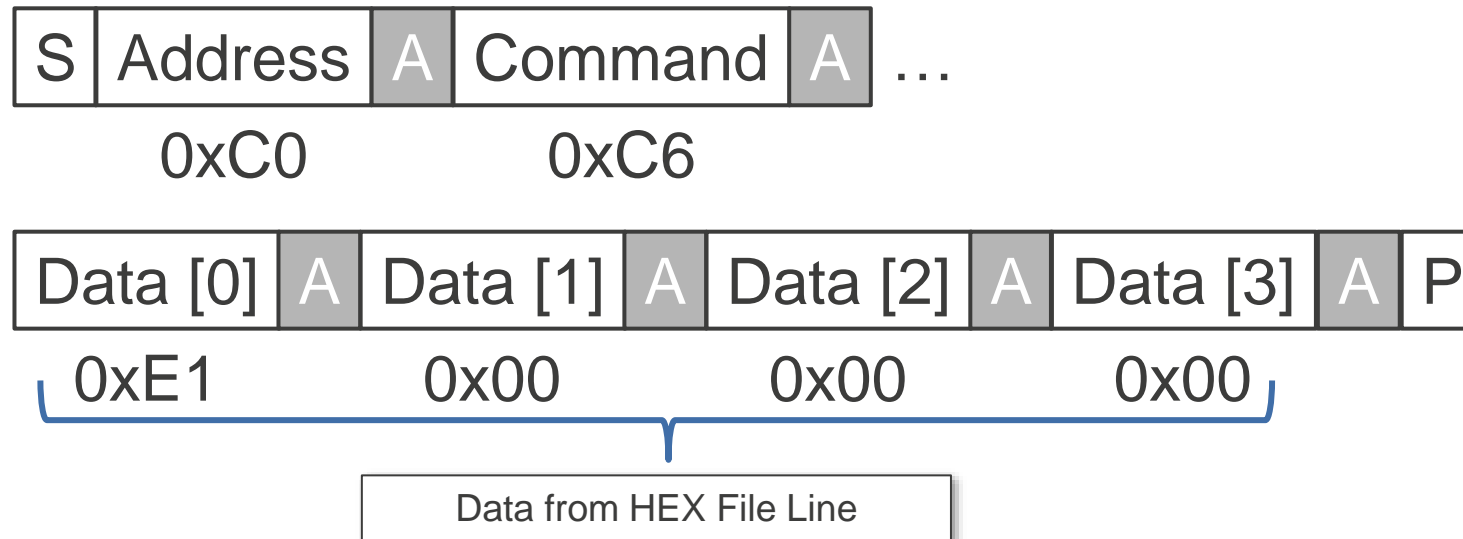
Step 4: Parse and write to device all lines in HEX file that do not start with 0x49

When last line has been written to device, Step 4 is complete.



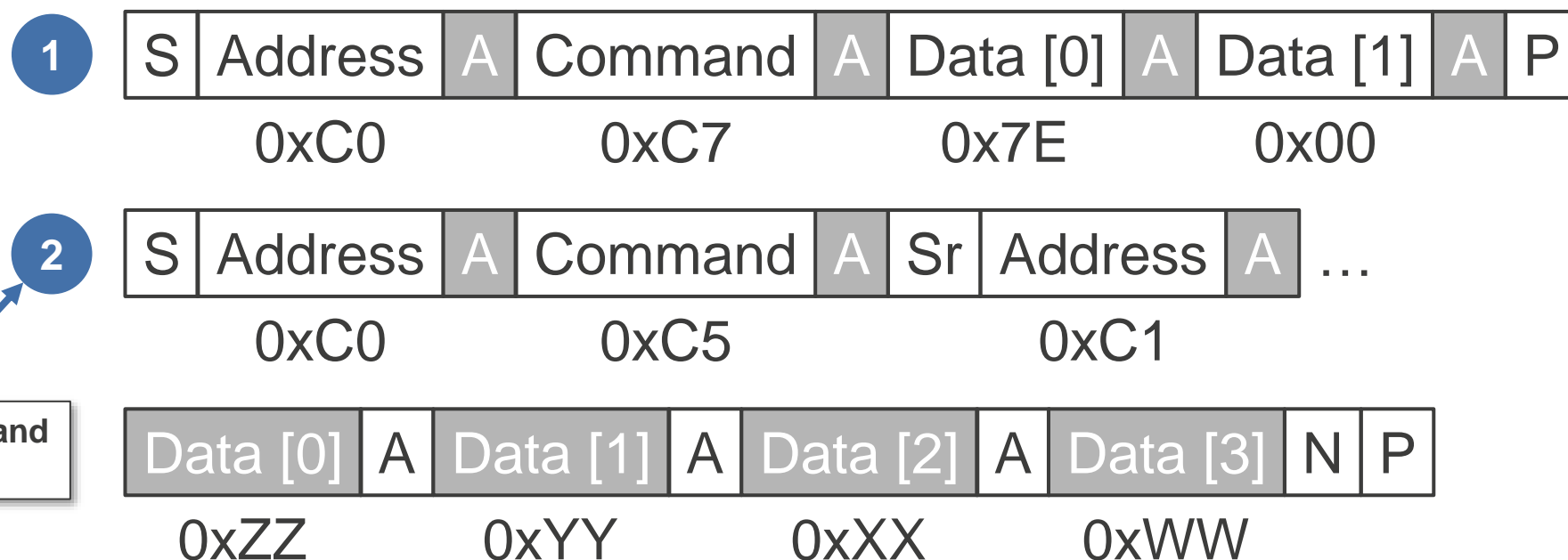
# Step 4 – Example Write of HEX File Data Line

Example Hex File Line Data: 0007C0C6E1000000FC



# Step 5a – Poll PROGRAMMER\_STATUS Register

Read the PROGRAMMER\_STATUS through DMA from address **0x007E** to determine when the programming routine is ready for verification.



The device data is represented in the format 0xWWXXYYZZ.

**If after a 2 second timeout bit 0 = 0, the part has failed programming.**

**See Step 5b for more details.**

# Step 5b – Programming Failure

Bits[5:0] from Step 5a

Sr	Address	A	Data [0]
	0xC1		0xZZ

To determine programming failure, take bits[5:0] from Step 5a and decode using the following...

bXX000001

If bit 5 is 1, programming has failed. Programming fails **after** OTP banks are consumed.

If this bit is 0, programming has failed.

If bit 4 is 1, the CRC check fails on the OTP memory. Programming fails **after** OTP banks are consumed.

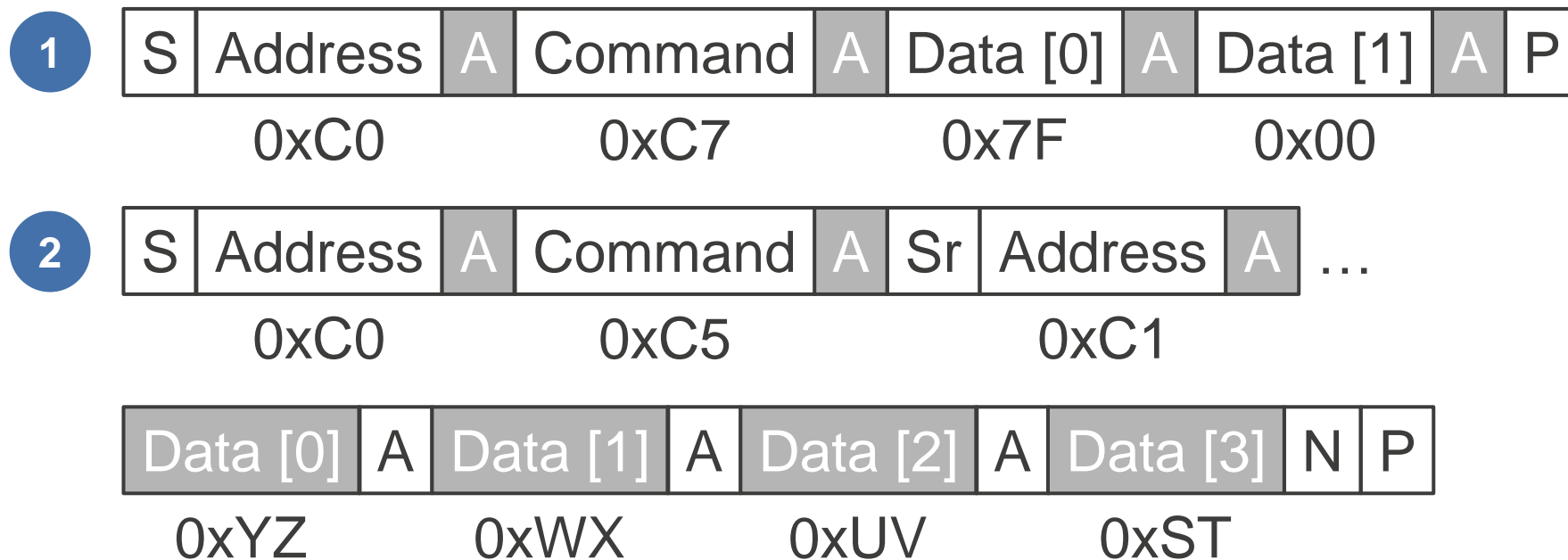
If bit 1 is 1, programming has failed.

If bit 3 is 1, a CRC mismatch exists within the configuration data. Programming fails before OTP banks are consumed.

If bit 2 is 1, the HEX file contains more configurations than are available.

## Step 5c – Read BANK\_STATUS Register

Read the BANK\_STATUS through DMA from address **0x007F** to retrieve status information for programming verification.



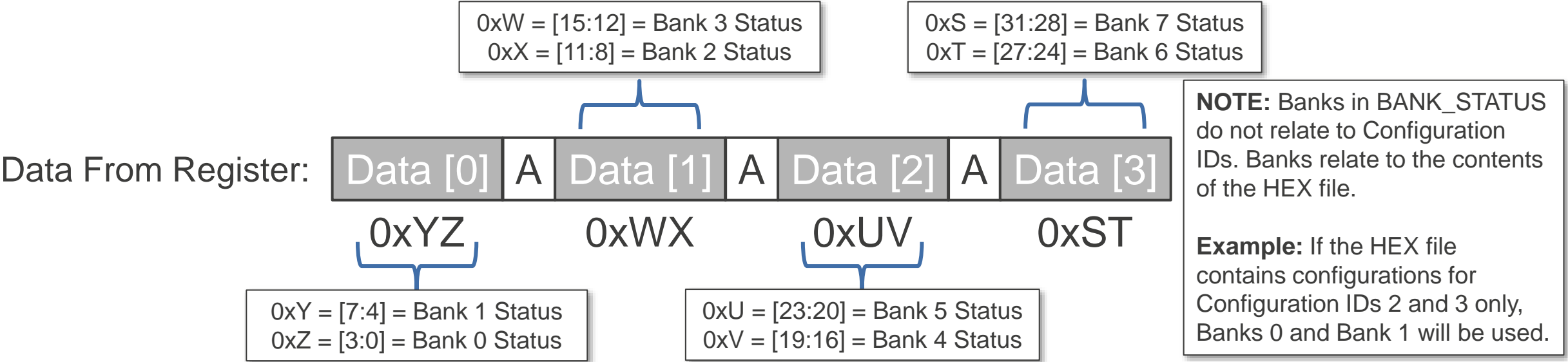
BANK\_STATUS is represented as 0xSTUVWXYZ, where 0xS is Bank 7 and 0xZ is Bank 0.

**See the next page for more information on BANK\_STATUS bits.**

# Step 5d – Read BANK\_STATUS Register

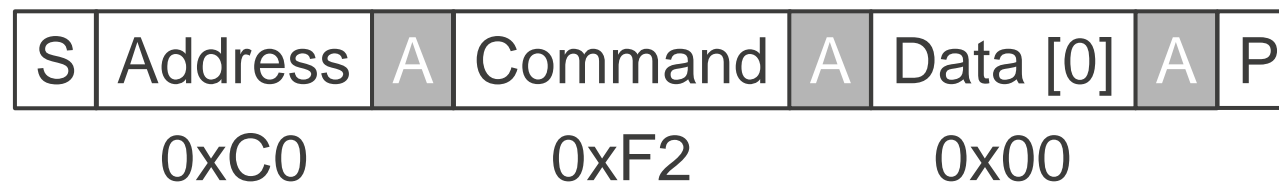
Bank status bits will correspond to the table below:

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 6a – Use RESTORE\_CFG Command

After completing Step 5, use the RESTORE\_CFG command code in the format shown below to load the new configuration. Do **not** use this command while the device is regulating.



Bits [3:0] are the selected configuration.

Configuration ID 0 = 0x00

Configuration ID 1 = 0x01

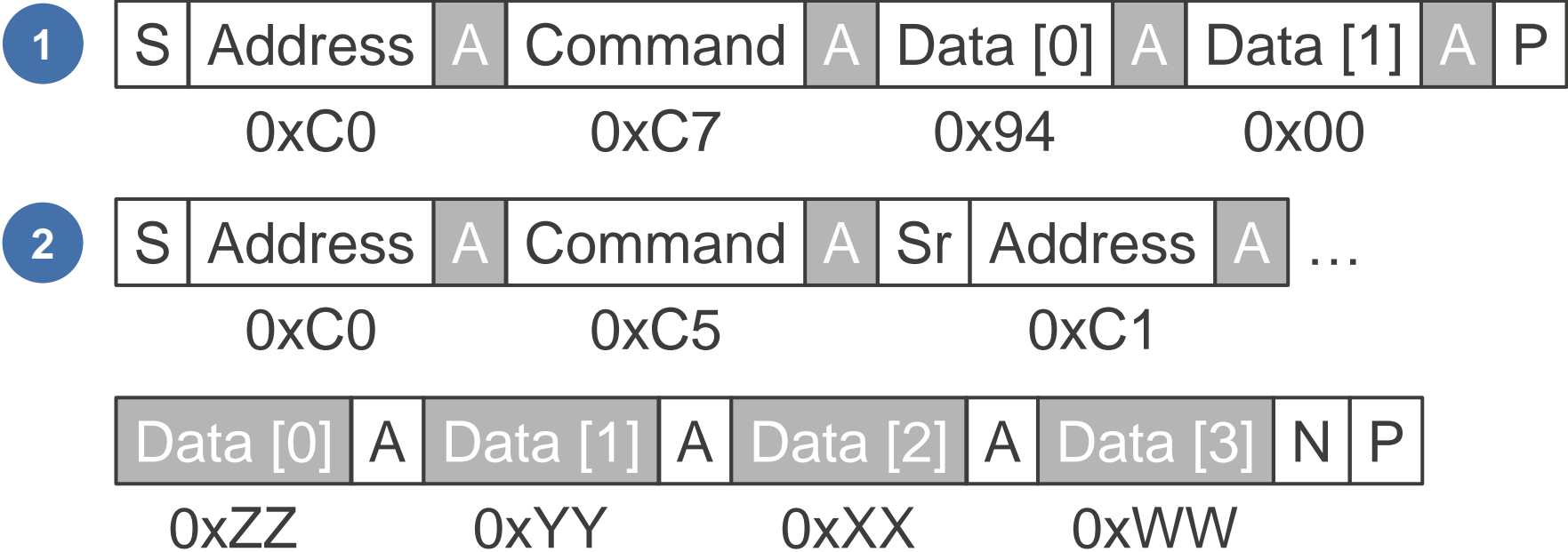
...

Configuration ID 14 = 0x0E

Configuration ID 15 = 0x0F

# Step 6b – Retrieve CRC Value

Read the CRC value through DMA from address **0x0094** to verify the configuration saved at the selected Configuration ID.

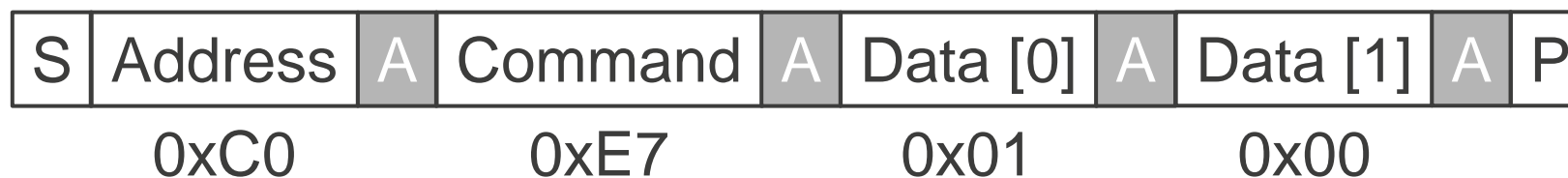


The device data is represented in the format 0xWWXXYYZZ.

If there is no configuration in the selected slot, the device will return the currently loaded CRC value.

# Algorithm Completion

- After successfully completing programming, 3.3V VCC can be powered down **or** send the following command to continue operating the part.



- Depending on the configurations in use, re-enabling packet capture manually may be required if VCC is not cycled.



# SECTION 3: HEX FILE CRC INFORMATION

# Production HEX Files – Total Configurations

$$\text{Total number of configurations} = \frac{\# \text{ of lines in HEX file} - 12}{363}$$

1	4907C0AD49D28100FA
2	4907C0AE060000004F
3	490AC001352E342E32393742
4	490BC0020000017AEF1784C3F5
5	0005C0C77E0045
6	0007C0C50000000028
...	
731	0007C0C60000000008E
732	0007C0C60000000008E
733	0007C0C60000000008E
734	0007C0C60000000008E
735	0007C0C6361A040022
736	0007C0C6361A040022
737	0007C0C60F7F338033
738	0005C0E70800DA

In the example HEX file, there are 738 lines.

$$\text{Total number of configurations} = \frac{\# \text{ of lines in HEX file} - 12}{738}$$

$$\text{Total number of configurations} = \frac{738 - 12}{363}$$

$$\text{Total number of configurations} = \frac{726}{363}$$

$$\text{Total number of configurations} = 2$$

The example HEX file contains **2 configurations**.

# Production HEX Files – Configuration IDs

ID Line Number =  $(N * 363) + 12$ ,  $N = 0, 1, \dots$ , # of configurations in file – 1

1	4907C0AD49D28100FA
2	4907C0AE060000004F
3	490AC001352E342E32393742
4	490BC0020000017AEF1784C3F5
5	0005C0C77E0045
6	0007C0C50000000028
...	
10	0007C0C500171200E1
11	0005C0C7001754
12	0007C0C600FF0000A5
13	0007C0C60816008474
14	0007C0C603000000B4
...	

In the example HEX file, there are 738 lines and 2 configurations.

$N = 0, 1, \dots$ , # of configurations in file – 1

Configuration ID line number =  $(N * 363) + 12$

For the 1<sup>st</sup> configuration in the file,  $N = 0$ .

Configuration ID line number =  $(0 * 363) + 12$

Configuration ID line number = 12

The 10<sup>th</sup> character in the line is the Configuration ID.

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

# Production HEX Files – Configuration CRCs

CRC line number =  $(N * 363) + 290$ ,  $N = 0, 1, \dots$ , # of configurations in file – 1

1	4907C0AD49D28100FA
2	4907C0AE060000004F
3	490AC001352E342E32393742
4	490BC0020000017AEF1784C3F5
5	0005C0C77E0045
6	0007C0C50000000028
...	
274	0007C0C60006A47DCB
275	0007C0C600060000F3
276	0007C0C68298CDE0C1
277	0007C0C606000000FA
278	0007C0C6000000008E
...	

In the example HEX file, there are 738 lines and 2 configurations.

$N = 0, 1, \dots$ , # of configurations in file – 1

CRC line number =  $(N * 363) + 290$

For the 1<sup>st</sup> configuration in the file,  $N = 0$ .

CRC line number =  $(0 * 363) + 290$

CRC line number = 290

The 1<sup>st</sup> configuration CRC is **0xE0CD9882**.

# Legacy HEX Files – Total Configurations

$$\text{Total number of configurations} = \frac{\# \text{ of lines in HEX file} - 12}{337}$$

1	4907C0AD49D28200C5
2	4907C0AE010000002D
3	490AC001352E342E323334C9
4	490BC002000001744B5D7A533A
5	0005C0C77E0045
6	0007C0C50000000028
...	
342	0007C0C60000000008E
343	0007C0C62200000006C
344	0007C0C60000000008E
345	0007C0C60000000008E
346	0007C0C60000000008E
347	0007C0C60000000008E
348	0007C0C69EB385ED31
349	0005C0E70800DA

In the example HEX file, there are 349 lines.

$$\text{Total number of configurations} = \frac{\# \text{ of lines in HEX file} - 12}{337}$$

$$\text{Total number of configurations} = \frac{349 - 12}{337}$$

$$\text{Total number of configurations} = \frac{337}{337}$$

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

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

# Legacy HEX Files – Configuration IDs

ID Line Number =  $(N * 337) + 12$ ,  $N = 0, 1, \dots$ , # of configurations in file – 1

1	4907C0AD49D28200C5
2	4907C0AE010000002D
3	490AC001352E342E323334C9
4	490BC002000001744B5D7A533A
5	0005C0C77E0045
6	0007C0C50000000028
...	
10	0007C0C50017010089
11	0005C0C7001754
12	0007C0C600FF0000A5
13	0007C0C648160084EF
14	0007C0C603000000B4
...	

In the example HEX file, there are 349 lines and 1 configuration.

$N = 0, 1, \dots$ , # of configurations in file – 1

Configuration ID line number =  $(N * 337) + 12$

For the 1<sup>st</sup> configuration in the file,  $N = 0$ .

Configuration ID line number =  $(0 * 337) + 12$

Configuration ID line number = 12

The 10<sup>th</sup> character in the line is the Configuration ID.

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

# Legacy HEX Files – Configuration CRCs

CRC line number =  $(N * 337) + 276$ ,  $N = 0, 1, \dots, \# \text{ of configurations in file} - 1$

1	4907C0AD49D28200C5
2	4907C0AE010000002D
3	490AC001352E342E323334C9
4	490BC002000001744B5D7A533A
5	0005C0C77E0045
6	0007C0C50000000028
...	
274	0007C0C60000006A485
275	0007C0C67D007D7D7D
276	0007C0C67CC7606C39
277	0007C0C606000000FA
278	0007C0C60000000008E
...	

In the example HEX file, there are 349 lines and 1 configuration.

$N = 0, 1, \dots, \# \text{ of configurations in file} - 1$

CRC line number =  $(N * 337) + 276$

For the 1<sup>st</sup> configuration in the file,  $N = 0$ .

CRC line number =  $(0 * 337) + 276$

CRC line number = 276

The 1<sup>st</sup> configuration CRC is **0x6C60C77C**.

# DMA COMMAND FORMAT REFERENCE



# Direct Memory Access (DMA) Command Codes

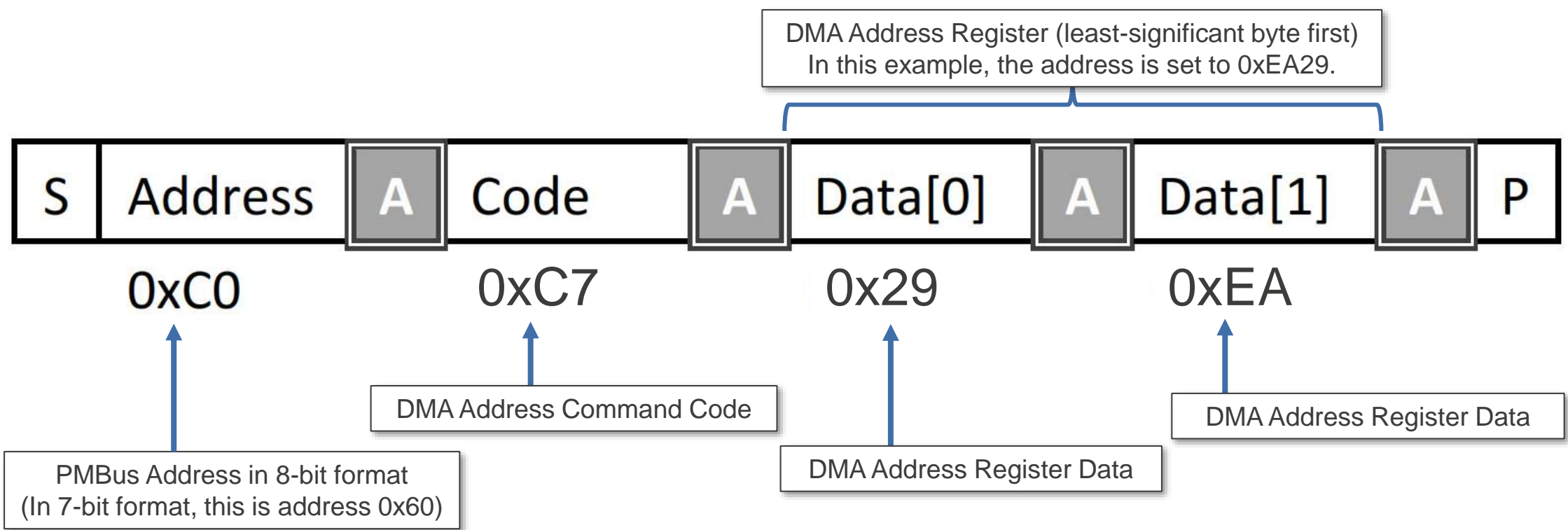
This section explains direct memory access (DMA) commands. DMA is completed through 3 command codes:

- **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.

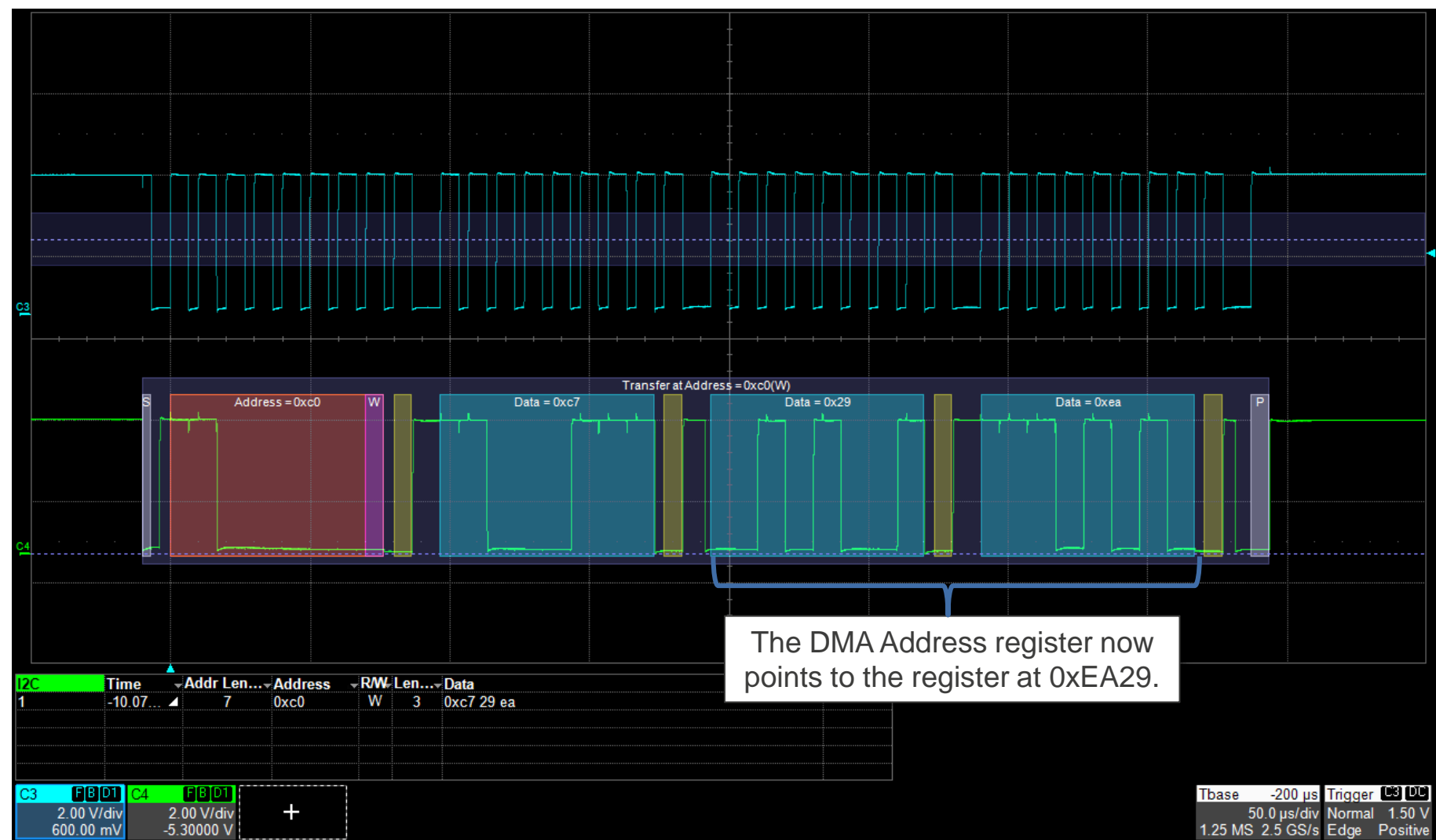
# DMA ADDRESS (0XC7)

# 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.

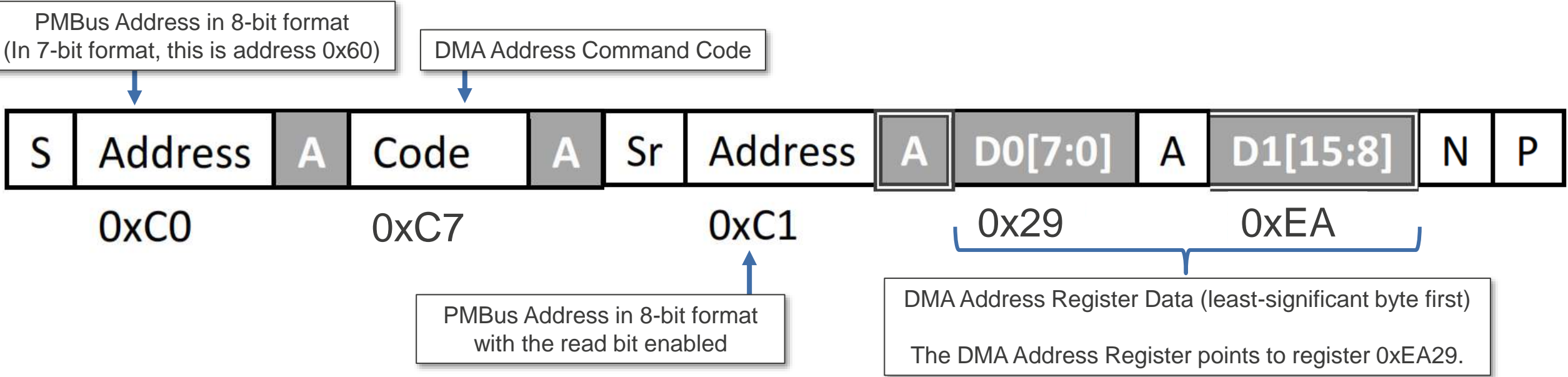


# DMA Address (0xC7) – Write Waveform

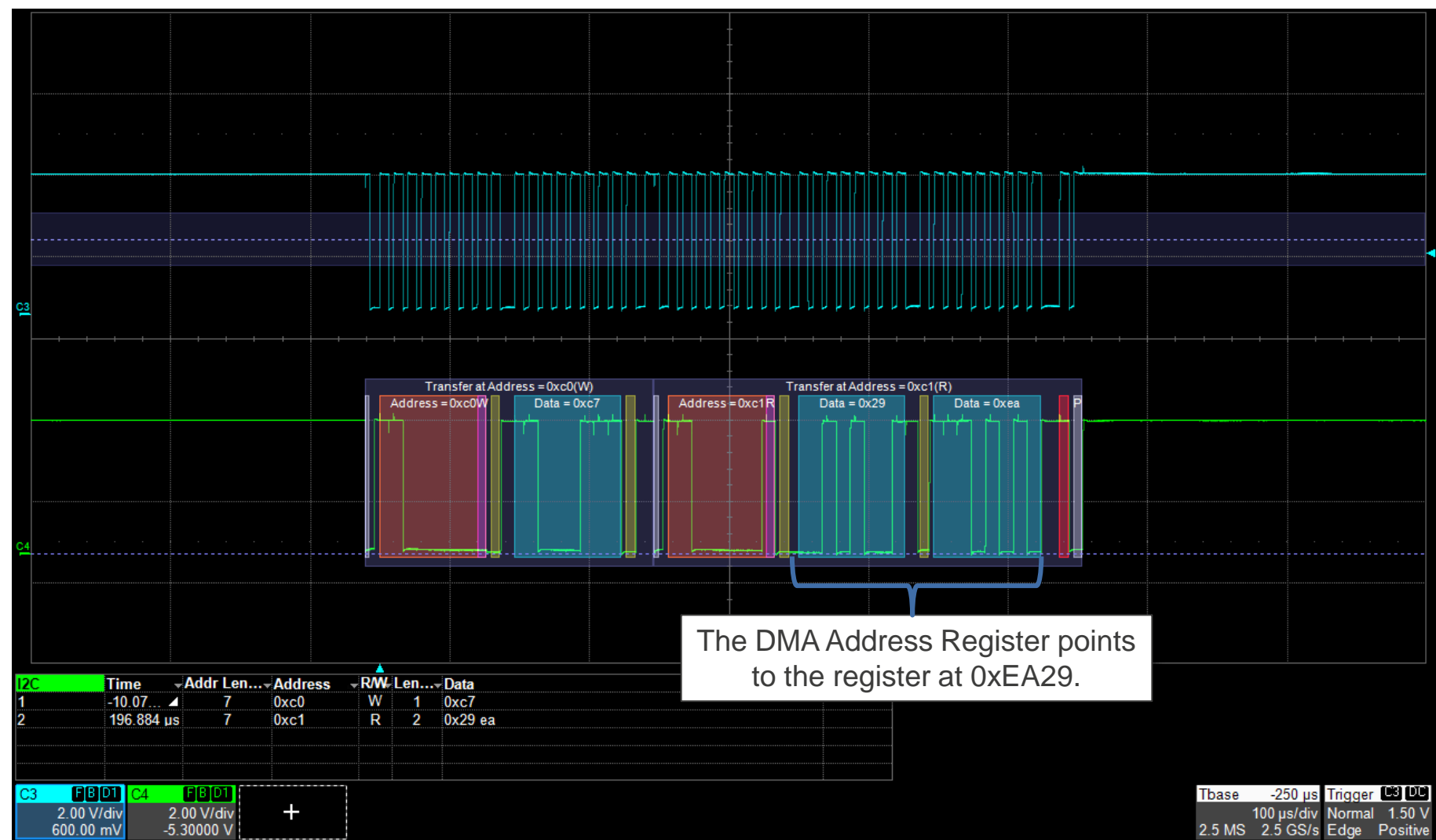


# DMA Address (0xC7) – Read

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.



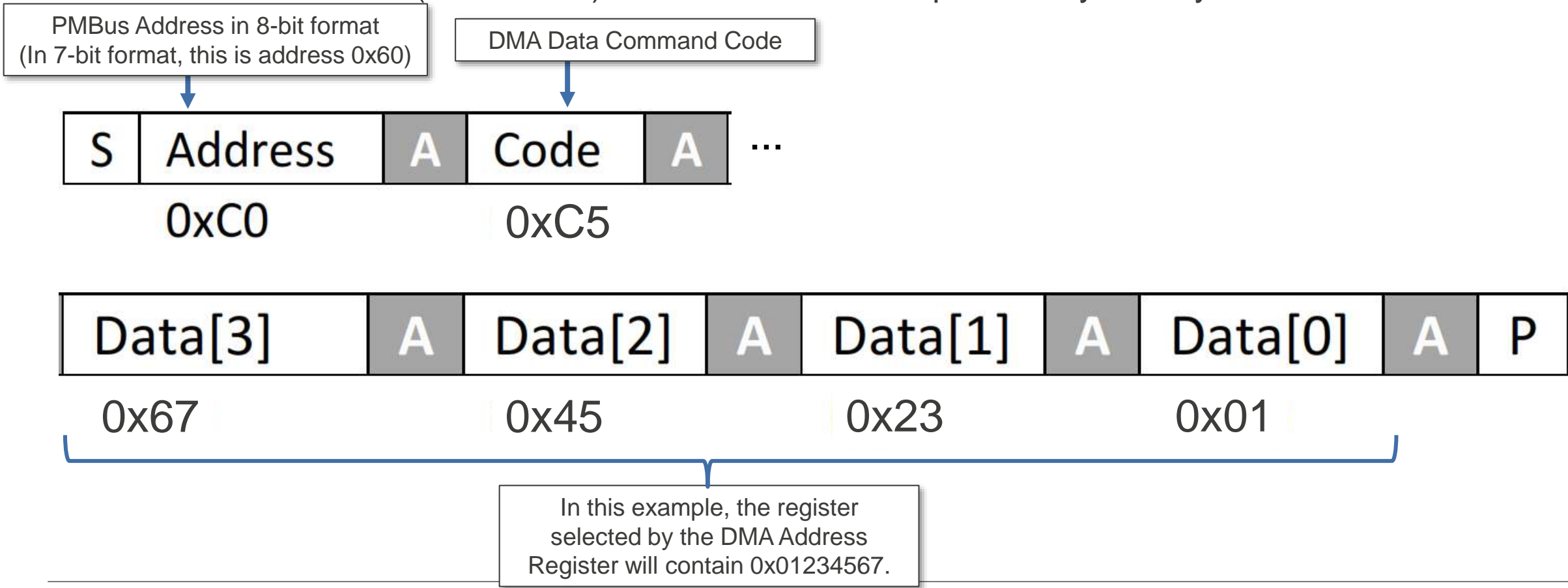
# DMA Address (0xC7) – Read Waveform



# DMA DATA (0XC5)

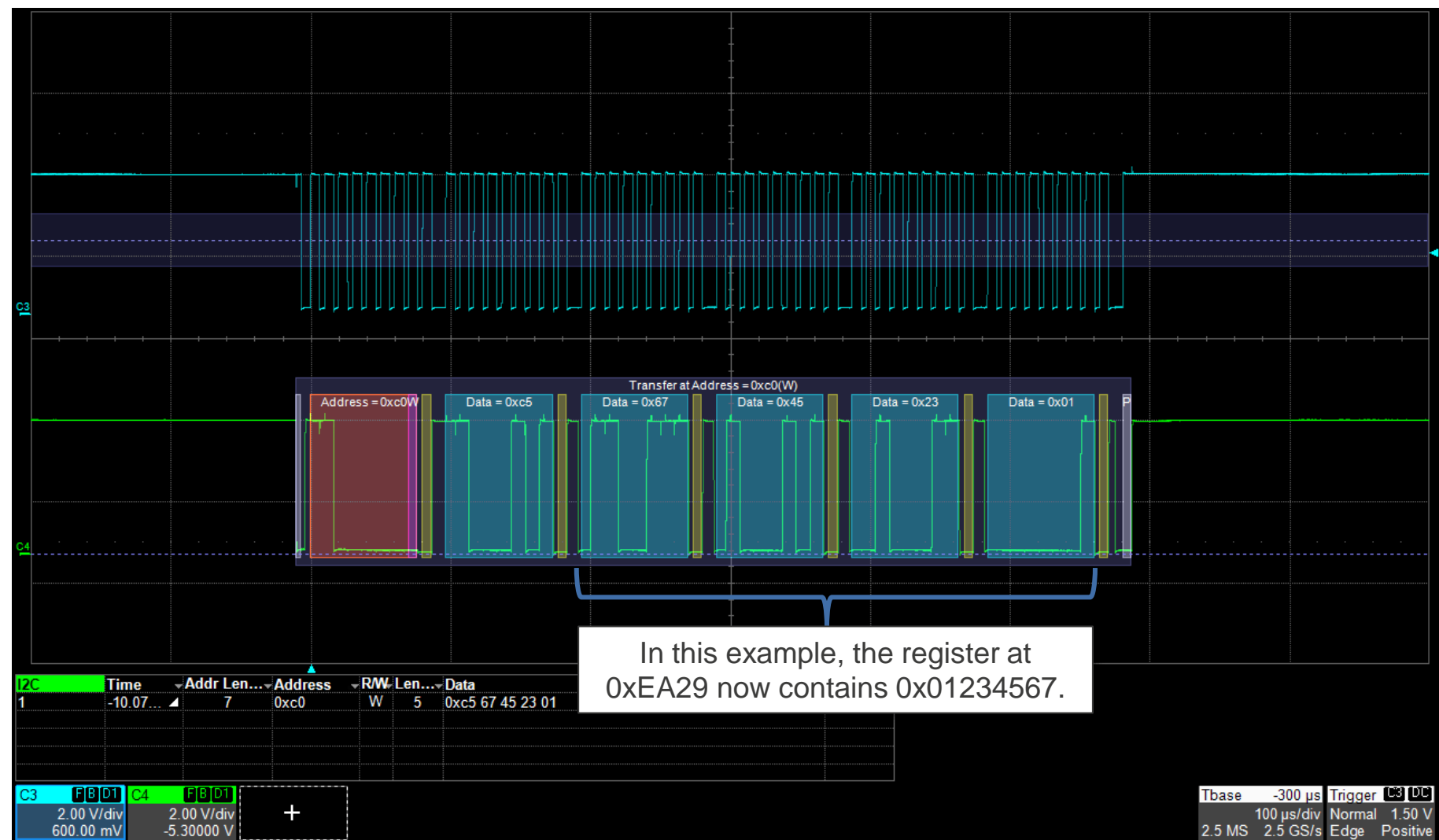
# DMA Data (0xC5) – Write

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.



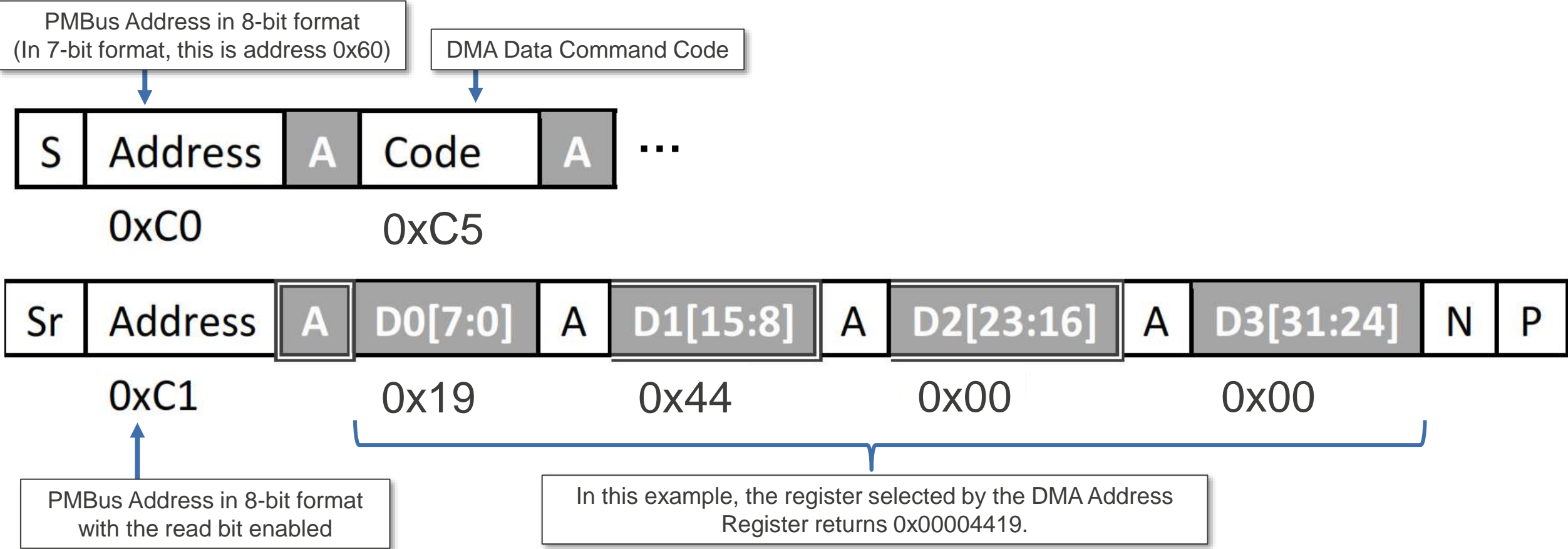


# DMA Data (0xC5) – Write Waveform

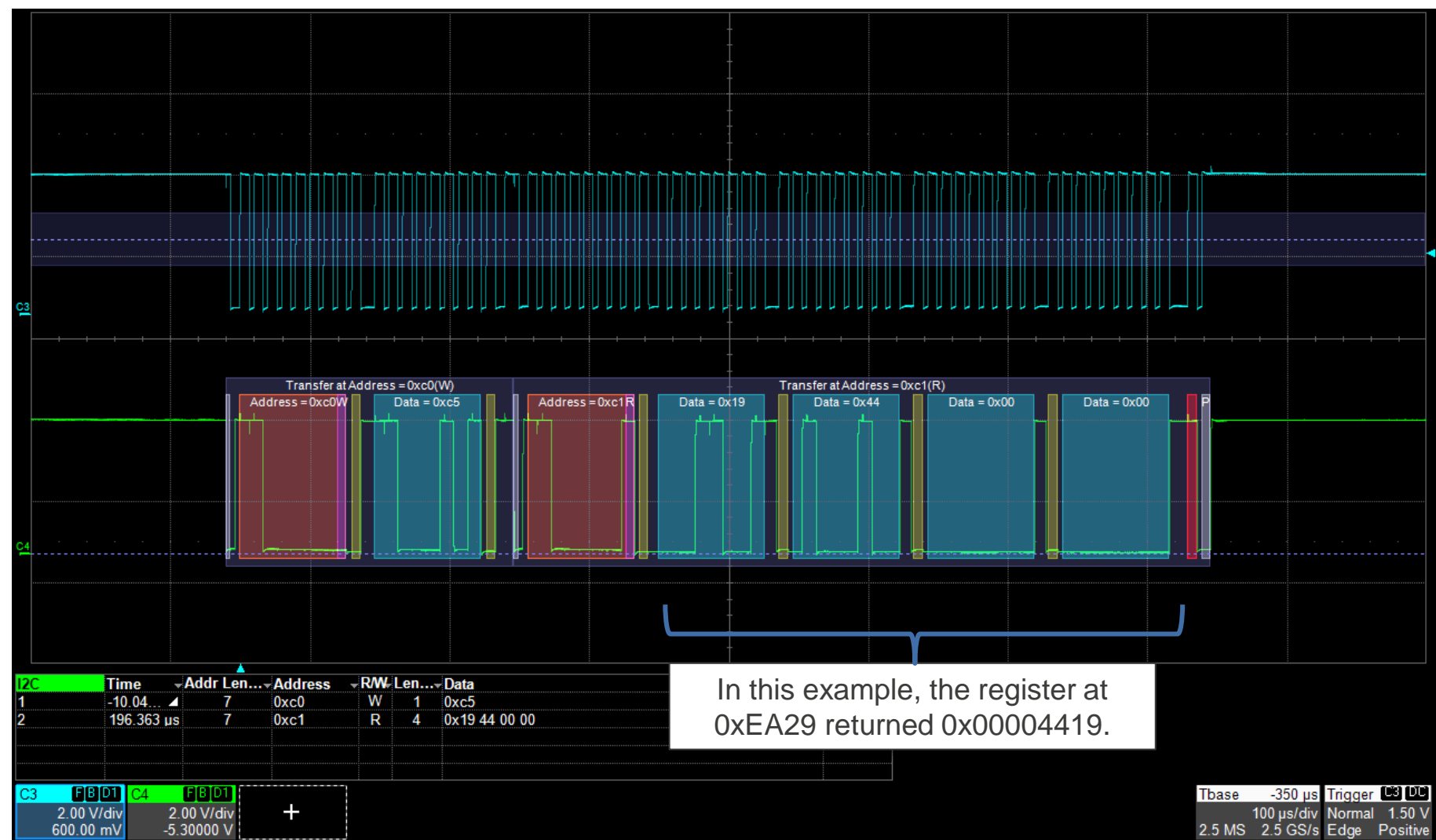


# DMA Data (0xC5) – Read

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

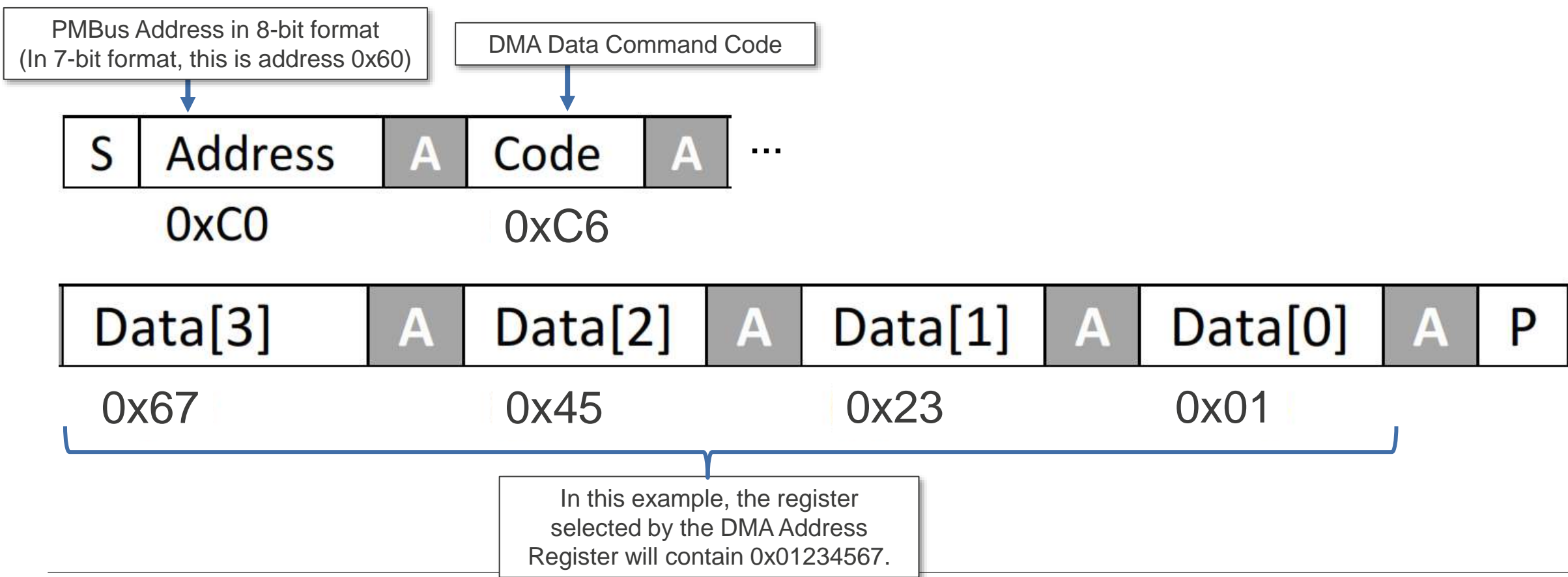


# DMA SEQUENTIAL (0XC6)

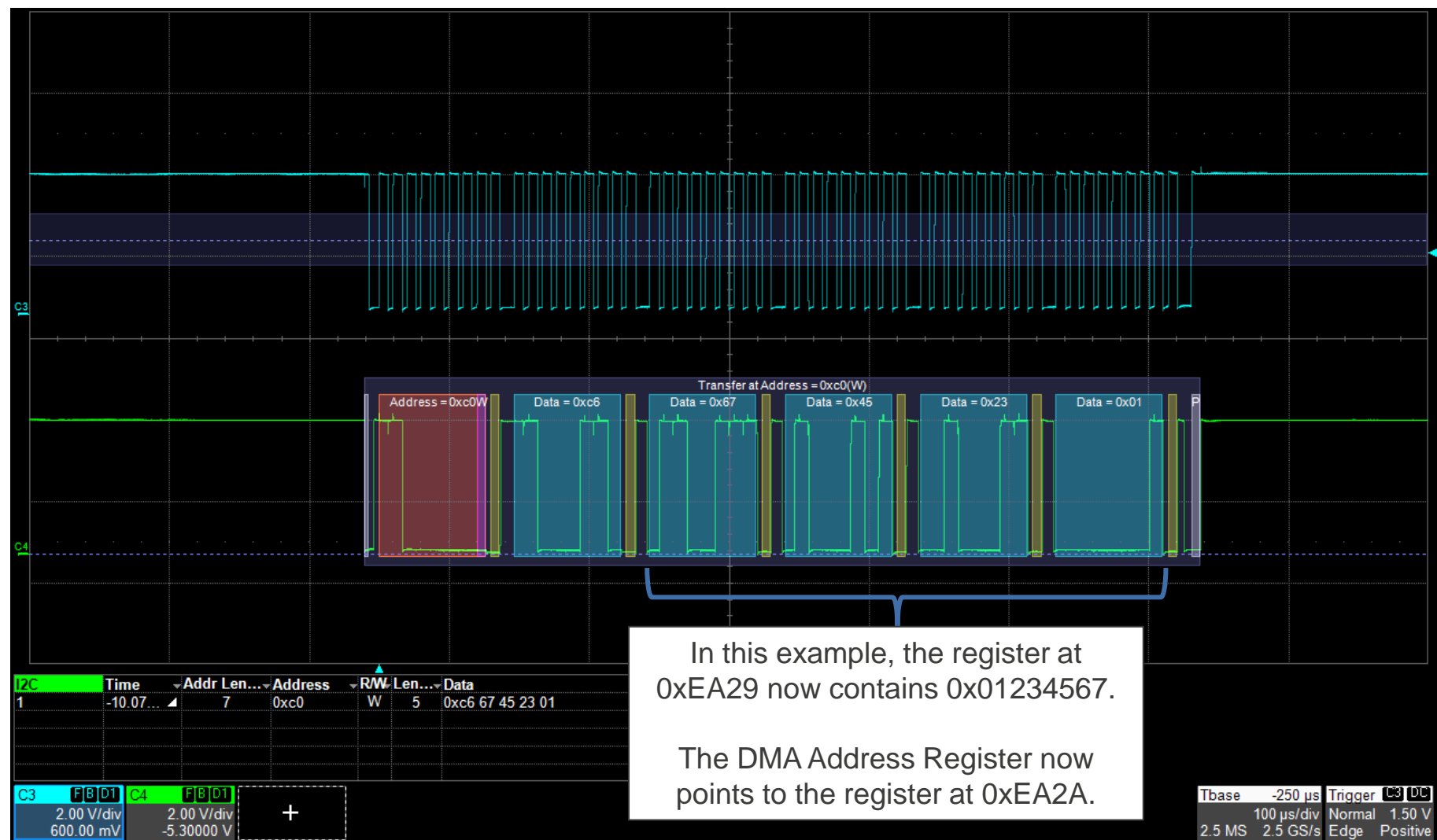
# DMA Sequential (0xC6) – Write

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.



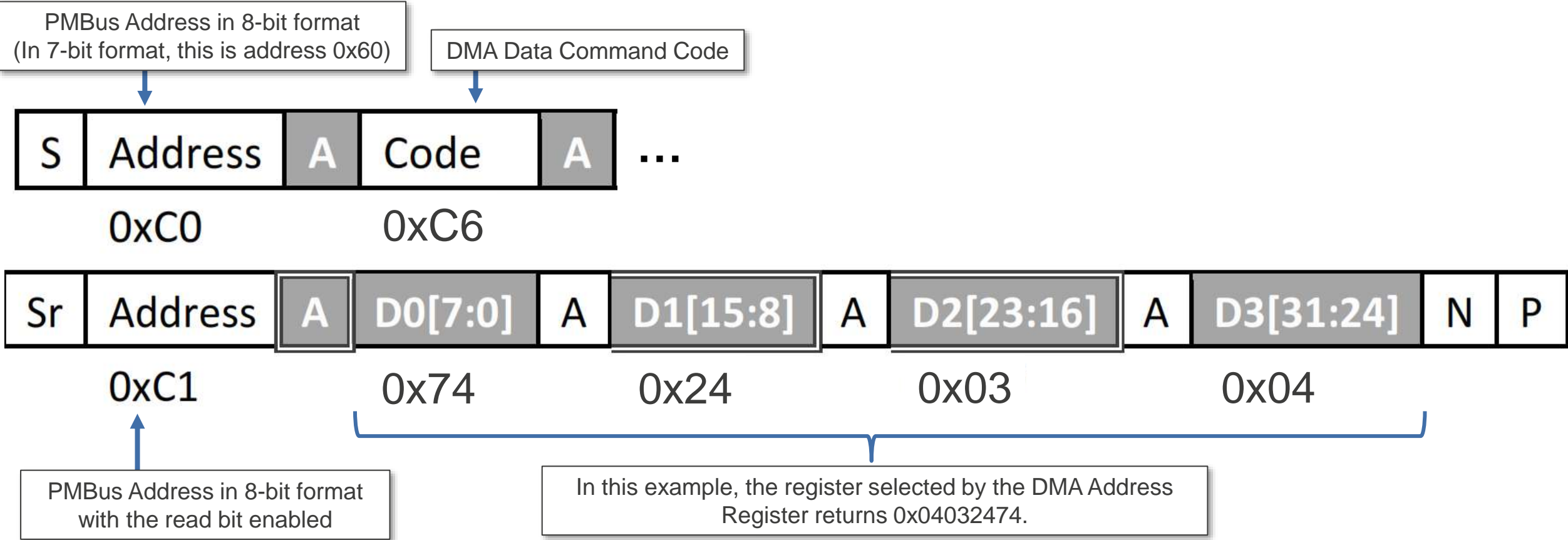
# DMA Sequential (0xC6) – Write Waveform



# 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





---

**BIG IDEAS FOR EVERY SPACE**

**[Renesas.com](https://www.renesas.com)**