PIC17CXX

EPROM Memory Programming Specification

This document includes the programming specifications for the following devices:

- PIC17C42
- PIC17C42A
- PIC17CR42
- PIC17C43
- PIC17CR43
- PIC17C44

1.0 PROGRAMMING THE PIC17CXX

The PIC17CXX is programmed using the TABLWT instruction. The table pointer points to the internal EPROM location start. Therefore, a user can program an EPROM location while executing code (even from internal EPROM). This programming specification applies to PIC17CXX devices in all packages.

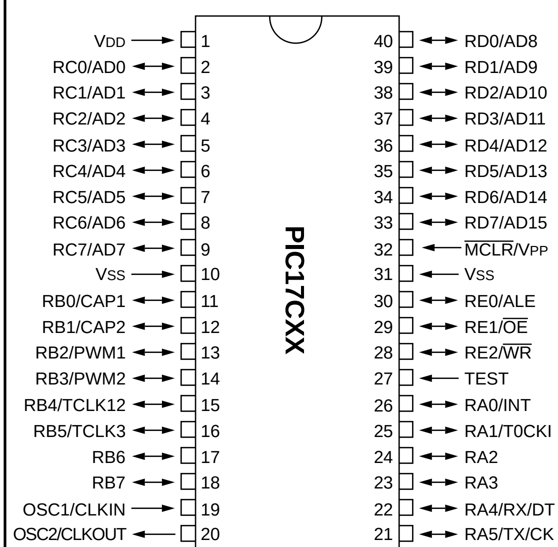
For the convenience of a programmer developer, a “program & verify” routine is provided in the on-chip test program memory space, the program resides in ROM and not EPROM. Therefore, it is not erasable. The “program/verify” routine allows the user to load any address, program a location, verify a location or increment to the next location. It allows variable programming pulse width.

1.1 Hardware Requirements

Since the PIC17CXX under programming is actually executing code from “boot ROM,” a clock must be provided to the part. Furthermore, the PIC17CXX under programming may have any oscillator configuration (EC, XT, LF or RC). Therefore, the external clock driver must be able to overdrive pulldown in RC mode. CMOS drivers are required since the OSC1 input has a Schmitt trigger input with levels (typically) of 0.2VDD and 0.8VDD. See the PIC17C4X data sheet (DS30412A) for exact specifications.

Pin Diagram

40L PDIP, Windowed CERDIP



The PIC17CXX requires two programmable power supplies, one for VDD (2.5V to 6.0V recommended) and one for VPP (13 ± 0.25V). Both supplies should have a minimum resolution of 0.25V.

The PIC17CXX uses an intelligent algorithm. The algorithm calls for program verification at VDDmin as well as VDDmax. Verification at VDDmin guarantees good “erase margin”. Verification at VDDmax guarantees good “program margin”. Three times (3X) additional pulses will increase program margin then beyond VDD (max.) and insure safe operation in user system.

PIN DESCRIPTIONS (DURING PROGRAMMING): PIC17C42/42A/43/44

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
RA <0:4>	RA <0:4>	I	Necessary in programming mode
TEST	TEST	I	Must be set to “high” to enter programming mode
RB <7:0>	PAD <15:8>	I/O	Address & data: high byte
RC <7:0>	PAD <7:0>	I/O	Address & data: low byte
$\overline{\text{MCLR}}$/VPP	VPP	P	Programming Power
VDD	VDD	P	Power Supply
VSS	VSS	P	Ground

Legend: I = Input, O = Output, P = Power

The actual programming must be done with VDD in the VDDP range (4.75 - 5.25V).

VDDP=VDD range required during programming.

VDD min.=minimum operating VDD spec for the part.

VDD max.=maximum operating VCC spec for the part.

Programmers must verify the PIC17CXX at its specified VDDmax and VDDmin levels. Since Microchip may introduce future versions of the PIC17CXX with a broader VDD range, it is best that these levels are user selectable (defaults are ok).

Note:	Any programmer not meeting these requirements may only be classified as "prototype" or "development" programmer but not a "production" quality programmer.

2.0 PROGRAM MODE ENTRY

To execute the programming routine, the user must hold TEST pin high, RA2, RA3 must be low and RA4 must be high (after power-up) while keeping $\overline{MCLR}$ low and then raise $\overline{MCLR}$ pin from VIL to VDD or VPP. This will force FFE0h in the program counter and execution will begin at that location (the beginning of the boot code) following reset. Execution is forced to Internal mode by overriding the fuse configuration. The code protect bit is not overwritten. The program immediately polls PORT RB<7:0> to determine a branch address. Presenting E1h on PORT RB will cause the program to jump to and execute the “program/verify” routine.

Note:	The OSC must not have 72 osc clocks while the device $\overline{MCLR}$ is between VIL and VIHH.

All unused pins during programming are in high impedance state.

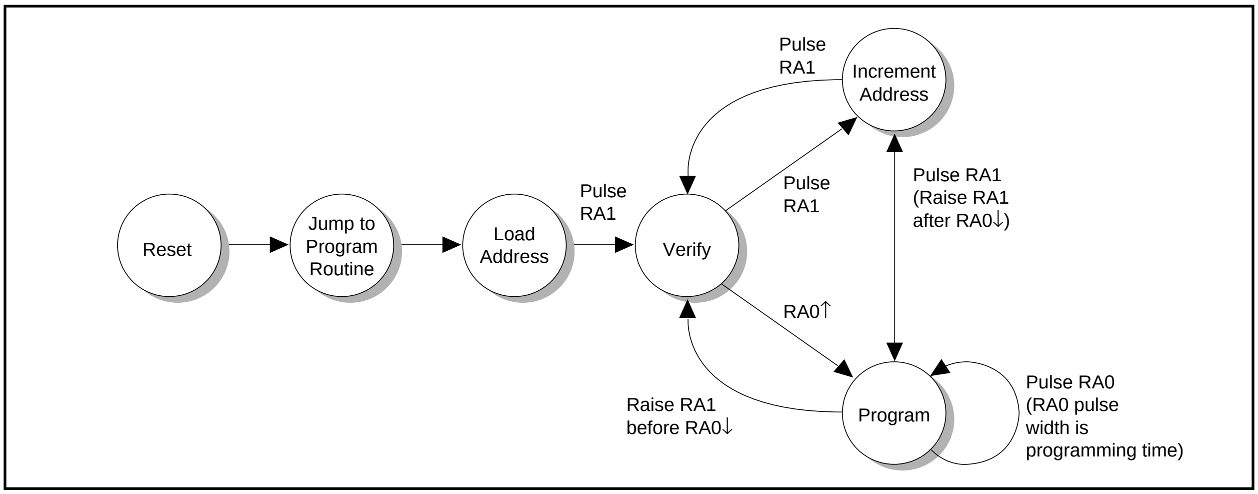
PORTB (RB) has internal weak pull-ups which are active during the programming mode. When TEST pin is high, Power-up timer (PWRT) and Oscillator Start-up Timers (OST) are disabled.

2.1 Program/Verify Mode

The program/verify mode is intended for full-feature programmers. This mode offers the following capabilities:

a) Load any arbitrary 16-bit address to start program and/or verify at that location.
b) Increment address to program/verify the next location.
c) Allows arbitrary length programming pulse width.
d) Following a “verify” allows option to program the same location or increment and verify the next location.
e) Following a “program” allows options to program the same location again, verify the same location or to increment and verify the next location.

FIGURE 2-1: PROGRAMMING/VERIFY STATE DIAGRAM



2.1.1 LOADING NEW ADDRESS

The program allows new address to be loaded right out of reset. A 16-bit address is presented on ports RB (high byte) and RC (low byte) and the RA1 is pulsed ($0 \rightarrow 1$, then $1 \rightarrow 0$). The address is latched on the rising edge of RA1. See timing diagrams for details. After loading an address, the program automatically goes into a “verify cycle”. To load a new address at any time, the PIC17C4X must be reset and the programming mode re-entered.

2.1.2 VERIFY (OR READ) MODE

“Verify mode” can be entered from “Load address” mode, “program mode” or “verify mode”. In verify mode pulsing RA1 will turn on PORTS RB and RC output drivers and output the 16-bit value from the current location. Pulsing RA1 again will increment location count and be ready for the next verify cycle. Pulsing RA0 will begin a program cycle.

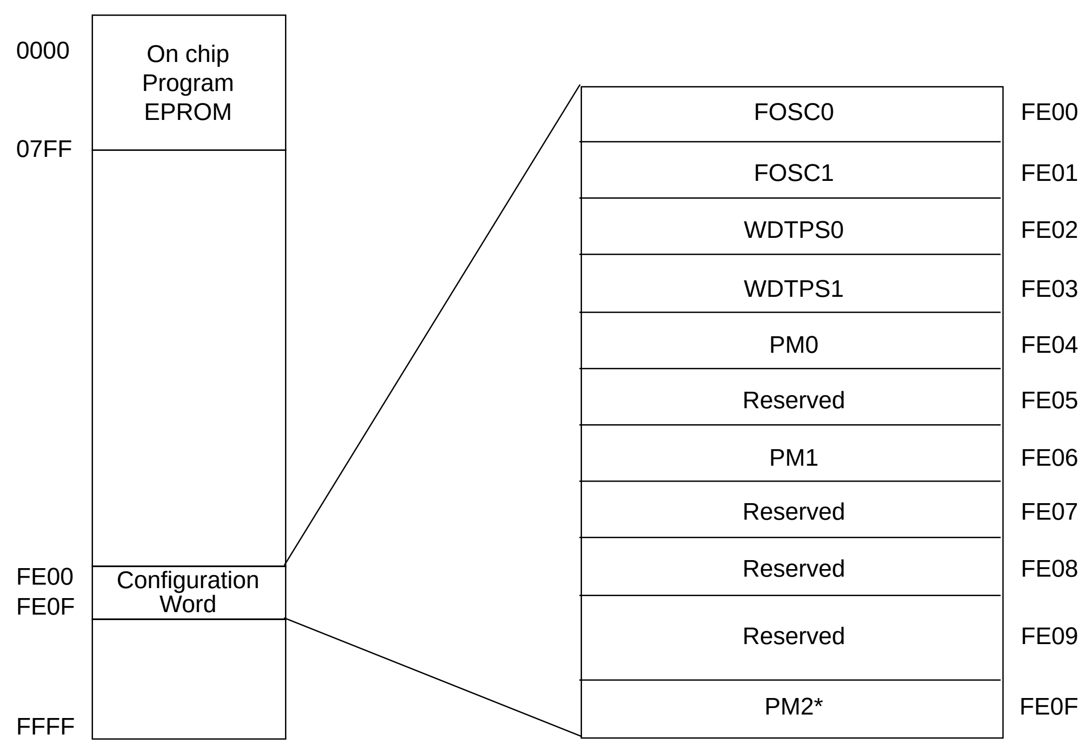
2.1.3 PROGRAM CYCLE

“Program cycle” is entered from “verify cycle” or program cycle” itself. After a verify, pulsing RA0 will begin a program cycle. 16-bit data must be presented on PORTS RB (high byte) and RC (low byte) before RA0 is raised.

The data is sampled 3 T_{CY} cycles after the rising edge of RA0. Programming continues for the duration of RA0 pulse.

At the end of programming the user can choose one of three different routes. If RA1 is kept low and RA0 is pulsed again, the same location will be programmed again. This is useful for applying over programming pulses. If RA1 is raised before RA0 falling edge, then a verify cycle is started without address increment. Raising RA1 after RA0 goes low will increment address and begin verify cycle on the next address.

FIGURE 2-2: PIC17C4X PROGRAM MEMORY MAP



*This location does not exist for PIC17C42

3.0 PROGRAMMING SPECIFICATIONS

FIGURE 3-1: PROGRAMMING ROUTINE FLOWCHART

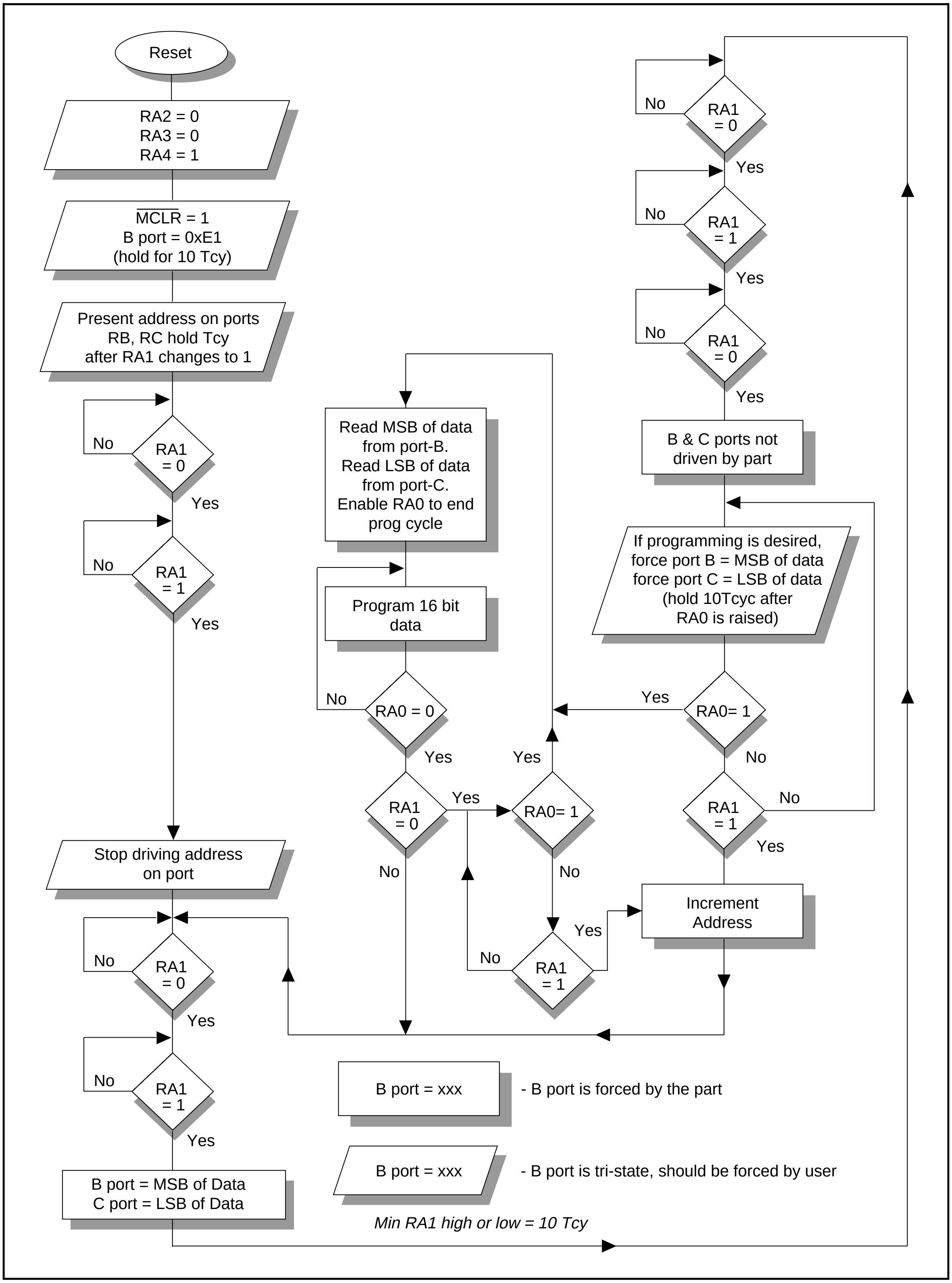


FIGURE 3-2: RECOMMENDED PROGRAMMING ALGORITHM FOR USER EPROM

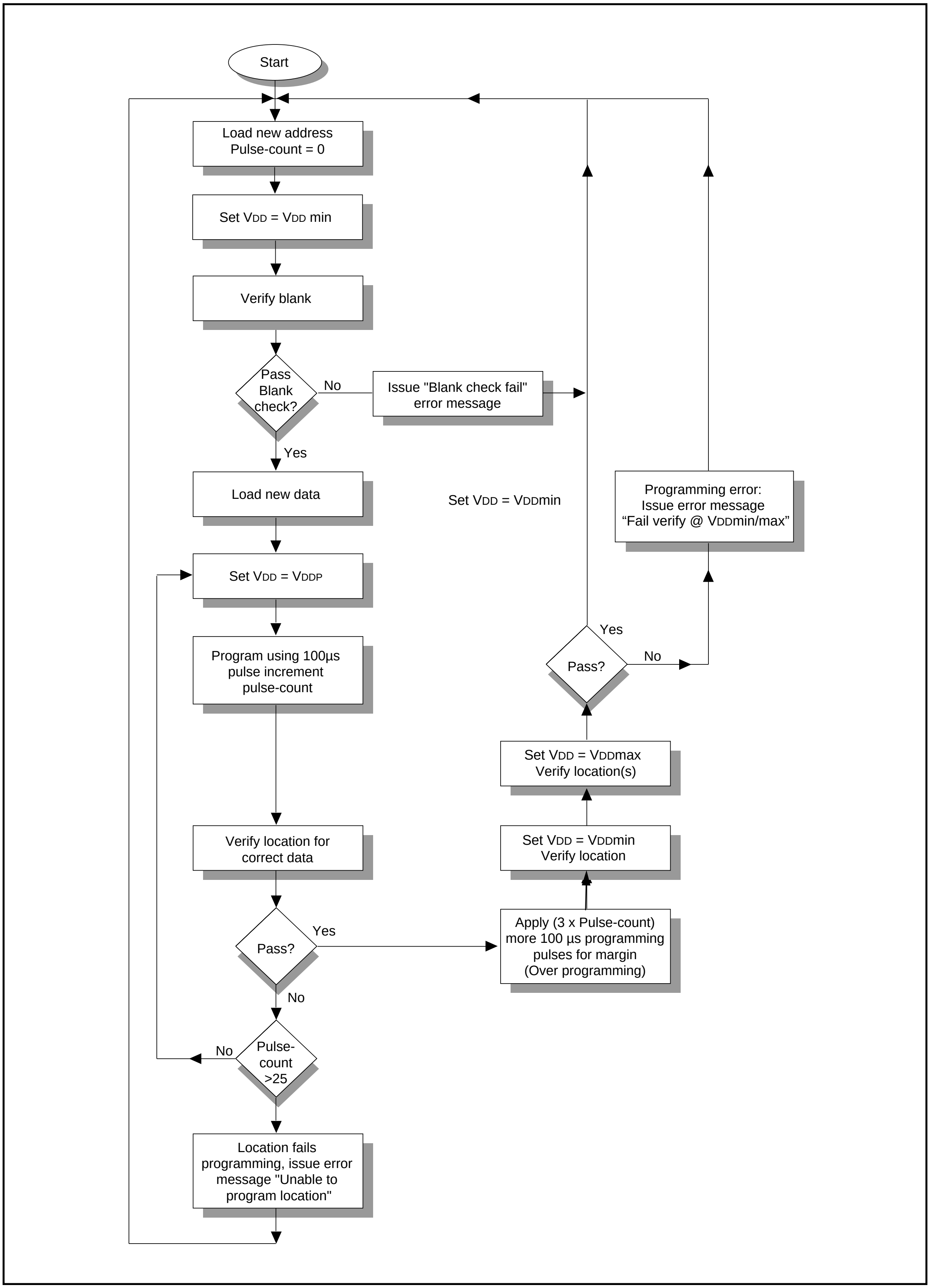
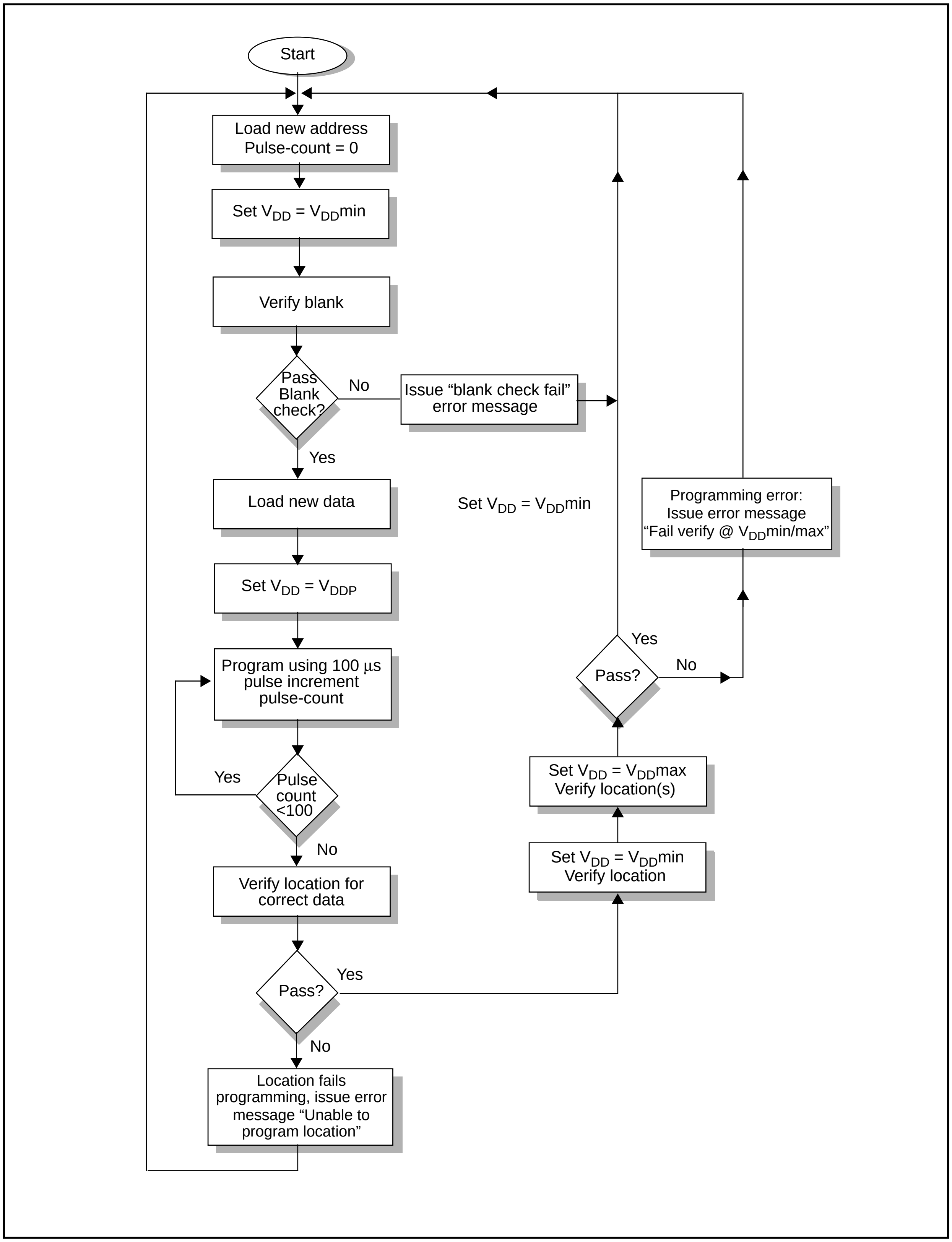


FIGURE 3-3: RECOMMENDED PROGRAMMING ALGORITHM FOR CONFIGURATION WORDS



4.0 CONFIGURATION WORD

Configuration bits are mapped into program memory. Each bit is assigned one memory location. In erased condition a bit will read as '1'. To program a bit, the user needs to write to the memory address. The data is immaterial; the very act of writing will program the bit. The configuration word locations are shown in Table 4-3. **The programmer should not program the reserved locations to avoid unpredictable results and to be compatible with future variations of the PIC17C4X**. **It is also mandatory that configuration locations are programmed in the strict order starting from the first location (0xFE00) and ending with the last (0xFE0F). Unpredictable results may occur if the sequence is violated.**

4.1 Reading Configuration Word

The PIC17CXX has seven configuration locations (see Table 4-1). These locations can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. Any write to a configuration location, regardless of the data, will program that configuration bit. Reading any configuration location between 0xFE00 and 0xFE07 will place the low byte of the configuration word (see Table 4-2) into PAD<7:0> (PORTC). PAD<15:8> (PORTB) will be set to 0xFF. Reading a configuration location between 0xFE08 and 0xFE0F will place the high byte of the configuration word into PAD<7:0> (PORTC). PAD<15:8> (PORTB) will be set to 0xFF.

TABLE 4-1: CONFIGURATION BIT PROGRAMMING LOCATIONS

Bit	Address
FOSC0	0xFE00
FOSC1	0xFE01
WDTPS0	0xFE02
WDTPS1	0xFE03
PM0	0xFE04
PM1	0xFE06
PM2†	0xFE0F

†This location does not exist on the PIC17C42.

TABLE 4-2: READ MAPPING OF CONFIGURATION BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	—	PM1	—	PM0	WDTPS1	WDTPS0	FOSC1	FOSC0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	PM2*	—	—	—	—	—	—	—

—=Unused

PM<2:0>, Processor Mode Select bits

111 = Microprocessor mode
110 = Microcontroller mode
101 = Extended Microcontroller mode
000 = Code protected microcontroller mode

WDTPS<1:0>, WDT Prescaler Select bits.

11 = WDT enabled, postscaler = 0
10 = WDT enabled, postscaler = 256
01 = WDT enabled, postscaler = 64
00 = WDT disabled, 16-bit overflow timer

FOSC<1:0>, Oscillator Select bits

11 = EC oscillator
10 = XT oscillator
01 = RC oscillator
00 = LF oscillator

* This bit does not exist on PIC17C42.

4.2 Embedding Configuration Word Information in the Hex File

To allow portability of code, a PIC17C4X programmer is required to read the configuration word locations from the hex file when loading the hex file. If configuration word information was not present in the hex file then a simple warning message may be issued. Similarly, while saving a hex file, all configuration word information must be included. An option to not include the configuration word information may be provided. When embedding configuration word information in the hex file, it should be to address FE00h.

Microchip Technology Inc. feels strongly that this feature is important for the benefit of the end customer.

TABLE 4-3: CONFIGURATION WORD

PIC17C42

To code protect:

- Protect all memory `XXXXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

PIC17C42A

To code protect:

- Protect all memory `0XXXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

PIC17CR42

To code protect:

- Protect all memory `0XXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

PIC17C43

To code protect:

- Protect all memory `0XXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

PIC17CR43

To code protect:

- Protect all memory `0XXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

PIC17C44

To code protect:

- Protect all memory `0XXXXXXX0X0XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0xFE00)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
All memory	Read Scrambled, Write Disabled*	Read Unscrambled, Write Enabled

Legend: X = Don’t care

*Write to on-chip EPROM memory is disabled. The only way these locations can be programmed is if a TABLWT instruction is issued from an “on-chip” program memory space to program an on-chip memory location.

4.3 CHECKSUM COMPUTATION

The checksum is calculated by summing the following:

- The contents of all program memory locations
- The configuration word, appropriately masked
- Masked ID locations (when applicable)

The least significant 16 bits of this sum is the checksum.

The following table describes how to calculate the checksum for each device. Note that the checksum calculation differs depending on the code protect setting. Since the program memory locations read out differently depending on the code protect setting, the table describes how to manipulate the actual program memory values to simulate the values that would be read from a protected device. When calculating a checksum by reading a device, the entire program memory can simply be read and summed. The configuration word and ID locations can always be read.

Note that some older devices have an additional value added in the checksum. This is to maintain compatibility with older device programmer checksums.

TABLE 4-4: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	0xC0DE at 0 and max address
PIC17C42	MP mode	SUM[0x000:0x7FF] + CFGW & 0x005F + 0xFFA0	0xF7FF	0x79BD
	MC mode	SUM[0x000:0x7FF] + CFGW & 0x005F + 0xFFA0	0xF7EF	0x79AD
	EMC mode	SUM[0x000:0x7FF] + CFGW & 0x005F + 0xFFA0	0xF7BF	0x797D
	PMC mode	SUM_XNOR8[0x000:0x7FF] + CFGW & 0x005F + 0xFFA0	0xF7AF	0xBB73
PIC17C42A	MP mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF95F	0x7B1D
	MC mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF94F	0x7B0D
	EMC mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF91F	0x7ADD
	PMC mode	SUM_XNOR8[0x000:0x7FF] + CFGW & 0x015F	0xF80F	0xBBD3
PIC17CR42	MP mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF95F	0x7B1D
	MC mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF94F	0x7B0D
	EMC mode	SUM[0x000:0x7FF] + CFGW & 0x015F	0xF91F	0x7ADD
	PMC mode	SUM_XNOR8[0x000:0x7FF] + CFGW & 0x015F	0xF80F	0xBBD3
PIC17C43	MP mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF15F	0x731D
	MC mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF14F	0x730D
	EMC mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF11F	0x72DD
	PMC mode	SUM_XNOR8[0x000:0xFFF] + CFGW & 0x015F	0xF00F	0xB3D3
PIC17CR43	MP mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF15F	0x731D
	MC mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF14F	0x730D
	EMC mode	SUM[0x000:0xFFF] + CFGW & 0x015F	0xF11F	0x72DD
	PMC mode	SUM_XNOR8[0x000:0xFFF] + CFGW & 0x015F	0xF00F	0xB3D3
PIC17C44	MP mode	SUM[0x000:0x1FFF] + CFGW & 0x015F	0xE15F	0x631D
	MC mode	SUM[0x000:0x1FFF] + CFGW & 0x015F	0xE14F	0x630D
	EMC mode	SUM[0x000:0x1FFF] + CFGW & 0x015F	0xE11F	0x62DD
	PMC mode	SUM_XNOR8[0x000:0x1FFF] + CFGW & 0x015F	0xE00F	0xA3D3

Legend: CFGW = Configuration Word
SUM[a:b] = [Sum of locations a to b inclusive]
SUM_XNOR8(a:b) = [Sum of 8-bit wide XNOR copied into upper and lower byte, of locations a to b inclusive]
*Checksum = [Sum of all the individual expressions] **MODULO** [0xFFFF]
+ = Addition
& = Bitwise AND

5.0 AC/DC CHARACTERISTICS TIMING REQUIREMENTS FOR PROGRAM/VERIFY TEST MODE

Standard Operating Conditions

Operating Temperature: +10°C ≤ TA ≤ +70°C, unless otherwise stated, (25°C is recommended)
Operating Voltage: 4.5V ≤ VDD ≤ 5.25V, unless otherwise stated.

Parameter No.	Sym.	Characteristic	Min.	Typ.	Max.	Units	Conditions/Comments
PD1	VDDP	Supply voltage during programming	4.75	5.0	5.25	V	
PD2	IDDP	Supply current during programming			50	mA	Freq = 10MHz, VDD = 5.5V Note 3
PD3	VDDV	Supply voltage during verify	VDD min.		VDD max.	V	Note 2
PD4	VPP	Voltage on VPP/$\overline{MCLR}$ pin during programming	12.75		13.25	V	Note 1
PD6	IPP	Programming current on VPP/$\overline{MCLR}$ pin		25	50	mA	Note 3
P1	FOSCP	Osc/clockin frequency during programming	4		10	MHz	
P2	TCY	Instruction cycle	1		0.4	µs	TCY = 4/FOSCP
P3	TirV2tsH	RA0, RA1, RA2, RA3, RA4 setup before TEST↑	1			µs	
P4	TtsH2mcH	TEST↑ to $\overline{MCLR}$↑	1			µs	
P5	TbcV2irH	RC<7:0>, RB<7:0> valid to RA1 or RA0↑ :Address/Data input setup time	0			µs	
P6	TirH2bcl	RA1 or RA0↑ to RB<7:0>, RC<7:0> invalid ; Address data hold time;	10 TCY			µs	
P7	T0ckiL2rbcZ	RT↓ to RB<7:0>, RC<7:0> high impedance			8 TCY		
P8	T0ckiH2bcV	RA1↑ to data out valid			10 TCY		
P9	Tprog	Programming pulse width	10	100	1000	µs	
P10	TirH2irL	RA0, RA1 high pulse width	10 TCY			µs	
P11	TirL2irH	RA0, RA1 low pulse width	10 TCY			µs	
P12	T0ckiV2inL	RA1↑ before INT↓ (to go from prog cycle to verify w/o increment)	0			µs	
P13	TinL2rtl	RA1 valid after RA0 (to select increment or no increment going from program to verify cycle	10 TCY			µs	
P14	Tvpps	VPP setup time before RA0↑	100			µs	Note 1
P15	Tvpph	VPP hold time after INT↓	0			µs	Note 1
P16	TvdV2tsH	VDD stable to TEST↑	10			ms	
P17	TrbV2mcH	RB input (E1h) valid to VPP/$\overline{MCLR}$↑	0			µs	
P18	TmcH2rbI	RB input (E1h) hold after VPP/$\overline{MCLR}$↑	10 TCY			ns	
P19	TvpL2vdL	VDD power down after VPP power down	10			ms	

Note 1: VPP/$\overline{MCLR}$ pin must only be equal to or greater than VDD at times other than programming.
Note 2: Program must be verified at the minimum and maximum VDD limits for the part.
Note 3: These parameters are for design guidance only and are not tested nor characterized.

FIGURE 5-1: PROGRAMMING AND VERIFY TIMINGS I

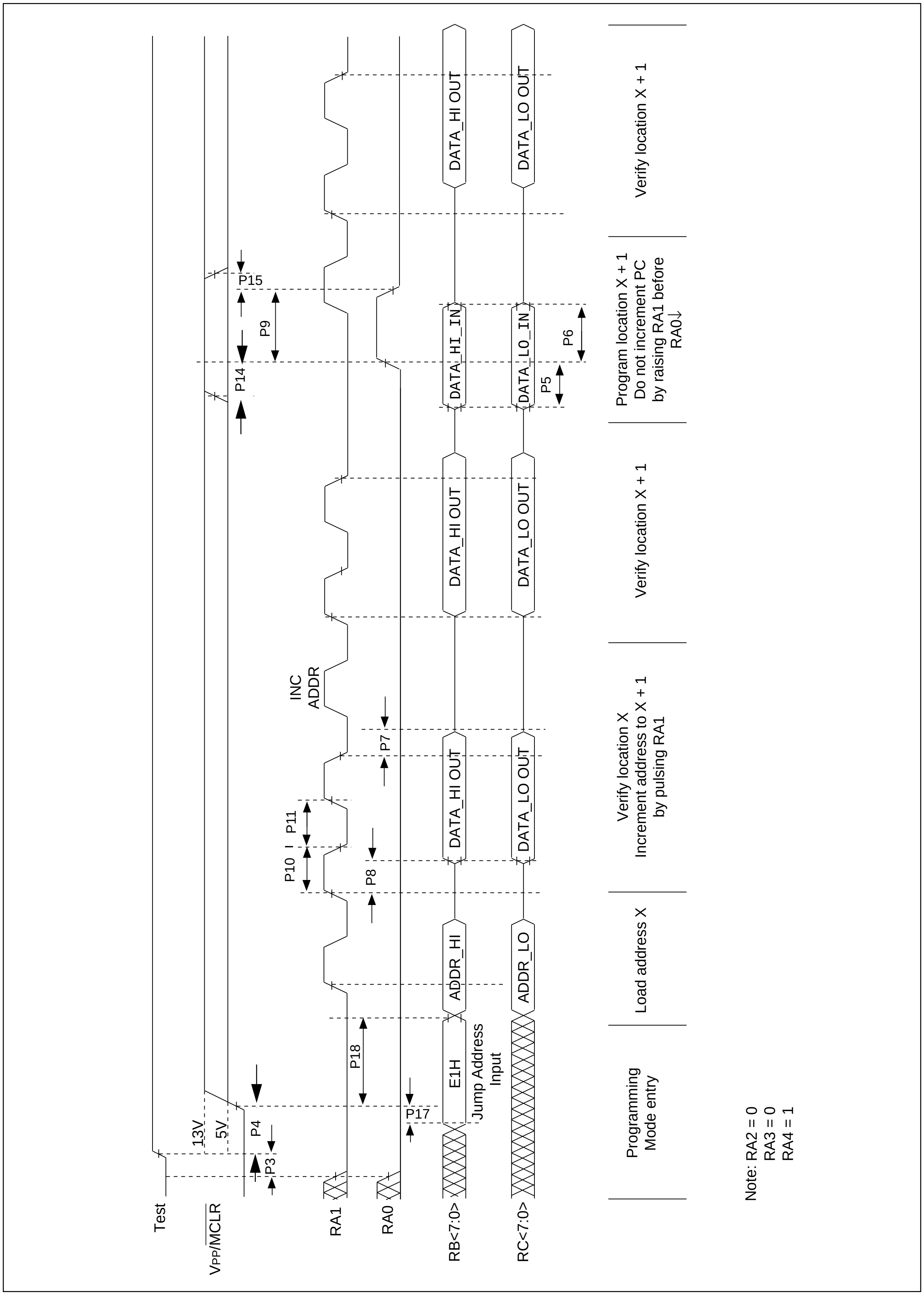


FIGURE 5-2: PROGRAMMING AND VERIFY TIMINGS II

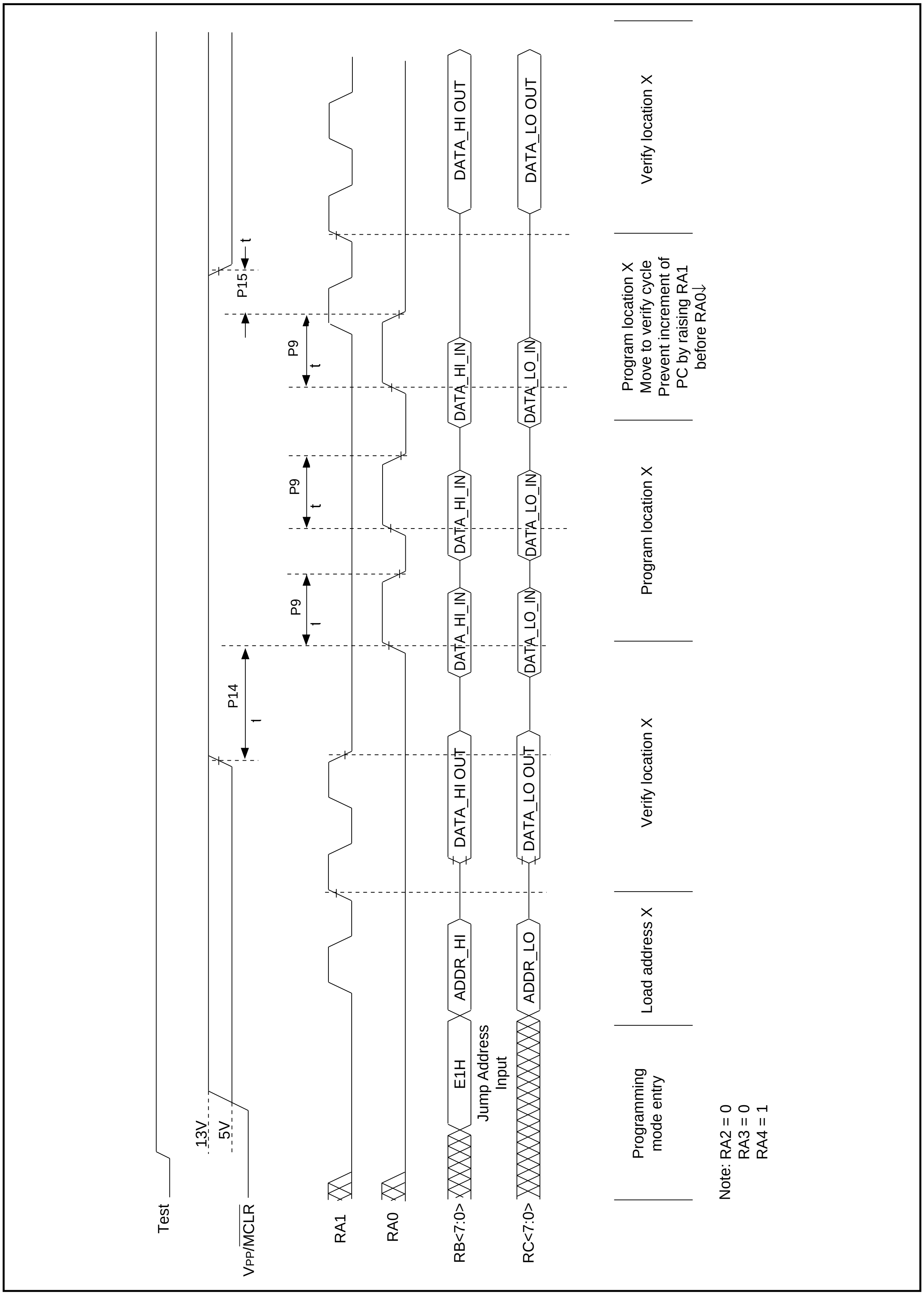


FIGURE 5-3: PROGRAMMING AND VERIFY TIMINGS III

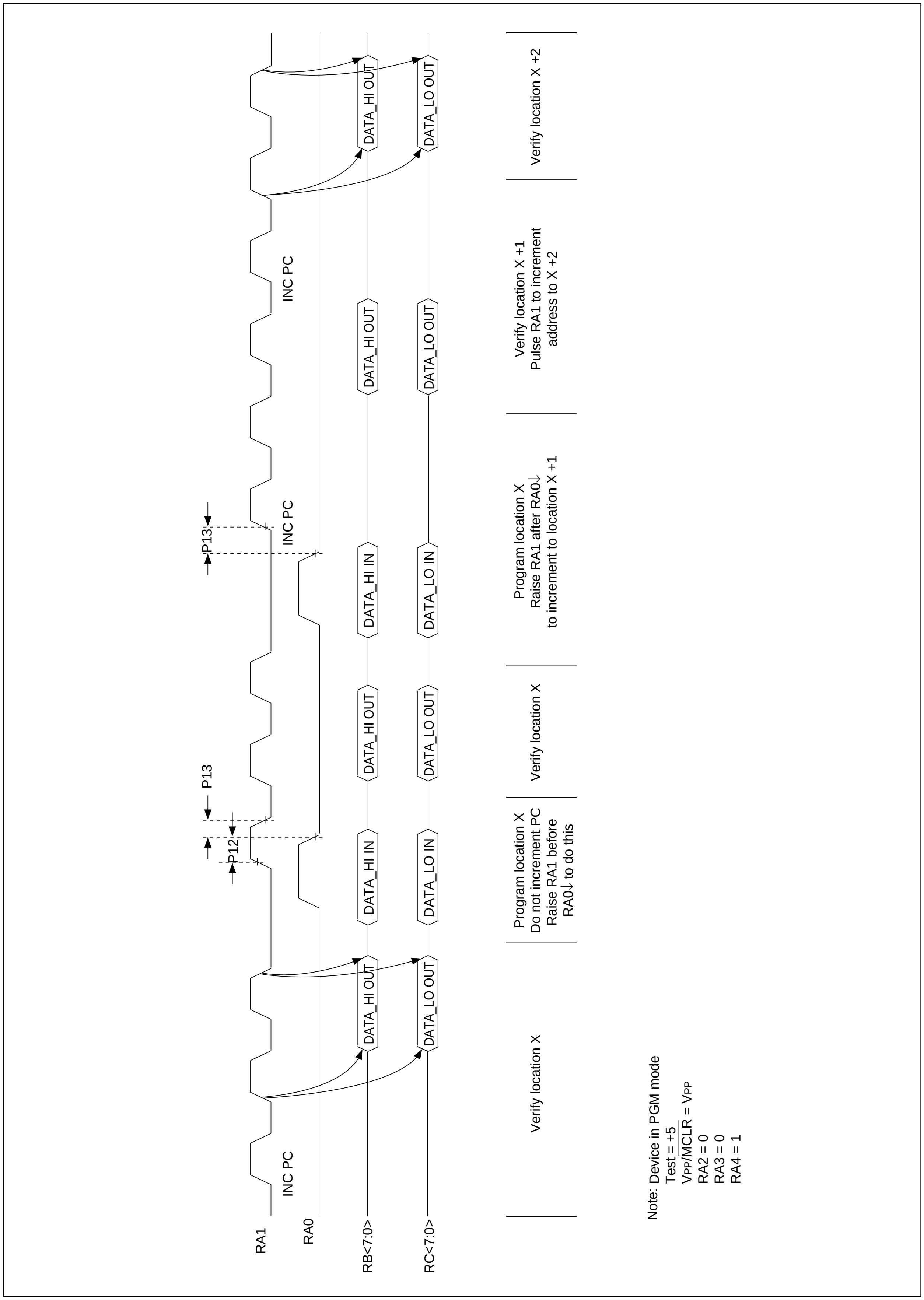
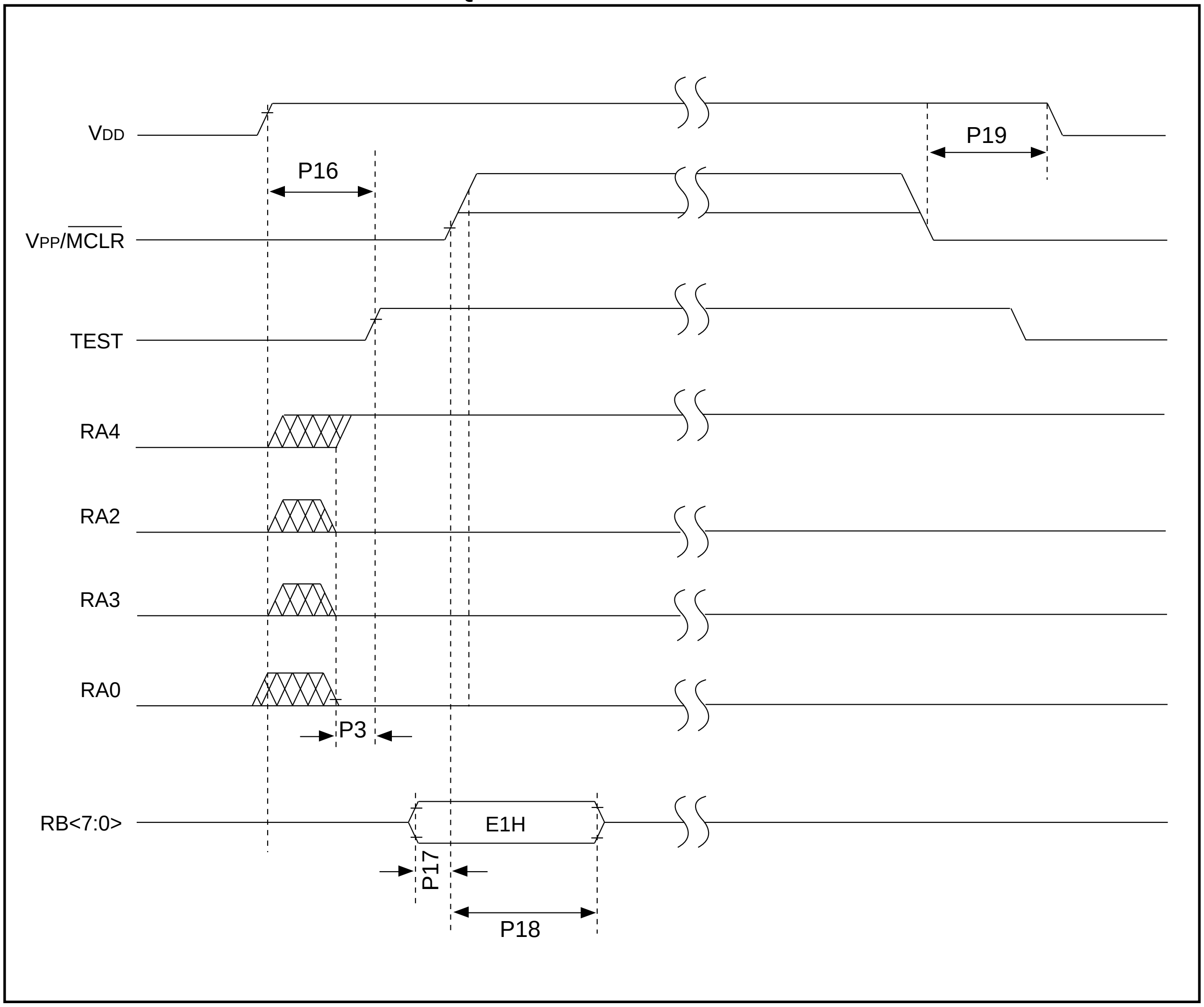


FIGURE 5-4: POWER-UP/DOWN SEQUENCE FOR PROGRAMMING



NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com/

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
Suite 150
Two Prestige Place
Miamisburg, OH 45342
Tel: 513 291-1654 Fax: 513 291-9175

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

AMERICAS *(continued)*

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905 405-6279 Fax: 905 405-6253

ASIA/PACIFIC

Hong Kong
Microchip Technology
Rm 3801B, Tower Two
Metroplaza,
223 Hing Fong Road,
Kwai Fong, N.T., Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 1 628 850303 Fax: 44 1 628 850178

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041, Agrate Brianza, Milan Italy
Tel: 39 39 689 9939 Fax: 39 39 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

5/10/96