AN-1453
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Programming the ADM1266 Firmware and Configuration via the PMBus

by Hossain Opal and Navdeep Dhanjal

INTRODUCTION

This application note details the steps required to program firmware and configuration on the ADM1266 in a mass programming environment or in a system application. The firmware and configuration updates can be performed individually based on the requirement.

TABLE OF CONTENTS

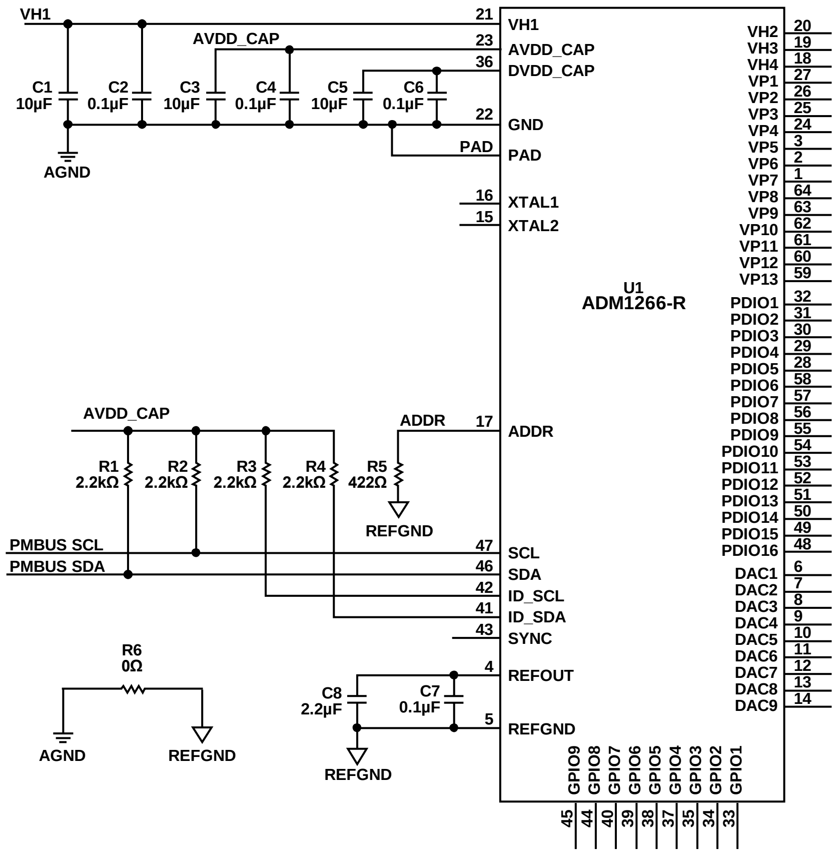
REVISION HISTORY

7/2018—Revision 0: Initial Version

HARDWARE SETUP

Figure 1 shows the recommended pin settings for the ADM1266 for a mass programming environment, only. It is recommended that all components used in the hardware setup are surface-mount components.

The PMBus address of the ADM1266 is set by connecting an external resistor (R5) from the ADDR pin to GND. Table 1 lists the recommended resistor values and associated PMBus addresses.


VH1
C1 10µF
C2 0.1µF
C3 10µF
C4 0.1µF
C5 10µF
C6 0.1µF
AVDD_CAP
AGND
PAD
U1
ADM1266-R
VH1
AVDD_CAP
DVDD_CAP
GND
PAD
XTAL1
XTAL2
ADDR
SCL
SDA
ID_SCL
ID_SDA
SYNC
REFOUT
REFGND
R1 2.2kΩ
R2 2.2kΩ
R3 2.2kΩ
R4 2.2kΩ
R5 422Ω
REFGND
PMBUS SCL
PMBUS SDA
R6 0Ω
AGND
REFGND
C8 2.2µF
C7 0.1µF
REFGND
VH2 VH3 VH4 VP1 VP2 VP3 VP4 VP5 VP6 VP7 VP8 VP9 VP10 VP11 VP12 VP13
PDIO1 PDIO2 PDIO3 PDIO4 PDIO5 PDIO6 PDIO7 PDIO8 PDIO9 PDIO10 PDIO11 PDIO12 PDIO13 PDIO14 PDIO15 PDIO16
DAC1 DAC2 DAC3 DAC4 DAC5 DAC6 DAC7 DAC8 DAC9
GPIO9 GPIO8 GPIO7 GPIO6 GPIO5 GPIO4 GPIO3 GPIO2 GPIO1

NOTES
1. FLOATING PINS ARE NOT CONNECTED (NC).
2. VH1 MUST BE CONNECTED TO 12V.

15641-001

Figure 1. Recommended Pin Settings for a Mass Programming Environment

Table 1. PMBus Address Settings

PMBus Address	1% Resistor (kΩ), E96 Series
0x40	0.422
0x41	1.5
0x42	2.67
0x43	4.12
0x44	5.36
0x45	7.15
0x46	8.87
0x47	10.7
0x48	12.7
0x49	14.7
0x4A	16.9
0x4B	19.1
0x4C	21.5
0x4D	24.3
0x4E	27.4
0x4F	31.6

MEMORY PARTITION

Figure 2 shows an overview of the memory partition applicable to this application note.

The ADM1266 is preprogrammed with the mini bootloader and bootloader only. The mini bootloader and bootloader are not upgradable by the user.

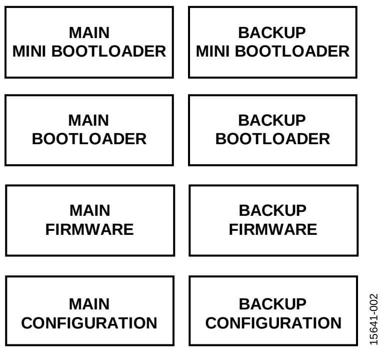


Figure 2. The ADM1266 Memory Partition

When the ADM1266 is preprogrammed with only bootloaders, the ADM1266 must first be programmed with firmware, followed by the configuration data.

The firmware and configuration for the ADM1266 is generated by ADI Power Studio® graphical user interface (GUI) software, which is available for download from the EVAL-ADM1266 evaluation board product page.

The same firmware is used for all devices, and a different configuration is used for different devices.

READING THE .HEX FILE

The .hex files generated for the firmware and the configuration follow the Intel® hexadecimal format.

Each line of the .hex file contains a hexadecimal number that encodes multiple binary numbers for representing the data and the register address where the data must be written.

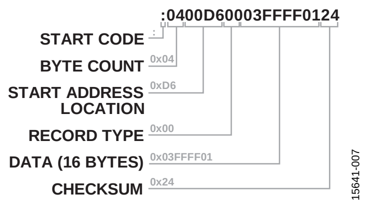


Figure 3. A Typical Record from the ADM1266 .Hex File

The end of the .hex file is represented by :00000001FF.

The PMBus write type is based on the byte count, such as 1 byte results in a byte write, 2 bytes result in a word write, and 3 or more bytes result in a block write.

PROGRAMMING FIRMWARE

The ADM1266 is powered up in bootloader mode and stays in bootloader mode if no firmware is present. As shown in Figure 4, only Step 4 and Step 5 are required to load the firmware.

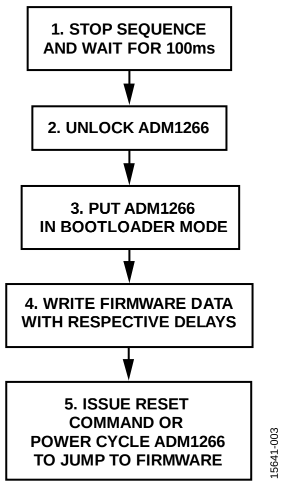

1. STOP SEQUENCE AND WAIT FOR 100ms
2. UNLOCK ADM1266
3. PUT ADM1266 IN BOOTLOADER MODE
4. WRITE FIRMWARE DATA WITH RESPECTIVE DELAYS
5. ISSUE RESET COMMAND OR POWER CYCLE ADM1266 TO JUMP TO FIRMWARE
15641-003

Figure 4. Flowchart for Programming Firmware

STEP 1: STOP THE SEQUENCE BEFORE FIRMWARE UPGRADED

Before the firmware is updated via the PMBus, the current sequence must be stopped. The sequence is stopped by writing 0x0003 to the GO_COMMAND command (Register 0xD8). A group command can be issued to stop multiple ADM1266 devices on the same PMBus simultaneously, as shown in Figure 5.

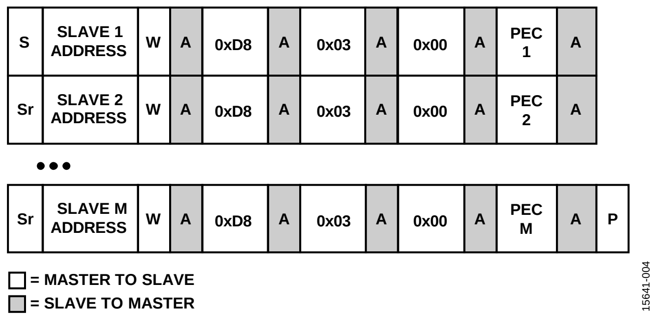

S | SLAVE 1 ADDRESS | W | A | 0xD8 | A | 0x03 | A | 0x00 | A | PEC 1 | A
Sr | SLAVE 2 ADDRESS | W | A | 0xD8 | A | 0x03 | A | 0x00 | A | PEC 2 | A
•••
Sr | SLAVE M ADDRESS | W | A | 0xD8 | A | 0x03 | A | 0x00 | A | PEC M | A | P
☐ = MASTER TO SLAVE
▨ = SLAVE TO MASTER
15641-004

Figure 5. Group Command for Stopping a Sequence

If the group command cannot be issued, each ADM1266 device can be stopped individually.

After issuing a stop command, it is required to wait for 100 ms before sending the next PMBus command.

STEP 2: UNLOCK THE ADM1266

The ADM1266 must be unlocked before the firmware can be updated.

The ADM1266 can be unlocked by performing two consecutive writes of the correct password to the FW_PASSWORD command (Register 0xFD). The block write command to unlock the ADM1266 is shown in Figure 6. The default password is 0xFF for the first 16 bytes, and the 17th byte (password command) is set to 0x02.

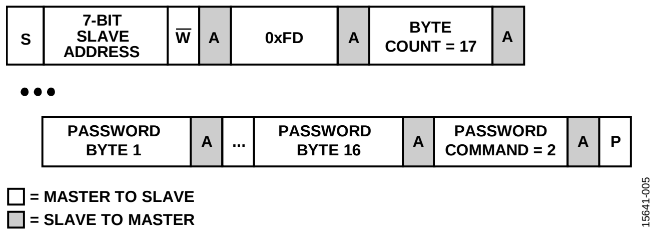

S | 7-BIT SLAVE ADDRESS | $\overline{W}$ | A | 0xFD | A | BYTE COUNT = 17 | A
•••
PASSWORD BYTE 1 | A | ... | PASSWORD BYTE 16 | A | PASSWORD COMMAND = 2 | A | P
☐ = MASTER TO SLAVE
▨ = SLAVE TO MASTER
15641-005

Figure 6. Block Write Command for Unlocking the ADM1266

The unlock status can be confirmed by the PART_LOCKED bit (Bit 2) of the STATUS_MFR_SPECIFIC command (Register 0x80), which is set to 0 when the ADM1266 is unlocked. See Table 5 for more information.

STEP 3: PLACE THE ADM1266 IN BOOTLOADER MODE

The ADM1266 must be placed in bootloader mode before the firmware can be programmed. The ADM1266 can be placed in bootloader mode by writing 0x02, 0x00, and 0x00 to the UPDATE_FW command (Register 0xFC) as a group command, as shown in Figure 7.

If the command cannot be issued, each ADM1266 device can be placed in bootloader mode individually.

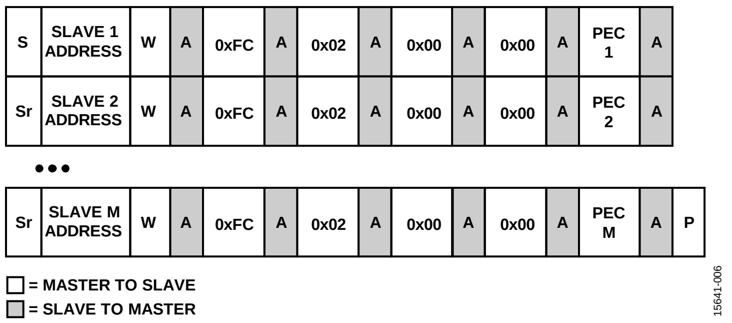

S | SLAVE 1 ADDRESS | W | A | 0xFC | A | 0x02 | A | 0x00 | A | 0x00 | A | PEC 1 | A
Sr | SLAVE 2 ADDRESS | W | A | 0xFC | A | 0x02 | A | 0x00 | A | 0x00 | A | PEC 2 | A
•••
Sr | SLAVE M ADDRESS | W | A | 0xFC | A | 0x02 | A | 0x00 | A | 0x00 | A | PEC M | A | P
☐ = MASTER TO SLAVE
▨ = SLAVE TO MASTER
15641-006

Figure 7. Group Command to Place the ADM1266 in Bootloader Mode

STEP 4: DELAYS BETWEEN WRITING FIRMWARE DATA

When the firmware is programmed in the ADM1266 via the PMBus, the user is not required to erase the EEPROM. The bootloader automatically erases the EEPROM pages that are reserved for firmware when the bootloader receives the first stream of firmware data. Table 2 states the respective delays between writing data.

STEP 5: ISSUE A RESET COMMAND OR POWER CYCLE AFTER PROGRAMMING FIRMWARE

After the firmware data is successfully written to the ADM1266, it is required to issue a reset command or power cycle before programming the configuration data.

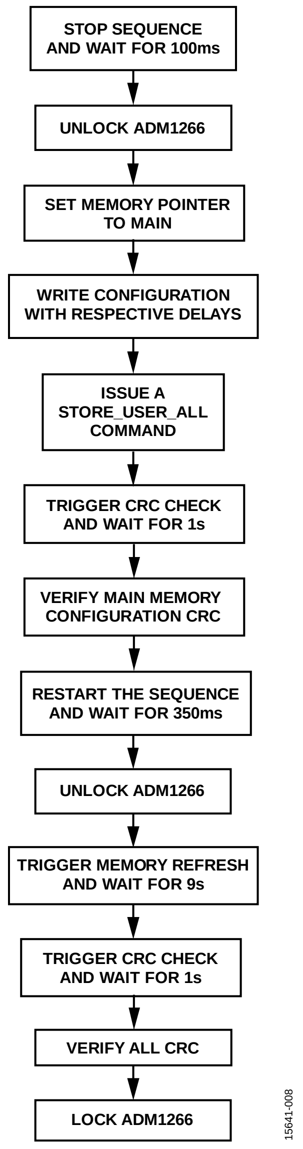
The ADM1266 can be reset by writing 0x0004 to the GO_COMMAND command (Register 0xD8).

Table 2. Delay Between Writing to the UPDATE_FW Command

Command	Wait Time Required Before Writing Next Block of Data
First Write to 0xFC (UPDATE_FW)	2 sec
Subsequent Writes to 0xFC	40 ms

PROGRAMMING THE CONFIGURATION

See Figure 8 for the flowchart for programming the configuration.



STOP SEQUENCE AND WAIT FOR 100ms
UNLOCK ADM1266
SET MEMORY POINTER TO MAIN
WRITE CONFIGURATION WITH RESPECTIVE DELAYS
ISSUE A STORE_USER_ALL COMMAND
TRIGGER CRC CHECK AND WAIT FOR 1s
VERIFY MAIN MEMORY CONFIGURATION CRC
RESTART THE SEQUENCE AND WAIT FOR 350ms
UNLOCK ADM1266
TRIGGER MEMORY REFRESH AND WAIT FOR 9s
TRIGGER CRC CHECK AND WAIT FOR 1s
VERIFY ALL CRC
LOCK ADM1266
15641-008

Figure 8. Flowchart for Programming the Configuration

Follow the procedures detailed in the Step 2: Unlock the ADM1266 section and the Reading the .Hex File section.

STOP THE SEQUENCE BEFORE PROGRAMMING THE CONFIGURATION DATA

Before the configuration is programmed via the PMBus, the current sequence must be stopped. There are two options for stopping the sequence: seamless update reset stop and soft reset stop. When updating the configuration, it is recommended to use seamless update reset stop, which causes the sequence to jump directly to the power-good state of the new configuration when the sequence is resumed.

In a seamless update reset stop, the sequence is stopped by writing 0x0011 to the GO_COMMAND command (Register 0xD8) as a group command to stop multiple ADM1266 devices on the same PMBus simultaneously, as shown in Figure 9.

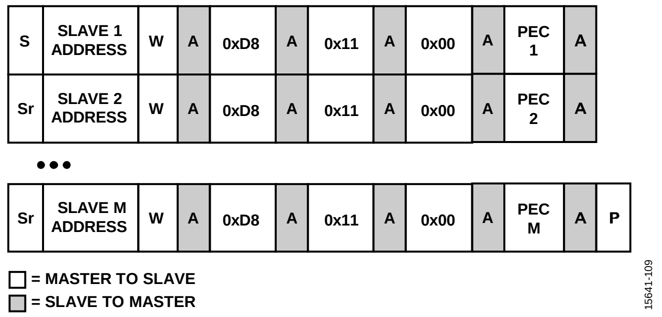


Figure 9. Group Command for Stopping a Sequence for Seamless Update

Seamless update reset stops have a default timeout of 10 minutes. If the sequence is not resumed by the user, the sequence automatically resumes after the timeout expires. The timeout value can be updated using the HITLESS_TIMEOUT command (Register 0xF6) before stopping the sequence. Figure 10 shows an example for setting the reset time to 60 sec.

It is only recommended to use a soft reset stop in a mass programming environment or when there is no previous configuration programmed.

In a soft reset stop, the sequence is stopped by writing 0x0003 to the GO_COMMAND command (Register 0xD8) as a group command to stop multiple ADM1266 devices on the same PMBus simultaneously, as shown in Figure 9.

If the group command cannot be issued, each ADM1266 device can be stopped individually.

After issuing a stop command, it is required to wait for 100 ms before sending the next PMBus command.

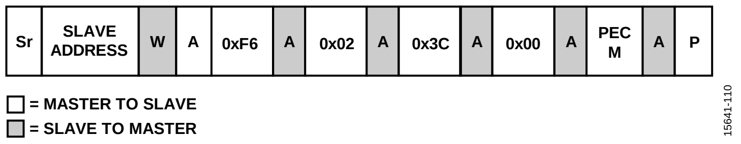


Figure 10. Hitless Timeout Command

SETTING THE MEMORY POINTER TO MAIN MEMORY

The configuration data for the ADM1266 must be programmed to the main memory. If the configuration data in the main memory is valid, the ADM1266 always runs the configuration from the main memory. If the configuration data in the main memory is corrupt and the configuration data in the backup memory is valid, the ADM1266 runs the configuration data from the backup memory.

The memory pointer can be set to the main memory by using the SWITCH_MEMORY command (Register 0xFA), as shown in Figure 11.

DELAYS BETWEEN WRITING DATA

When the configuration data is programmed in the ADM1266 via the PMBus, the user is not required to erase the EEPROM.

The firmware automatically erases the EEPROM pages based on the data sent to the ADM1266. The typical time needed to erase one EEPROM page is 50 ms.

Due to the time needed for the automatic erase of the EEPROM, the user is required to wait for a specific time before sending the next stream of data.

Table 3 states the typical delay that must be maintained after writing to specific commands and the corresponding line number in the sample .hex file shown in Figure 12.

After the configuration write is complete, issue a send command to the STORE_USER_ALL command (Register 0x15) to commit the system configuration settings to the EEPROM.

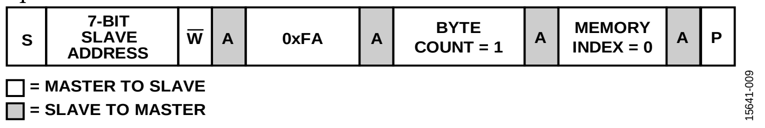
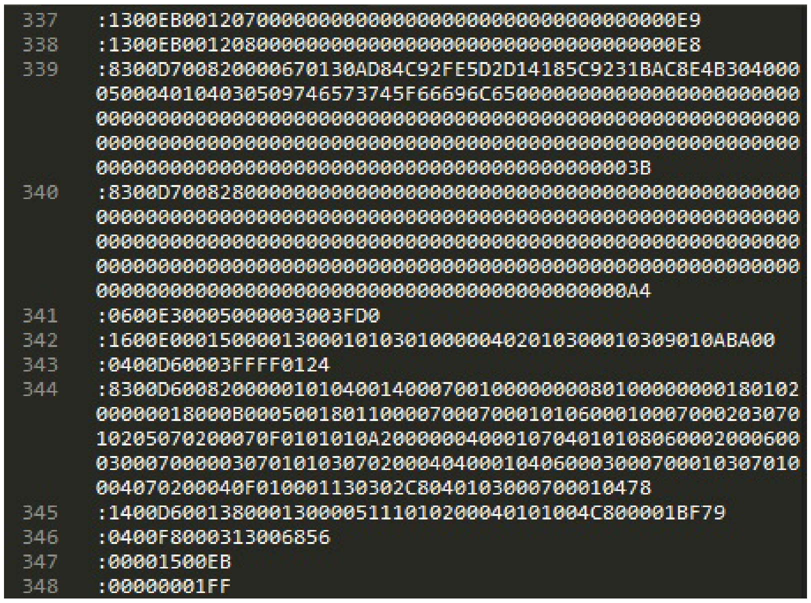


Figure 11. Setting the Memory Pointer

Table 3. Delays Between Writing to Commands

Command	Wait Time Required Before Writing Next Block of Data	Line Number in .hex File (See Figure 12)
0xD7 (SYSTEM_CONFIGURATION)	400 ms (after first write)	339
0xE3 (USER_DATA)	100 ms (after first write)	341
0xE0 (LOGIC_CONFIGURATION)	200 ms (after first write)	342
0xD6 (SEQUENCE_CONFIGURATION)	2.5 sec (after first write)	343
0xF8 (MEMORY_CONFIG)	100 ms	346
Subsequent Writes to 0xD7, 0xE3, 0xE0, 0xD6	40 ms	340, 344, 345
0x15 (STORE_USER_ALL)	300 ms	347



```
337  :1300EB001207000000000000000000000000000000000000E9
338  :1300EB001208000000000000000000000000000000000000E8
339  :8300D700820000670130AD84C92FE5D2D14185C9231BAC8E4B304000
     050004010403050974657374 5F66696C6500000000000000000000000
     000000000000000000000000000000000000000000000000000000000
     000000000000000000000000000000000000000000000000000000000
     00000000000000000000000000000000000000000003B
340  :8300D700828000000000000000000000000000000000000000000000
     000000000000000000000000000000000000000000000000000000000
     000000000000000000000000000000000000000000000000000000000
     000000000000000000000000000000000000000000000000000000000
     0000000000000000000000000000000000000000000A4
341  :0600E30005000003003FD0
342  :1600E00015000013000101030100000402010300010309010ABA00
343  :0400D60003FFFF0124
344  :8300D600820000010104001400070010000000080100000000180102
     000000180000B000500180110000700070001010600010007000203070
     102050702000 70F0101010A2000000400010704010108060002000600
     030007000003070101030702000404000104060003000700010307010
     004070200040F010001130302C804010300070001 0478
345  :1400D60013800013000051110102000040101004C800001BF79
346  :0400F8000313006856
347  :00001500EB
348  :00000001FF
```

Figure 12. Sample .hex File

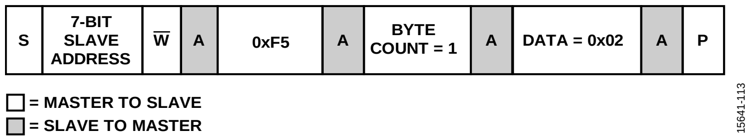


Figure 13. Triggering Memory Refresh

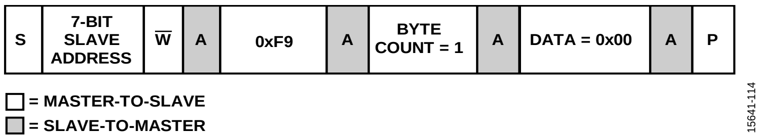


Figure 14. Triggering CRC Recalculate

VERIFYING THE UPDATED MAIN MEMORY CONFIGURATION DATA

Verification is required only when updating the configuration. Verification of the updated main memory configuration data is optional when no previous configuration is programmed in the ADM1266.

After writing the configuration data, cyclic redundancy check (CRC) calculations of all the memory sections can be triggered by writing to the MEMORY_RECALCULATE_CRC command (Register 0xF9), as shown in Figure 14.

Allow 1 sec to elapse before reading the status of the memory CRC, which can be check by a word read of the STATUS_MFR_SPECIFIC_2 command (Register 0xED). Bit mapping for this register is detailed in Table 4. Verify that the MAIN_PROJECT_CRC_FAULT bit and the MAIN_ABCONFIG_CRC_FAULT bit are set to 0. These bits being set to 0 indicates that the main memory configuration was written successfully.

RESTARTING THE SEQUENCE AFTER UPDATING

When the new configuration data is written to the ADM1266, the new sequence can be run by writing 0x0000 to the GO_COMMAND command (Register 0xD8) as a group command to restart the sequence in multiple ADM1266 devices simultaneously. The group command protocol is shown in Figure 5.

If the group command cannot be issued, each ADM1266 device can be restarted individually.

If the sequence is stopped using seamless update reset stop, the sequence jumps directly to the power-good state defined in the configuration when restarted. If the no power-good state is defined, the sequence starts from the first user state.

If the sequence is stopped using soft reset stop, the sequence starts from the first user state when restarted, causing the board to power cycle.

After restarting the sequence, wait 350 ms before sending the next PMBus command.

TRIGERRING MEMORY REFRESH

The ADM1266 contains backup memory sections for the mini bootloader, bootloader, firmware, and configuration. After the firmware and configuration are successfully programmed, trigger memory refresh. Memory refresh copies the main memory content to the backup memory, and the backup memory to the main memory.

Confirm that the ADM1266 is unlocked before triggering the memory refresh. See the Step 2: Unlock the ADM1266 section.

Memory refresh for all the memory sections can be triggered by performing a block write to the REFRESH_FLASH command (Register 0xF5), as shown in Figure 13.

A memory refresh requires approximately 9 sec to complete. Check the status of the memory refresh by reading the RUNNING_REFRESH bit in the STATUS_MFR_SPECIFIC command (Register 0x80). If the RUNNING_REFRESH bit is set to 1, memory refresh is still running. Wait until this bit is set to 0 before proceeding.

VERIFYING FIRMWARE AND CONFIGURATION DATA

After writing the firmware, configuration data, and triggering a memory refresh, CRC calculations of all the memory sections can be triggered by writing to the MEMORY_RECALCULATE_CRC command (Register 0xF9), as shown in Figure 14.

Allow 1 sec to elapse before reading the status of the memory CRC. To check the status of the memory CRC, perform a word read of the STATUS_MFR_SPECIFIC_2 command (Register 0xED). The bit mapping for this register is detailed in Table 4.

Table 4. STATUS_MFR_SPECIFIC_2 Command Bit Mapping

Bit	Bit Name
15	BKUP_PASSWORD_CRC_FAULT
14	BKUP_FIRMWARE_CRC_FAULT
13	BKUP_PROJECT_CRC_FAULT
12	BKUP_ABCONFIG_CRC_FAULT
11	MAIN_PASSWORD_CRC_FAULT
10	MAIN_FIRMWARE_CRC_FAULT
9	MAIN_PROJECT_CRC_FAULT
8	MAIN_ABCONFIG_CRC_FAULT
7	BKUP_IAP_CRC_FAULT
6	BKUP_MINI_IAP_CRC_FAULT
5	MAIN_IAP_CRC_FAULT
4	MAIN_MINI_IAP_CRC_FAULT

ALL_CRC_FAULT is a single bit that is an OR of all the CRC bits shown in Table 4. This bit can read from the STATUS_MFR_SPECIFIC command (Register 0x80).

Check the status of the configuration compatibility (Bit 1) and silicon compatibly (Bit 0) by reading back the STATUS_MFR_SPECIFIC command (Register 0x80).

Table 5. STATUS_MFR_SPECIFIC Command Bit Mapping

Bits	Bit Name
5	ALL_CRC_FAULT
4	TESTMODE_ENABLED
3	RUNNING_REFRESH
2	PART_LOCKED
1	PART_DATA_COMPATIBLE_FAULT
0	SILICON_COMPATIBLE_FAULT

If the CRC and compatibility pass, the bit in which they passed is set to 0. If the CRC and compatibility fail, the bit in which they failed is set to 1.

The firmware version number can be read back from the IC_DEVICE_REV command (Register 0xAE). Use a block read to validate that the firmware is updated to the correct revision. The byte mapping for the command is shown in Table 6.

Table 6. IC_DEVICE_REV Command Mapping

Byte Name	Byte	Example	Notes
Data Length	Byte 0	0x08	Fixed to 0x08
Firmware Revision	Byte 1	0x01	Indicates Version 1.8.7
	Byte 2	0x08	
	Byte 3	0x07	
Bootloader Revision	Byte 4	0x00	Indicates Version 0.0.7
	Byte 5	0x00	
	Byte 6	0x07	
Chip Revision	Byte 7	0x42	Indicates Version B0
	Byte 8	0x30	

LOCKING THE ADM1266

After a power cycle, the ADM1266 is automatically locked. The ADM1266 can also be locked by writing any 16 bytes of data, and the 17th byte (password command), which is set to 0x03, once to the FW_PASSWORD command (Register 0xFD). The block write command for locking the ADM1266 is shown in Figure 15.

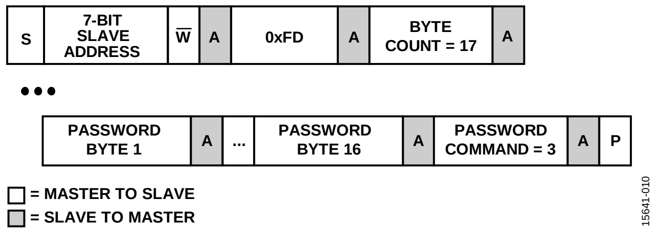


Figure 15. Block Write Command for Locking the ADM1266

The lock status can be confirmed by the PART_LOCKED bit (Bit 2) of the STATUS_MFR_SPECIFIC command (Register 0x80), which is set to 1 when the device is successfully locked.

PACKET ERROR CHECKING (PEC) CONTROL (OPTIONAL)

If the user has the capability to recognize acknowledge bits and no acknowledge bits in the I^{2}C communication, it is recommended to use the PEC byte for all writes to ensure proper data transfer.

The PMBus controller implements PEC to improve reliability and communication robustness. The PEC is implemented by appending a PEC byte at the end of the message transfer. The PEC byte is calculated using a CRC-8 algorithm on all address, command, and data bytes from the start bit to the stop bit. This process excludes the acknowledge, no acknowledge, start, restart, and stop bits. The PEC byte is appended to the end of the message by the device that supplied the last data byte. The receiver of the PEC byte is responsible for calculating the internal packet error code and comparing this code to the received PEC byte.

The PMBus slave device can communicate with master PMBus devices that support PEC, as well as those that do not support PEC. If a PEC byte is available, the PMBus checks the PEC byte and acknowledges if it is correct. If the PEC byte comparison fails, the PMBus device does not acknowledge the PEC byte and does not process the command sent from the master.

The PMBus uses built in hardware to calculate the PEC code using the CRC-8 polynomial, $C(x) = x^8 + x^2 + x^1 + 1$. The PEC code is calculated one bye at a time, in the order that it is received. In a read transaction, the PMBus appends the PEC byte following the last data byte. In a write transaction, the PMBus compares the received PEC byte to the internally calculated PEC code.