AN-1139
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

Understanding the Parallel Programming Protocol

by Eckart Hartmann

INTRODUCTION

The main method of programming the ADuC8xx family of parts is via serial programming as described in the relevant data sheets, which are available at www.analog.com, in conjunction with Application Note AN-1074.

However, for some users, it may be more convenient to program the parts via a standard device programmer. The ADuC8xx family allows for parallel programming so that suppliers of standardized programmers can support this family of devices.

This application note describes this parallel programming protocol. The protocol is essentially the same as that used for many standalone EPROM and EEPROM devices available in the market; however, some additional considerations must be taken into account.

The information in this application note applies to all ADuC83x and ADuC84x devices.

TABLE OF CONTENTS

1/12—Revision 0: Initial Version

MEMORY MAP

In parallel programming mode, the various areas of memory are mapped into portions of the available addressable space. Figure 1 shows this mapping for the small memory devices, ADuC834/ADuC836. Figure 2 shows this mapping for the larger memory devices, ADuC831/ADuC832/ADuC841/ADuC842/ADuC843/ADuC845/ADuC847/ADuC848. The MS bit as described in the Command Functions section selects addressing of program memory (MS = 1) vs. data memory (MS = 0).

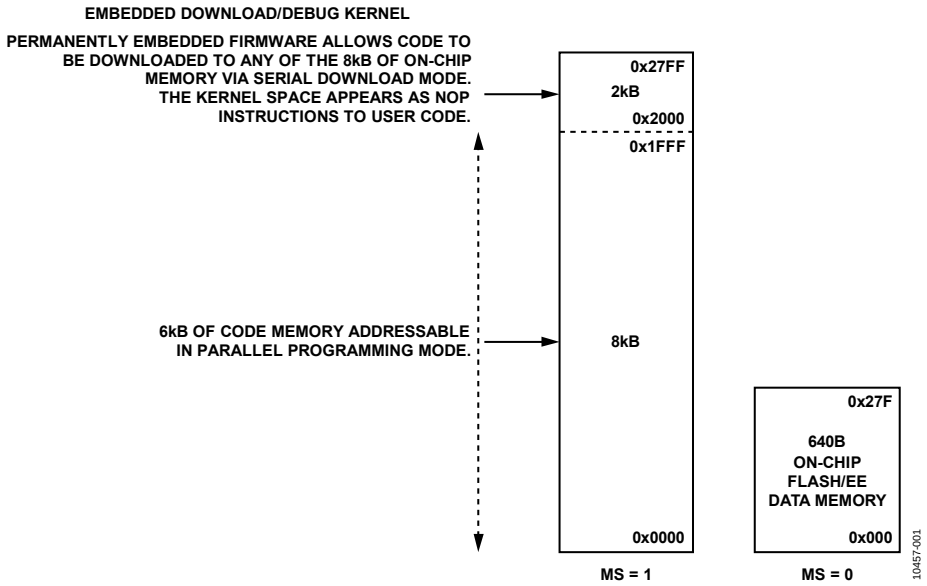

Figure 1. Parallel Programming Memory Map for Small Memory Devices

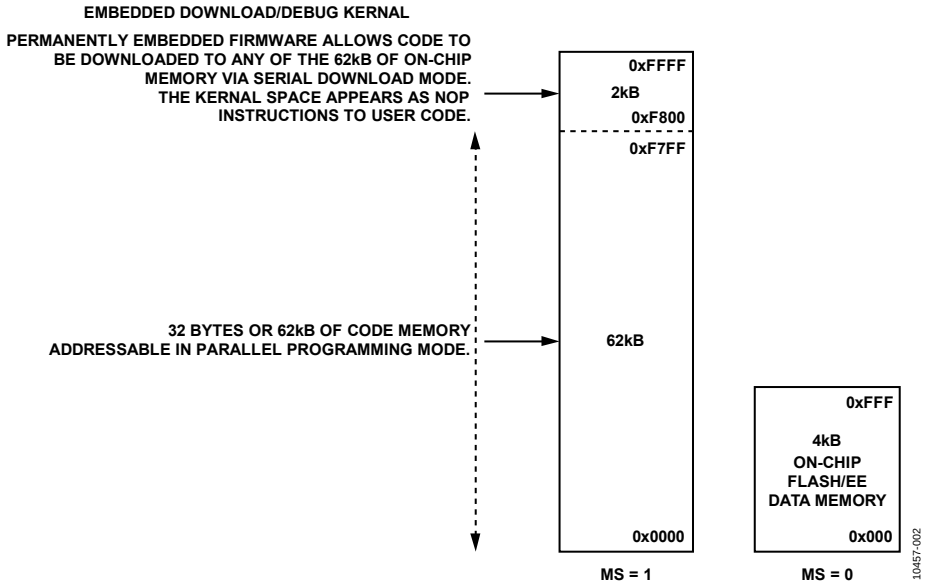

Figure 2. Parallel Programming Memory Map for Large Memory Devices

PIN CONFIGURATION

Parallel programming is achieved by sending a sequence of commands to the device using the signals shown in Figure 3. There is a special entry sequence followed by the different commands that are described in the Command Functions section.

- Port 3 is the 8-bit bidirectional data bus for programming and reading bytes.
- P1.1 to P1.4 is the 4-bit command input for specifying erase, program, and read.
- P1.5 to P1.7 supply the timing for the parallel programming.
- P1.0 is the active-low enable input for strobing a command on P1.1 to P1.4.
- $\overline{EA}$ is used for entry to parallel programming mode.
- For entry into parallel programming mode, PSEN must be connected to ground via a 1 kΩ resister at the base of the transistor shown.

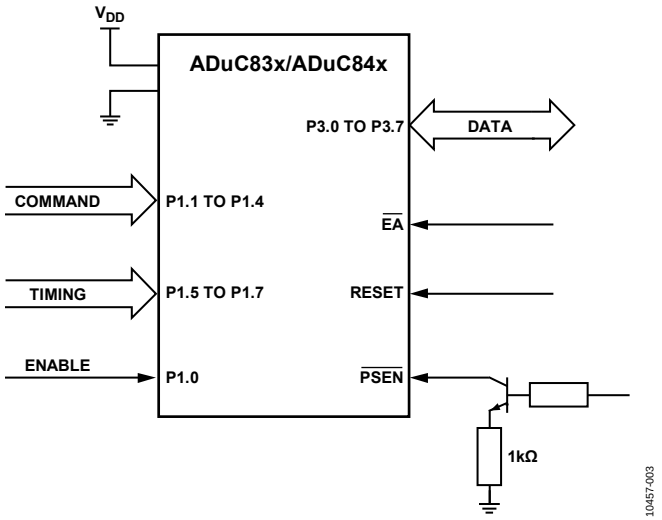

Figure 3. Pin Configuration for Parallel Programming

ENTRY INTO PARALLEL PROGRAMMING MODE

For entry into parallel programming, mode (P3.1 to P3.4) = 0b1101.

To ensure a safe command is placed on Port 1 when entering parallel mode, use the 0xCD command because this corresponds to a read address register command.

PSEN must be driven by a 1 kΩ resistor, as shown in Figure 3.

Note the following about the power supply:

- The ground pins (DGND and AGND) must be treated as a single node.
- The V_{DD} pins (AV_{DD} and DV_{DD}) must be treated as a single node.
- The voltage applied to any pin must never be greater than V_{DD} or less than ground.
- V_{DD} must remain stable and within specification throughout the entire programming process.

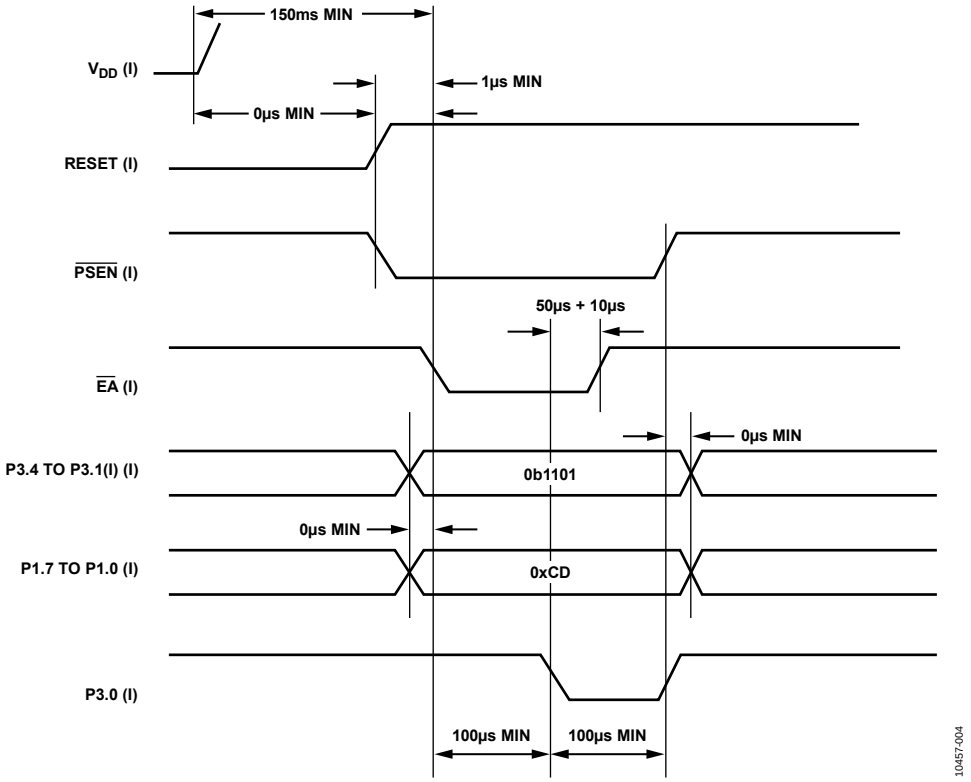

Figure 4. Entrance Sequence for Parallel Programming

COMMAND FUNCTIONS

Table 1 lists the commands that carry out the various parallel programming functions.

Table 1. Parallel Programming Commands

P1.4	P1.3	P1.2	P1.1	Function
0	0	0	0	Erase all (code/data plus security bits)
MS	0	1	0	Program byte
MS	0	1	1	Read byte
0	1	1	0	Read-address register
1	1	1	0	Program address register
1	1	1	1	Default pull-ups, do nothing

In the program byte function and read byte function, the MS bit in Table 1 is used to select program memory (P1.4 = 1) or data memory (P1.4 = 0).

PROGRAMMING THE ADDRESS REGISTER

EADRH and EADRL are the address registers for the microcontroller products. To program the address for byte programming, use the command and timing sequence shown in Figure 5.

Table 2. Program Address Register Command Key

P1.4	P1.3	P1.2	P1.1	Function
1	1	1	0	Program address register

Note that EADRL is automatically incremented following a byte or a page program command; therefore, no manual increment of EADRL is required for subsequent sequential programming commands.

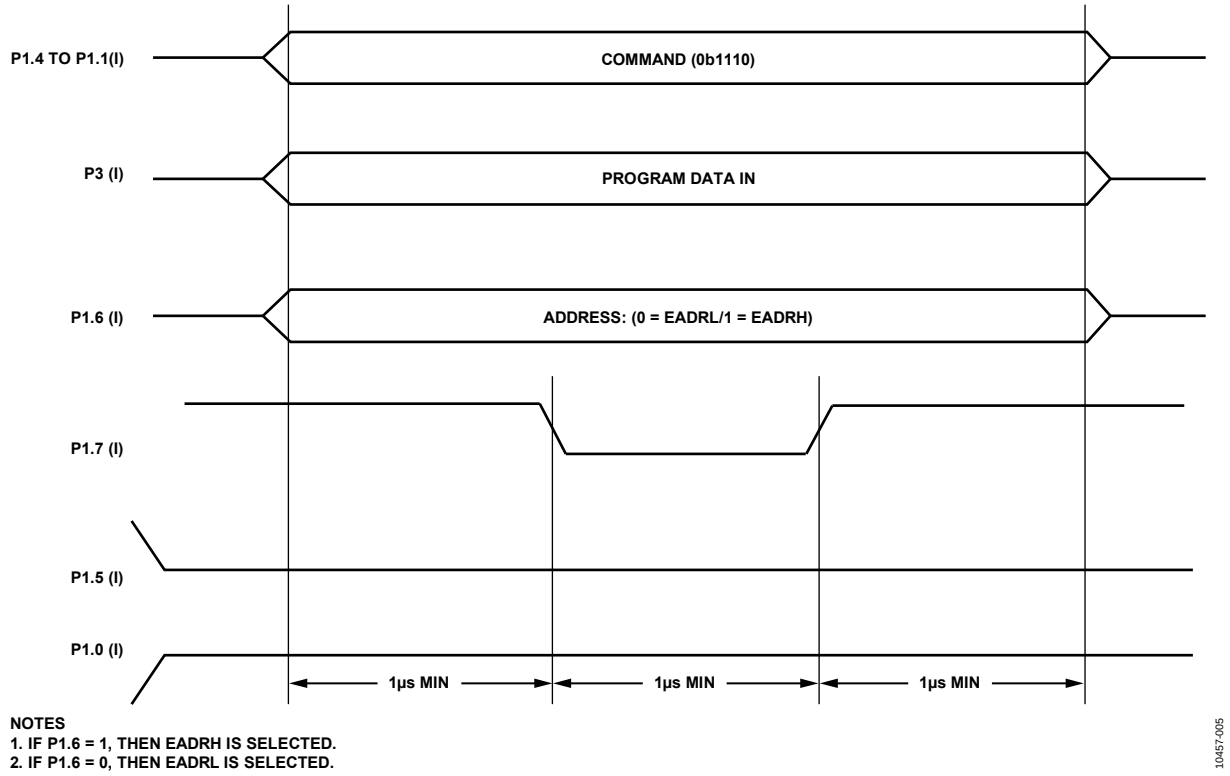

Figure 5. Program the Address Register

READING THE ADDRESS REGISTER

EADRH and EADRL are the address registers for the microcontroller products. To read the current address, use the command and timing sequence shown in Figure 6.

Table 3. Read Address Register Command Key

P1.4	P1.3	P1.2	P1.1	Function
0	1	1	0	Read address register

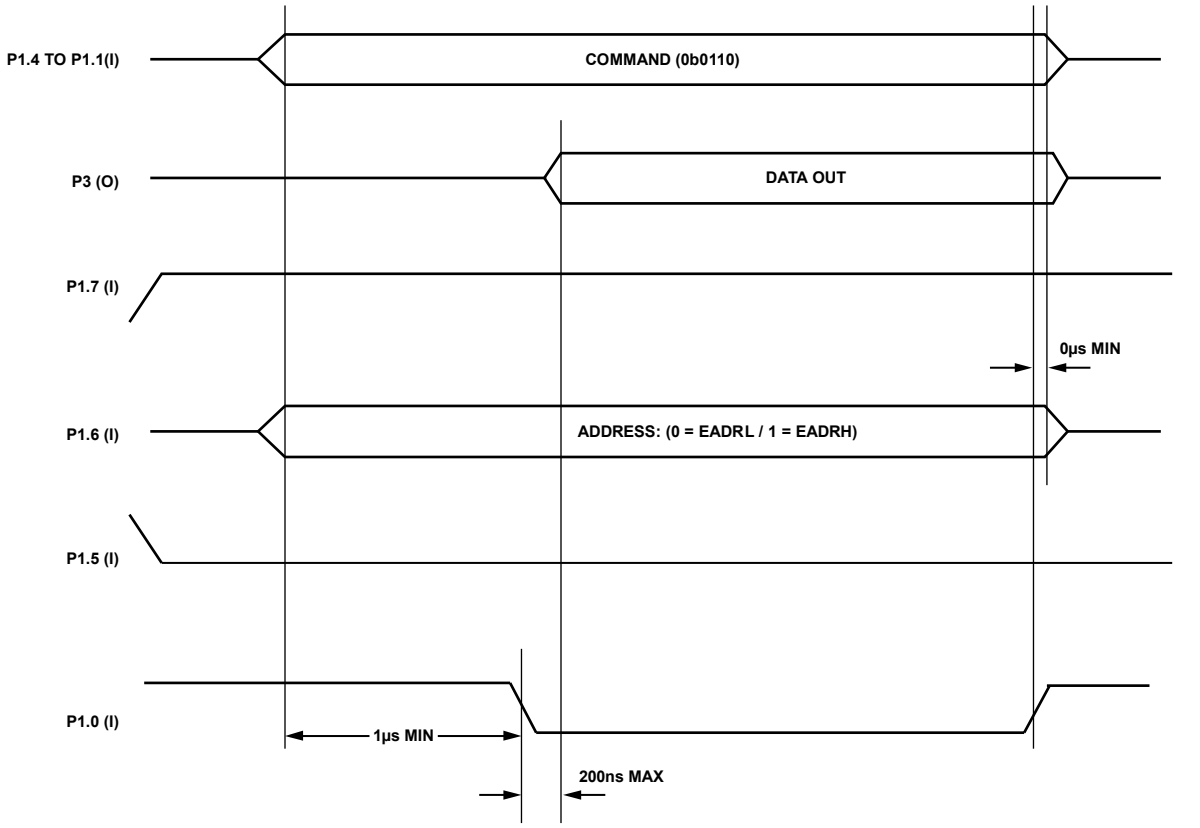

Figure 6. Read the Address Registers

ERASE ALL

To erase the NV data and program flash, use the command and timing sequence shown in Figure 7.

Table 4. Erase All Command Key

P1.4	P1.3	P1.2	P1.1	Function
0	0	0	0	Erase all (code/data plus security bits)

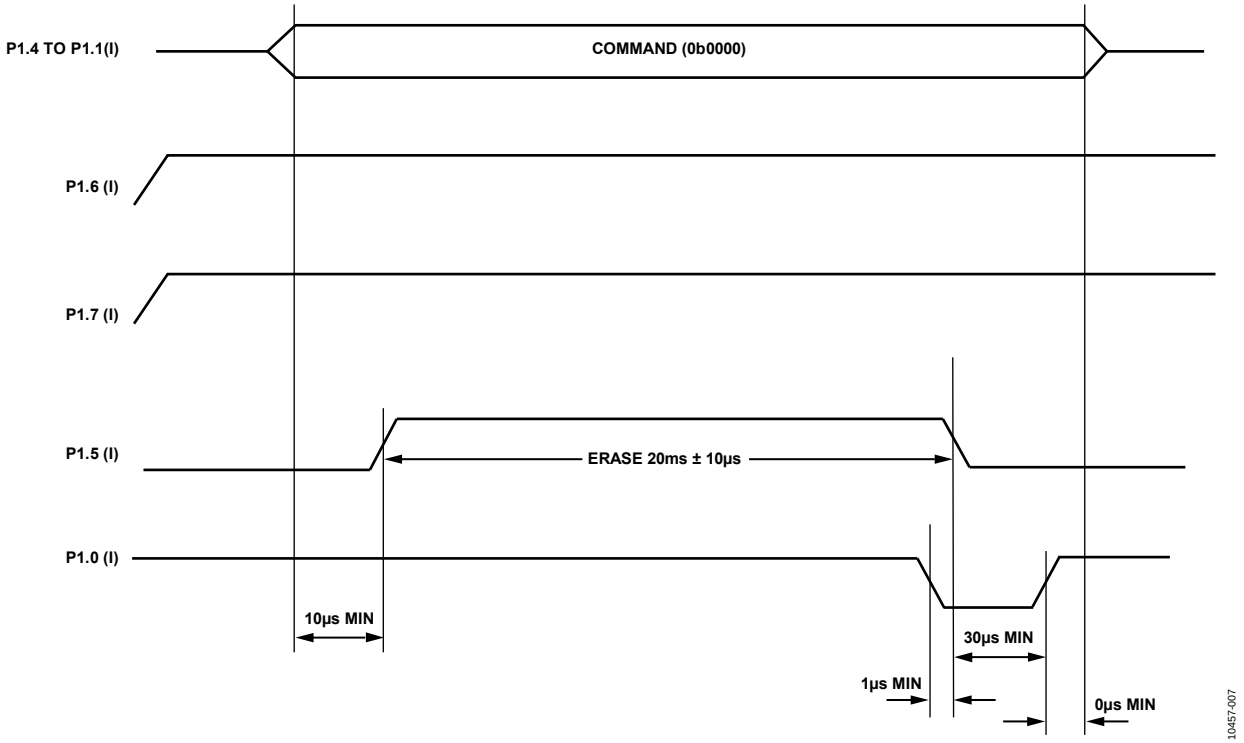

Figure 7. Erase All

PROGRAM BYTE

To program a byte at the location given by EADRH and EADRL (already programmed via the program address register operation, see Figure 5), use the command and timing sequence shown in Figure 8.

Table 5. Program Byte Command Key

P1.4	P1.3	P1.2	P1.1	Function
MS	0	1	0	Program byte

For a description of the MS bit, see the Command Functions section and Table 1.

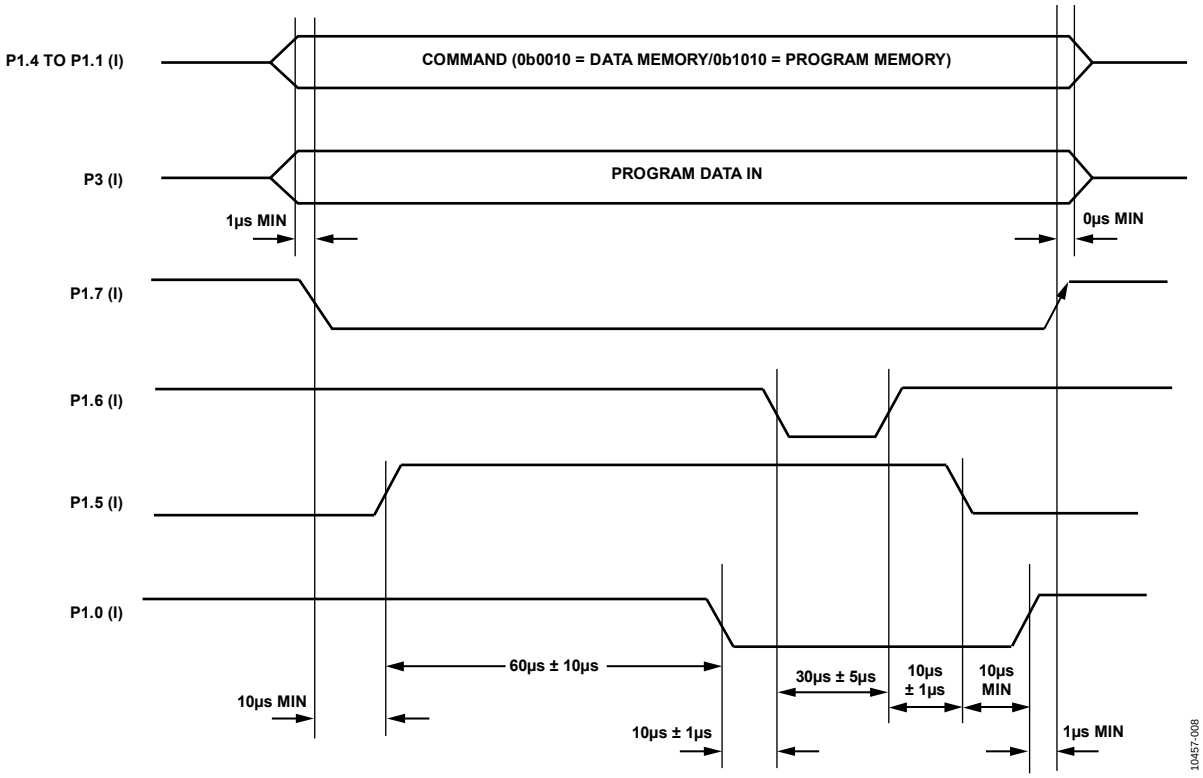

Figure 8. Program a Byte

PAGE PROGRAMMING

As an alternative to programming a single byte at a time, the page programming function can be chosen by following the timing diagram shown in Figure 9. Note that the command byte is the same as for a byte programming function, and in fact, the page programming function is little more than a series of consecutive single bytes programmed in quick succession within strict timing constraints. If for any reason, not all the timing requirements can be met, use byte programming instead.

Table 6. Page Programming Command Key

P1.4	P1.3	P1.2	P1.1	Function
MS	0	1	0	Program byte

For a description of the MS bit, see the Command Functions section and Table 1.

A page of program memory is 32 bytes in size, whereas a page of data memory is only two bytes in size. For program memory, the first address of a page ends in 0b00000 and the last address of a page ends in 0b11111. For data memory, the first address of a page ends in 0, and the second (that is, last) address in a page ends in 1. This is shown in Table 7 and Table 8. Page programming requires that all addresses in the page be programmed sequentially within a single sequence, starting with Address 0 and ending with Address 31 (or Address 1 for data memory).

Table 7. One Page of Program Memory

Address	Value
Address 0	XXXXXXXXXX00000
Address 1	XXXXXXXXXX00001
Address 2	XXXXXXXXXX00010
...	...
Address 31	XXXXXXXXXX11111

Table 8. One Page of Data Memory[1]

Address	Value
Address 0	XXXXXXXXXXXXXXX0
Address 1	XXXXXXXXXXXXXXX1

[1] In parallel programming mode, a page of data memory is 2 bytes, whereas in user mode, each page is 4 bytes.

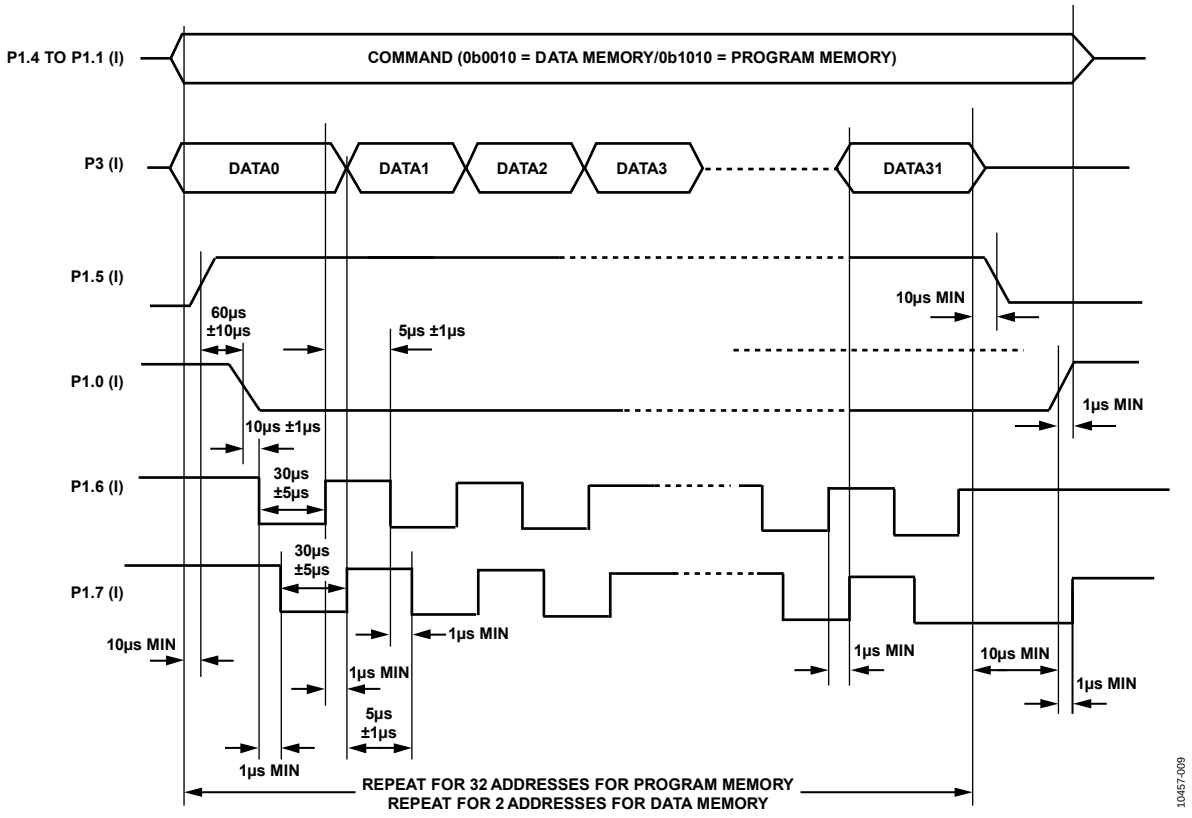

Figure 9. Page Programming

READ BYTE

To read the byte at address (EADRH:EADRL), use the command and timing sequence shown in Figure 10.

Table 9. Read Byte Command Key

P1.4	P1.3	P1.2	P1.1	Function
MS	0	1	1	Read byte

For a description of the MS bit, see the Command Functions section and Table 1.

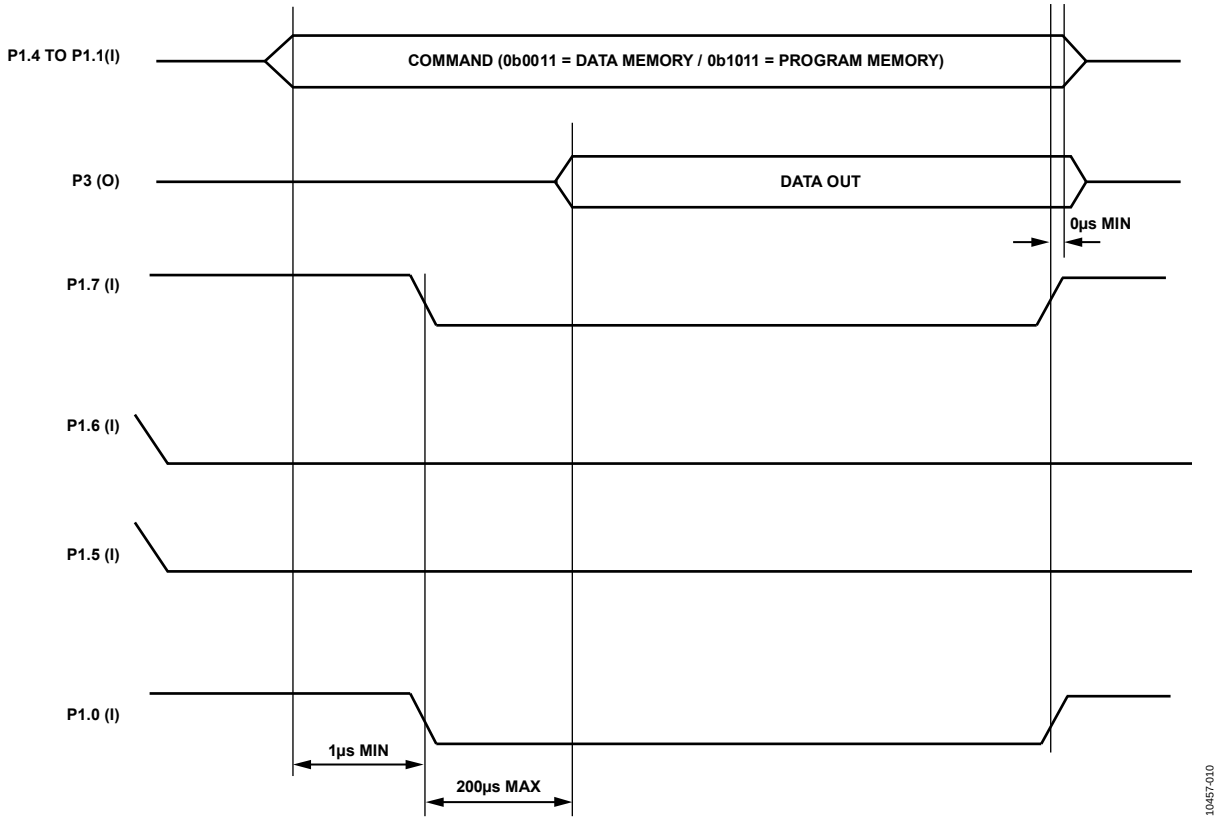

Figure 10. Read Byte

BYTE PROGRAM/BYTE READ PROGRAM FLOW

The sequence shown in Figure 11 assumes a fully erased part. If a byte location is not in an erased state, it may not program correctly.

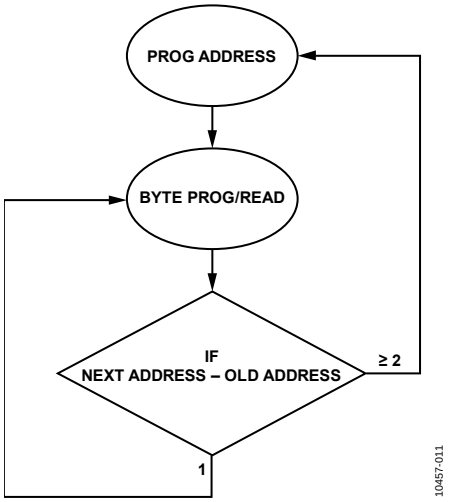

Figure 11. Byte Program/Read Sequence

AN10457-0-1/12(0)