PIC18CXXX

In-Circuit Serial Programming™ for PIC18CXXX OTP MCUs

This document includes the programming specifications for the following devices:

- PIC18C242
- PIC18C252
- PIC18C442
- PIC18C452
- PIC18C658
- PIC18C858

1.0 PROGRAMMING THE PIC18CXXX

The PIC18CXXX can be programmed using a serial method while in users' system, allowing increased design flexibility. This programming specification applies to PIC18CXXX devices in all package types.

1.1 Hardware Requirements

The PIC18CXXX requires two programmable power supplies, one for V_{DD} and one for V_{PP}. Both supplies should have a minimum resolution of 0.25 V.

1.2 Programming Mode

The programming mode for the PIC18CXXX allows programming of user program memory, special locations used for ID, and the configuration words for the PIC18CXXX.

Pin Diagrams

The pin diagram for the PIC18CXX2 family is shown below in Figure 1-1. Pin diagrams for the PIC18CXX8 family are provided in Figure 1-4 through Figure 1-7.

FIGURE 1-1: PIC18CXX2 FAMILY PIN DIAGRAM

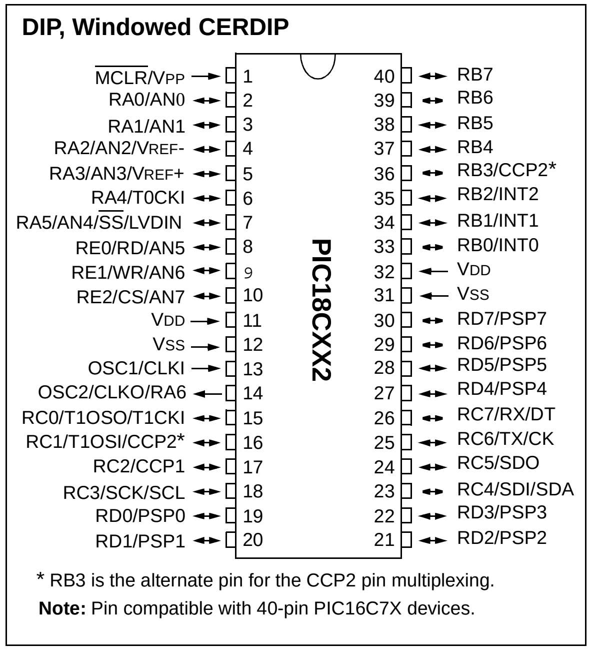
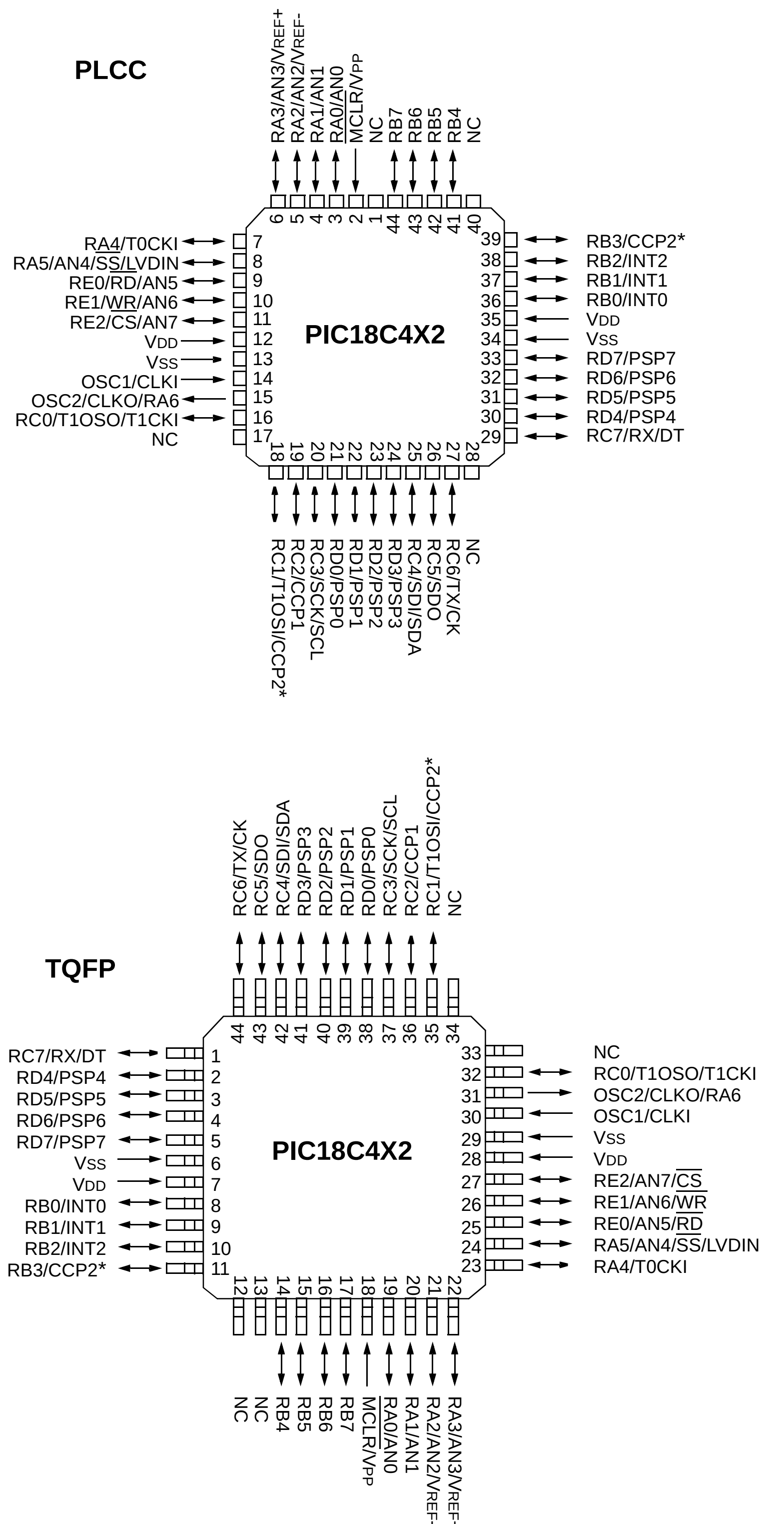


TABLE 1-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC18C242/252/442/452/658/858

Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
$\overline{MCLR}/V_{PP}$	V_{PP}	P	Programming Power
V_{DD}	V_{DD}	P	Power Supply
V_{SS}	V_{SS}	P	Ground
RB6	RB6	I	Serial Clock
RB7	RB7	I/O	Serial Data

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

FIGURE 1-2: PIC18C4X2 44-PIN PLCC AND 44-PIN TQFP DIAGRAMS



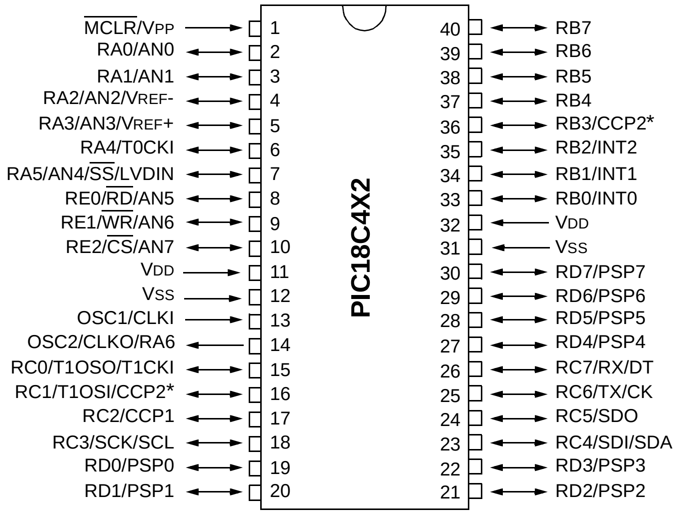
* RB3 is the alternate pin for the CCP2 pin multiplexing.

Note: Pin compatible with 44-pin PIC16C7X devices.

In-Circuit Serial Programming™ (ICSP) is a trademark of Microchip Technology Inc.

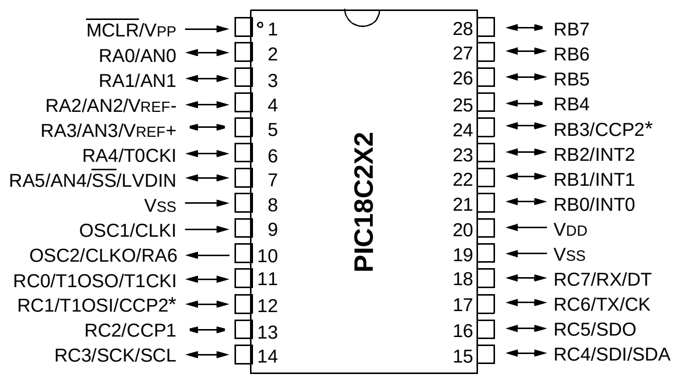
FIGURE 1-3: PIC18C4X2 40-PIN DIP, JW AND PIC18C2X2 28-PIN DIP, SOIC, JW DIAGRAMS

DIP, JW



Note: Pin compatible with 40-pin PIC16C7X devices.

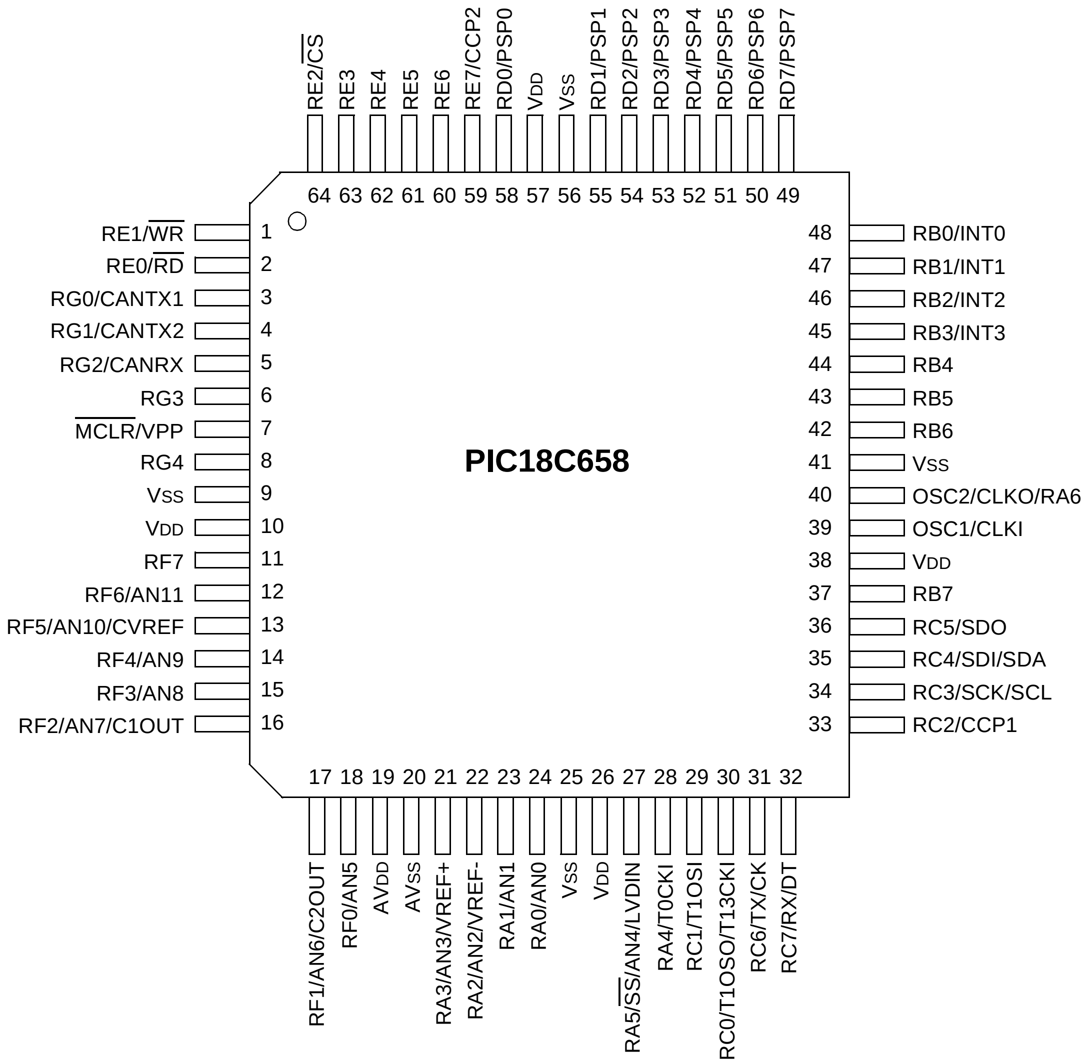
DIP, SOIC, JW



* RB3 is the alternate pin for the CCP2 pin multiplexing.

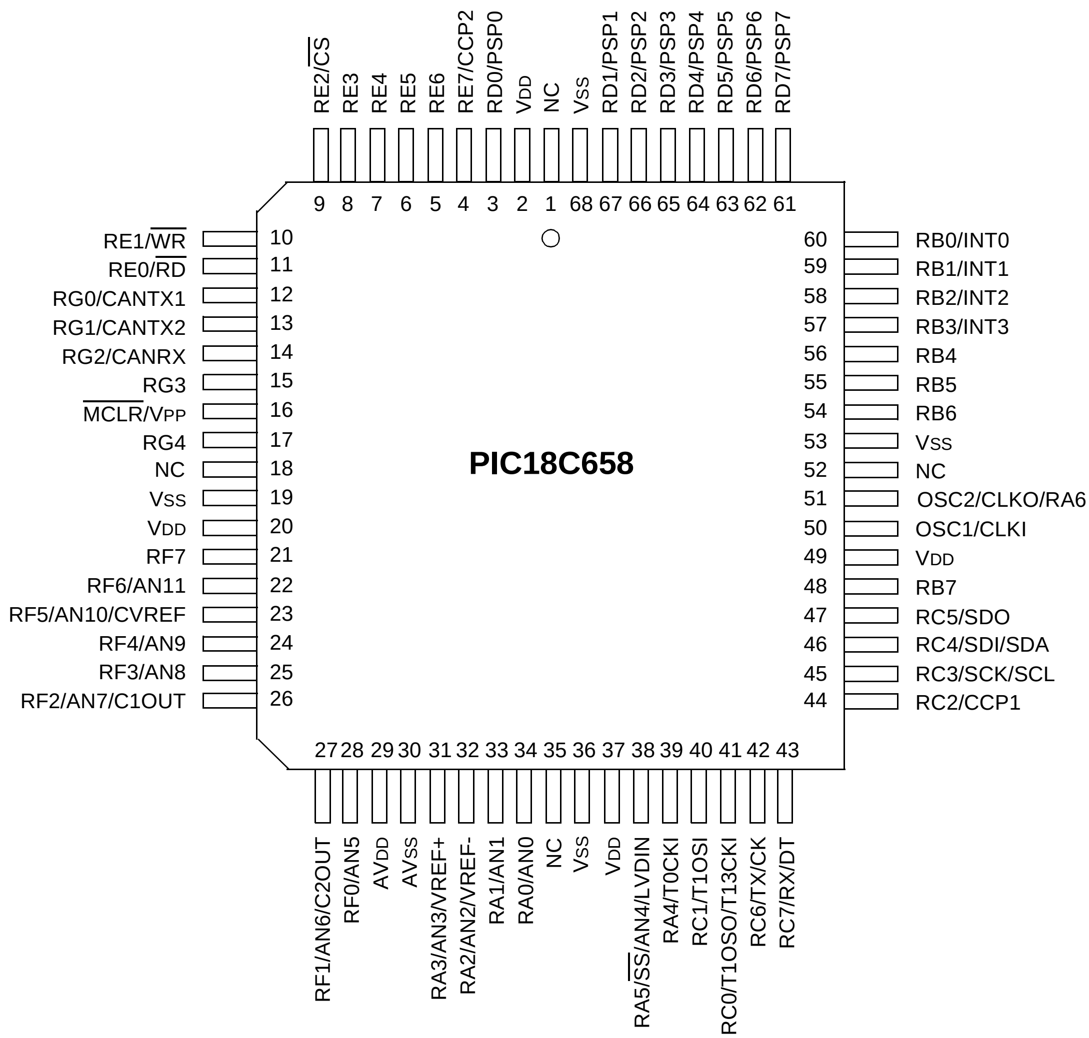
Note: Pin compatible with 28-pin PIC16C7X devices.

FIGURE 1-4: PIC18C658 64-PIN TQFP DIAGRAM



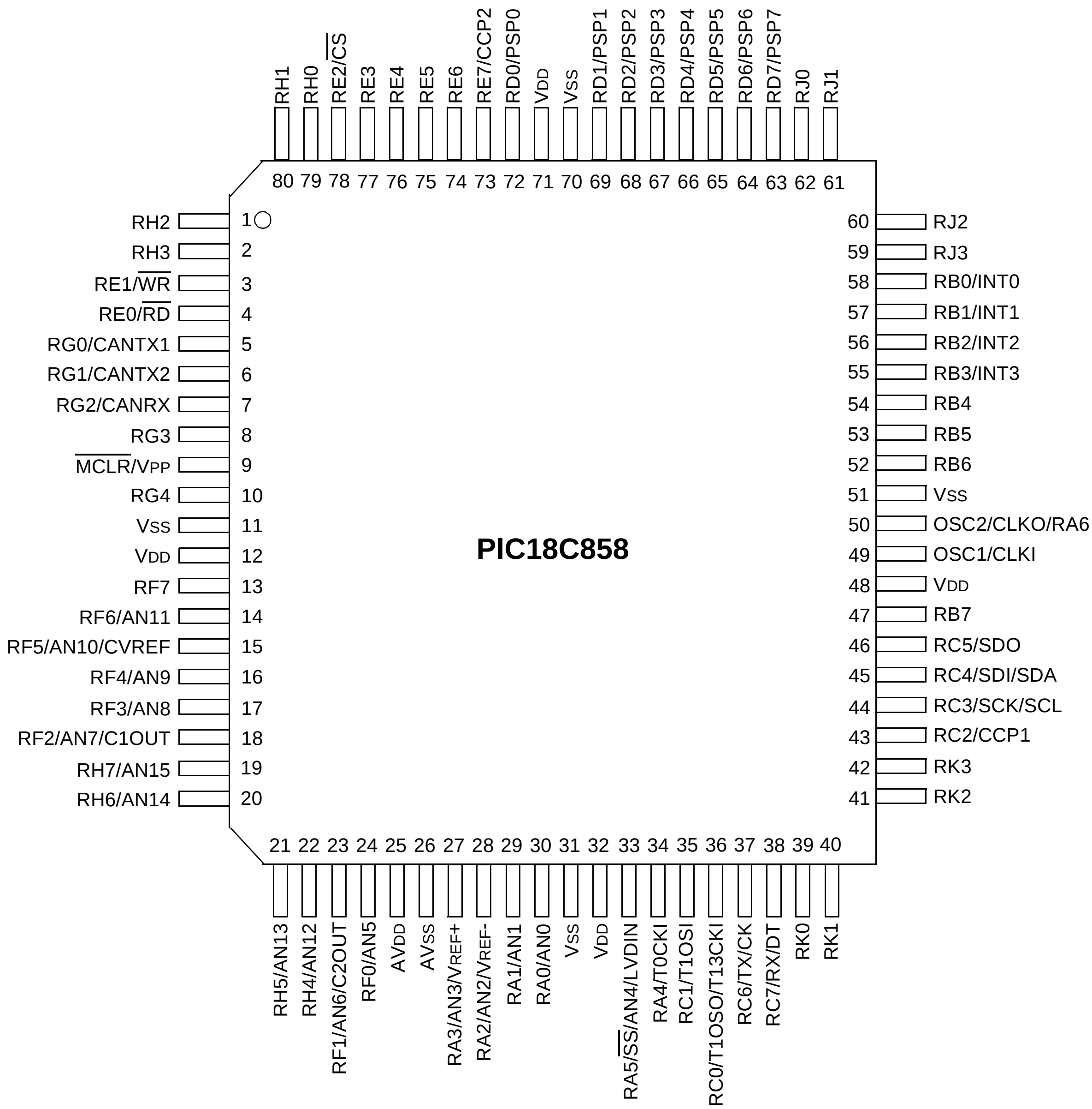
Note: All PIC18C658 and PIC18C858 package outlines are compatible with PIC17C7XX.

FIGURE 1-5: PIC18C658 68-PIN PLCC DIAGRAM



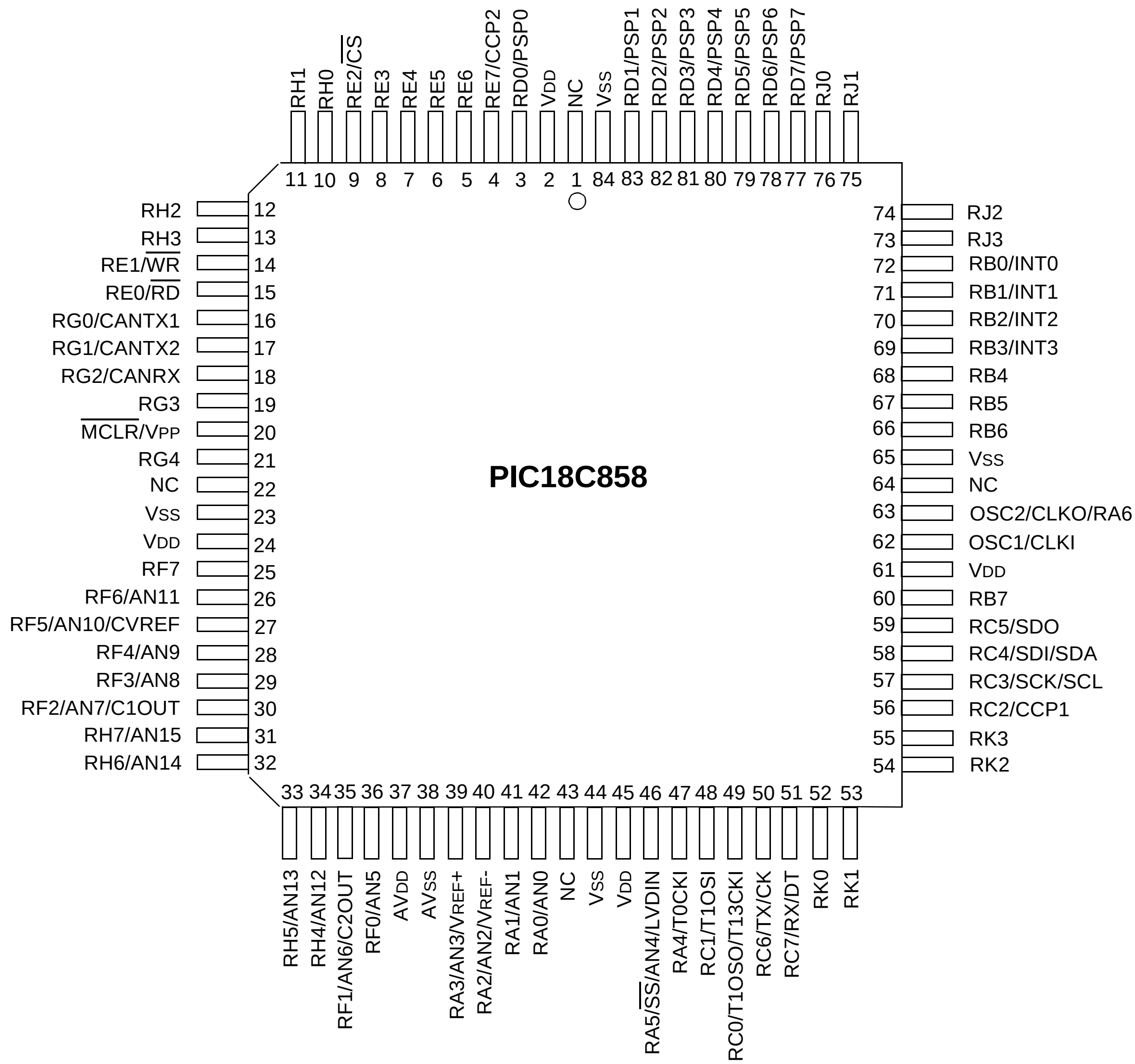
Note: All PIC18C658 and PIC18C858 package outlines are compatible with PIC17C7XX.

FIGURE 1-6: PIC18C858 80-PIN TQFP DIAGRAM



Note: All PIC18C658 and PIC18C858 package outlines are compatible with PIC17C7XX.

FIGURE 1-7: PIC18C858 84-PIN PLCC DIAGRAM



Note: All PIC18C658 and PIC18C858 package outlines are compatible with PIC17C7XX.

2.0 IN-CIRCUIT SERIAL PROGRAMMING (ICSP) MODE

2.1 Introduction

Serial programming mode is entered by asserting $\overline{MCLR}/V_{PP} = V_{IHH}$ and RB6, RB7 = 0V.

Instructions are fed into the CPU serially on RB7, and are shifted in on the rising edge of the serial clock presented on RB6. RB7 serves as data out, as well. Programming and verification are performed by executing `TBLRD` and `TBLWT` instructions. The address pointer to the program memory is simply the table pointer. The address pointer can be incremented and decremented by executing table reads and writes with auto-decrement and auto-increment.

2.2 ICSP Operation

In ICSP mode, instruction execution takes place through a serial interface using RB6 and RB7. RB7 is used to shift in instructions and shift out data from the TABLAT register. RB6 is used as the serial shift clock and the CPU execution clock. Instructions and data are shifted LSb first.

In this mode, all instructions are shifted serially, loaded into the instruction register, and executed. No program fetching occurs from internal or external program memory. 8-bit data bytes are read from the TABLAT register via the same serial interface.

2.2.1 4-BIT SERIAL INSTRUCTIONS

A set of 4-bit instructions are provided for ICSP mode, so the most common instructions used for ICSP can be fetched quickly, and reduce the amount of time required to program a device. The 4-bit opcode is shifted in while the previously fetched instruction executes. The 4-bit instruction contains the lower 4-bits of an instruction opcode. The upper 12 bits default to all 0's. Instructions with all 0's in the upper byte of the instruction word are by default, considered special instructions. The serial instructions are decoded as shown in Table 2-1.

TABLE 2-1: SPECIAL INSTRUCTIONS FOR SERIAL INSTRUCTION EXECUTION AND ICSP

Mnemonic, Operands	Description	Cycles	4-Bit Opcode	Status Affected
`NOP`	No Operation (Shift in16-bit instruction)	1	`0000`	None
`TBLRD *`	Table Read (no change to TBLPTR)	2	`1000`	None
`TBLRD *+`	Table Read (post-increment TBLPTR)	2	`1001`	None
`TBLRD *-`	Table Read (post-decrement TBLPTR)	2	`1010`	None
`TBLRD +*`	Table Read (pre-increment TBLPTR)	2	`1011`	None
`TBLWT *`	Table Write (no change to TBLPTR)	2	`1100`	None
`TBLWT *+`	Table Write (post-increment TBLPTR)	2	`1101`	None
`TBLWT *-`	Table Write (post-decrement TBLPTR)	2	`1110`	None
`TBLWT +*`	Table Write (pre-increment TBLPTR)	2	`1111`	None

Legend: Refer to the PIC18CXXX Data Sheet (DS39026 or DS30475) for opcode field descriptions.

Note: All special instructions not included in this table are decoded as `NOP`'s.

2.2.2 INITIAL SERIAL INSTRUCTION OPERATION

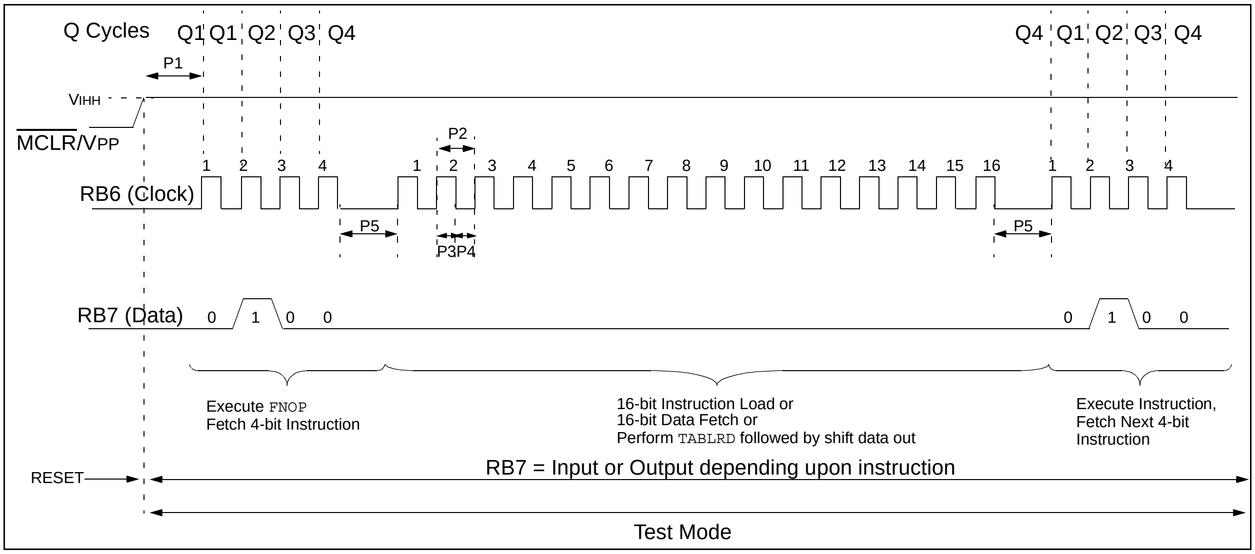
Upon ICSP mode entry, the CPU is idle. The execution of the CPU is governed by a state machine. While the first instruction is being clocked in, a forced `NOP` `(FNOP)` is executed.

Following the `FNOP` instruction execution and shifting in of the next instruction, the serial state machine will do one of three things, depending upon the 4-bit instruction fetched:

1. If the instruction fetched was a `NOP`, the state machine will suspend the CPU awaiting a 16-bit wide instruction to be shifted in.
2. If the instruction is a `TBLWT`, the state machine suspends the CPU from execution while sixteen bits of data are shifted in as data for the `TBLWT` instruction.
3. If the instruction is a `TBLRD`, then execution of the `TBLRD` instruction begins immediately for eight clock cycles, followed by eight clock cycles where the contents of the TABLAT register is shifted out onto RB7.

Once sixteen clock cycles have elapsed, the next 4-bit instruction is fetched while the current instruction is executed. Each instruction type is described in later sections.

FIGURE 2-1: SERIAL INSTRUCTION TIMING AFTER RESET



2.2.3 NOP SERIAL INSTRUCTION EXECUTION

The NOP serial instruction is used to allow execution of all other instructions not included in Table 2-1. When the NOP instruction is fetched, the serial execution state machine suspends the CPU for 16 clock cycles. During these 16 clock cycles, all 16-bits of an instruction are fed into the CPU and the NOP instruction is discarded. Once all 16 bits have been shifted in, the state machine will allow the instruction to be executed for the next four clock cycles.

Note:	16-bit TBLWT and TBLRD instructions are not permitted. They will cause timing problems with the serial state machine. If the user wishes to perform a TBLWT or TBLRD instruction, it must be performed as a 4-bit instruction.

2.2.4 ONE CYCLE 16-BIT INSTRUCTIONS

If the instruction fetched is a one cycle instruction, then the instruction operation will be completed in the four clock cycles following the instruction fetched. During instruction execution, the next 4-bit serial instruction is fetched (see Figure 2-2).

FIGURE 2-2: SERIAL INSTRUCTION TIMING FOR 1-CYCLE, 16-BIT INSTRUCTIONS

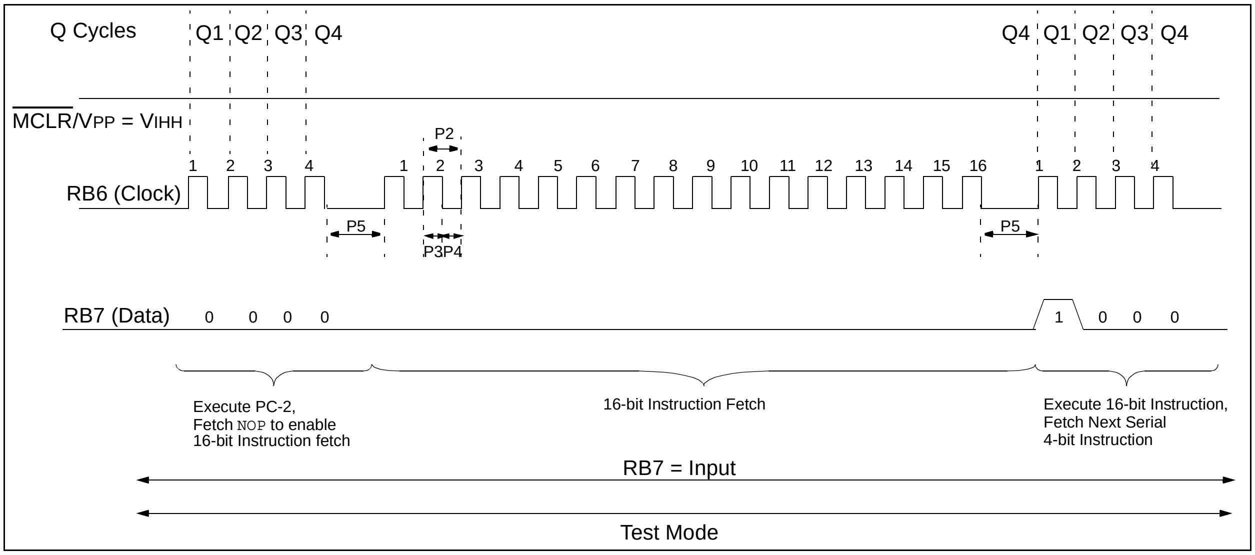


FIGURE 2-3: 16-BIT, 1-CYCLE SERIAL INSTRUCTION FLOW AFTER RESET

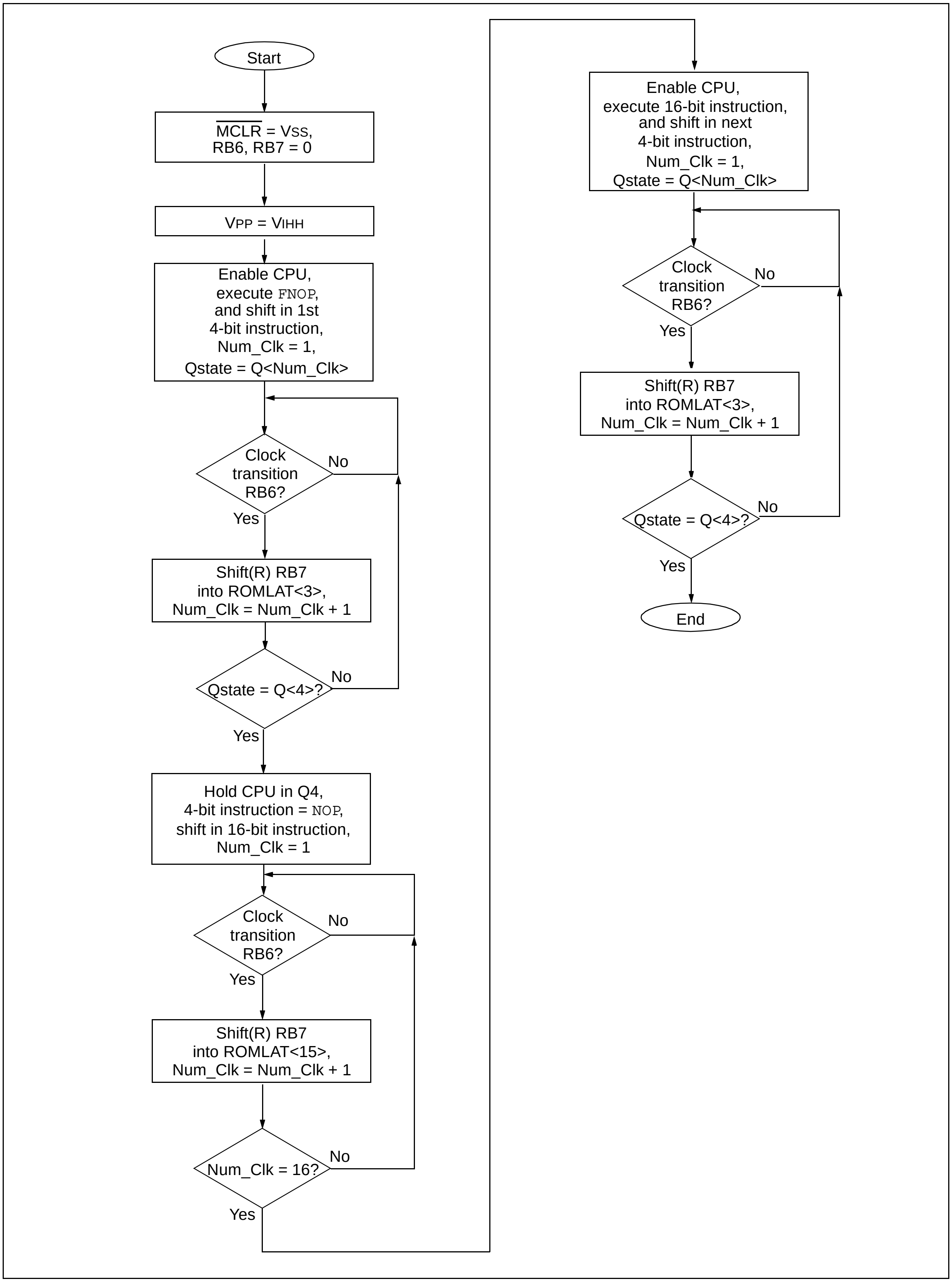
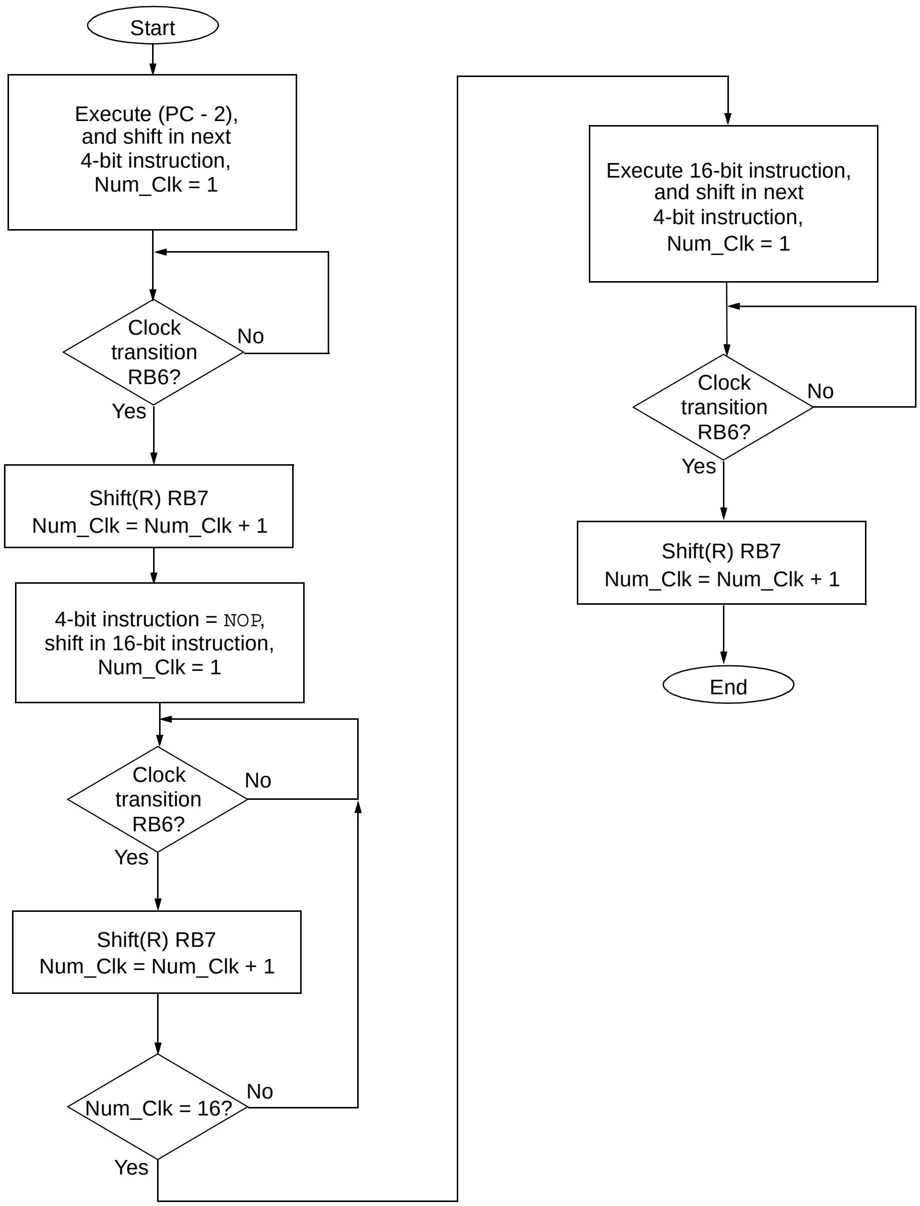


FIGURE 2-4: 16-BIT, 1-CYCLE SERIAL INSTRUCTION FLOW

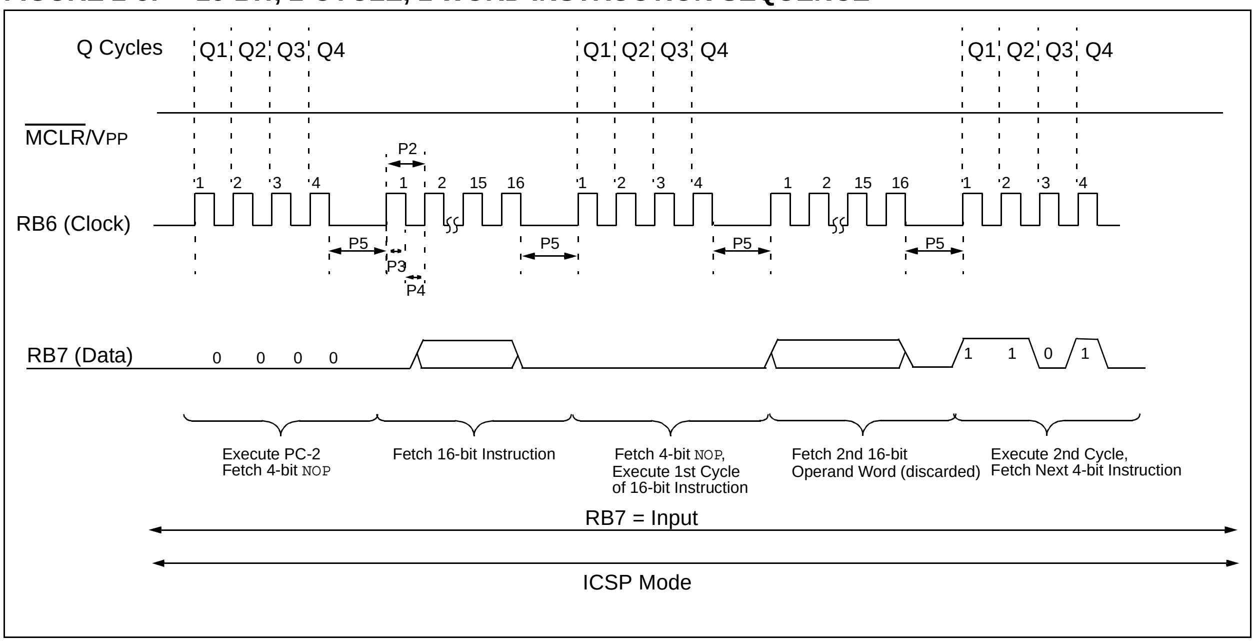


2.3 Serial Instruction Execution For Two-Cycle, One Word Instructions

When a `NOP` instruction is fetched, the serial execution state machine suspends the CPU for 16 clock cycles. During these 16 clock cycles, all 16-bits of an instruction are fed in and the `NOP` instruction is discarded.

If the instruction fetched is a two cycle, one word instruction, the instruction operation will require a second "dummy fetch" to be performed before the instruction execution can be completed. The first cycle of the instruction will be executed in the four clock cycles following the instruction fetched. During the first cycle of instruction execution, the next 4-bit serial instruction is fetched. To perform the second half of the two cycle instruction, this 4-bit instruction must be a `NOP`, so the state machine will remain idle for the second half of the instruction. Following the fetch of the second `NOP`, the state machine will shift 16 bits of data that will be discarded. After the 16 bits of data are shifted in, the state machine will release the CPU, and allow it to perform the second half of the two cycle instruction. During the second half of the two cycle instruction execution, the next 4-bit instruction is loaded (see Figure 2-5).

FIGURE 2-5: 16-BIT, 2-CYCLE, 1 WORD INSTRUCTION SEQUENCE



2.4 Serial Instruction Execution For Two Word, Two-Cycle Instructions

After a NOP instruction is fetched, the serial execution state machine suspends the CPU in the Q4 state for 16 clock cycles. During these 16 clock cycles, all 16 bits of an instruction are fed in and the NOP instruction is discarded.

If the 16-bit instruction fetched is a two cycle, two word instruction, the instruction operation will require a second operand fetch to be performed before the instruction execution can be completed. The first cycle of the instruction will be executed in the four clock cycles following the 16-bit instruction fetch. During the first cycle of instruction execution, the next 4-bit serial instruction is fetched. To perform the second half of the two cycle instruction, this 4-bit instruction must also be a NOP, so the state machine will remain idle for the second half of the instruction. Following the fetch of the second NOP, the state machine will shift 16 bits of data that will be used as an operand for the two cycle instruction. After the 16 bits of data are shifted in, the state machine will release the CPU, and allow it to execute the second half of the two cycle instruction. During the second half of the two cycle instruction execution, the next 4-bit instruction is loaded (see Figure 2-6).

FIGURE 2-6: 16-BIT, 2-CYCLE, 2 WORD INSTRUCTION SEQUENCE

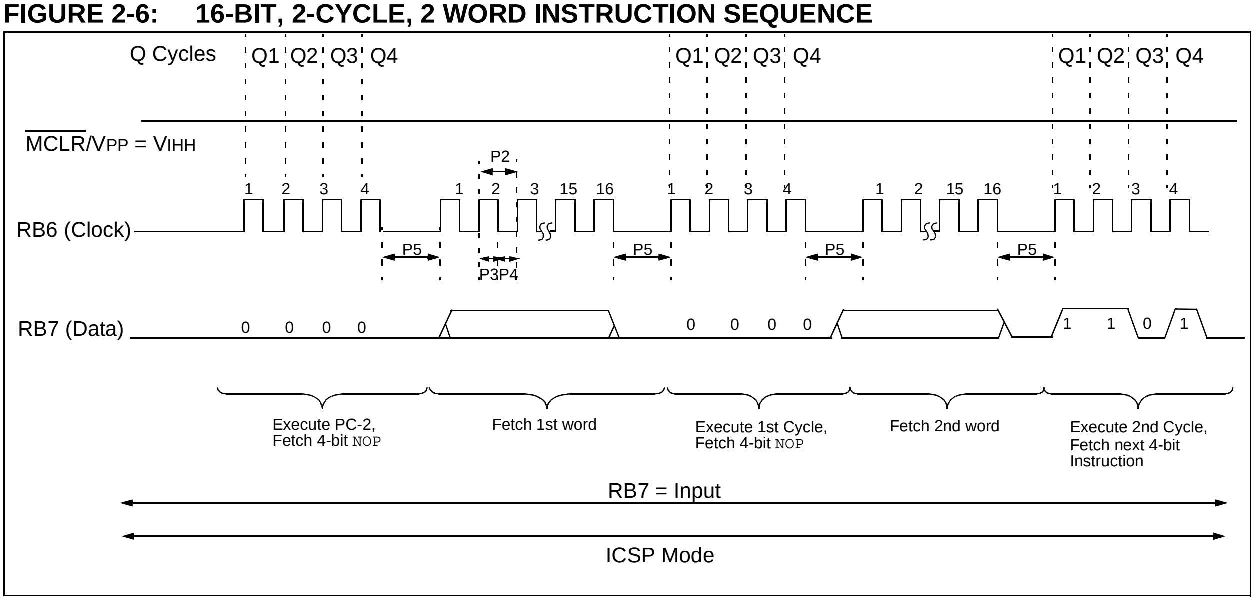


FIGURE 2-7: 16-BIT, 2-CYCLE, 2 WORD SERIAL INSTRUCTION FLOW AFTER RESET

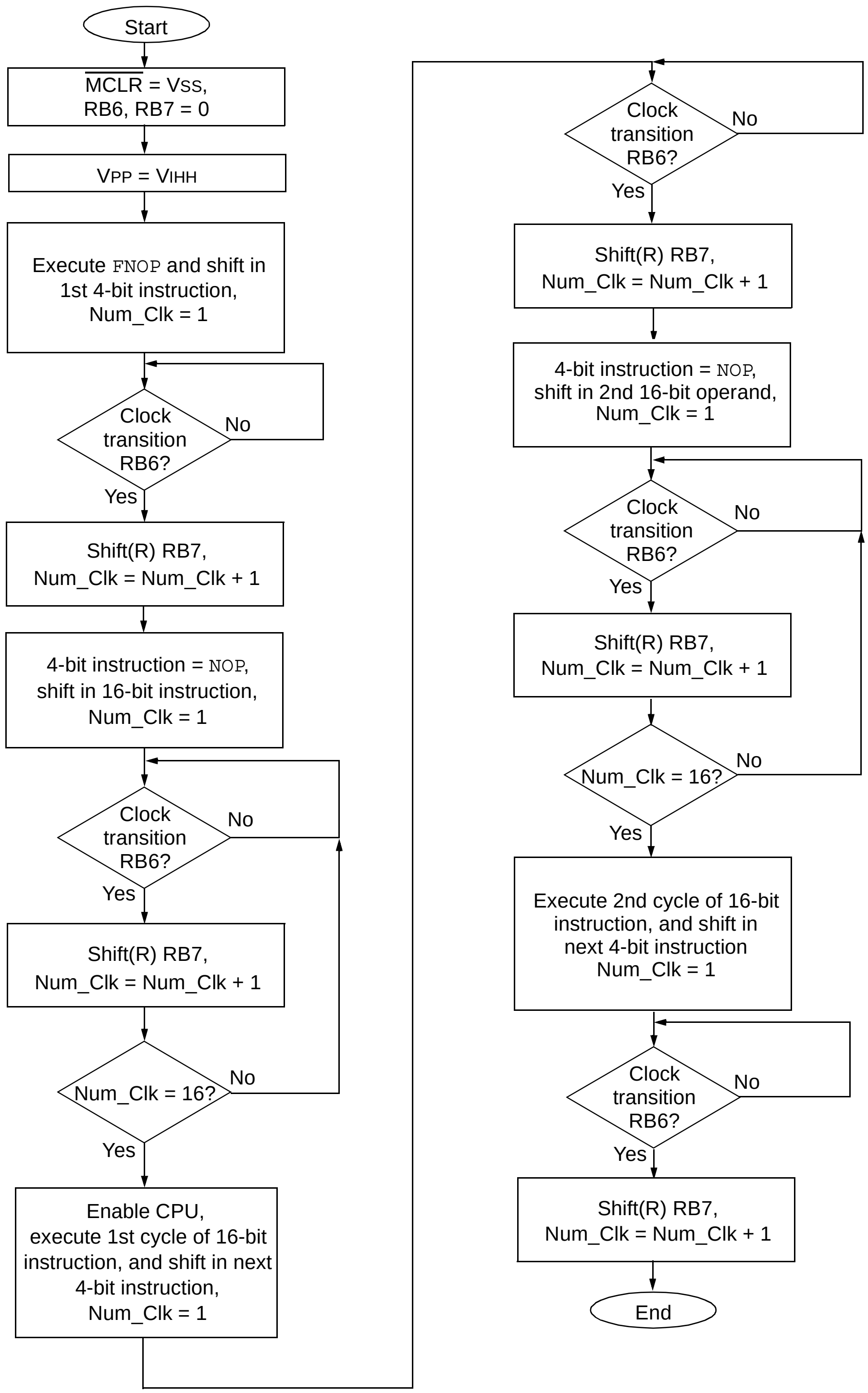
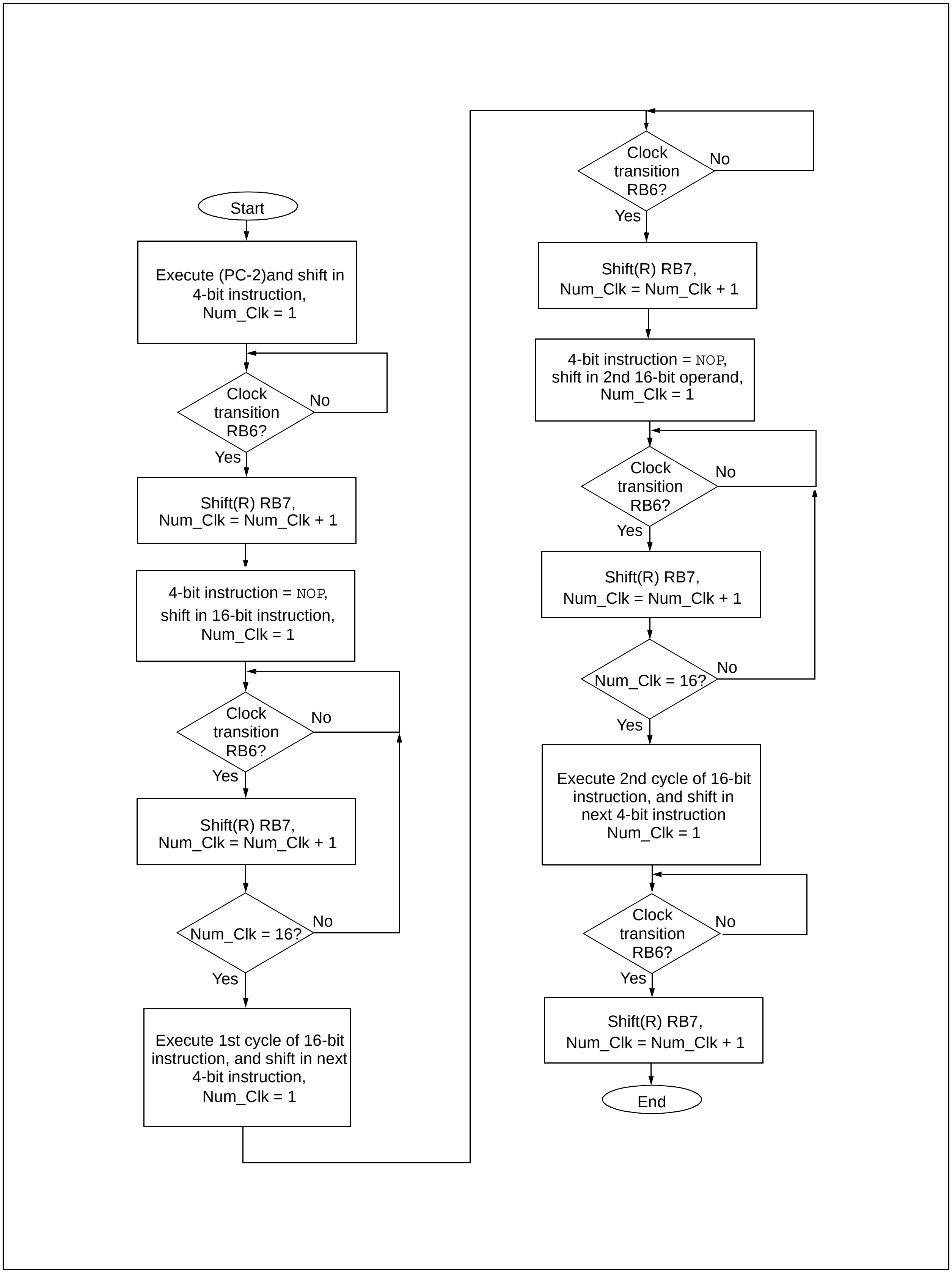


FIGURE 2-8: 16-BIT, 2-CYCLE, 2 WORD SERIAL INSTRUCTION FLOW



2.5 TBLWT Instruction

The TBLWT instruction is a special two-cycle instruction.

All forms of TBLWT instructions (post/pre-increment, post-decrement, etc.) are encoded as 4-bit special instructions. This is useful as TBLWT instructions are used repeatedly in ICSP mode. A 4-bit instruction will minimize the total number of clock cycles required to perform programming algorithms.

The TBLWT instruction sequence operates as follows:

1. The 4-bit TBLWT instruction is read in by the state machine on RB7 during the four clock cycle execution of the instruction fetched previous to the TBLWT (which is a FNOP if the TBLWT is executed following a RESET).
2. Once the state machine recognizes that the instruction fetched is a TBLWT, the state machine proceeds to fetch in the 16 bits of data that will be written into the program memory location pointed to by the TBLPTR.
3. The state machine releases the CPU to execute the first cycle of the TBLWT instruction while the first four bits of the 16-bit data word are shifted in. After the first cycle of TBLWT instruction has completed, the state machine shifts in the remaining 12 of the 16 bits of data. The data word will not be used until the second cycle of the instruction.
4. After all 16 bits of data are shifted in and the first cycle of the TBLWT is performed, the CPU will execute the second cycle of the TBLWT operation, programming the current memory location with the 16-bit value. The next instruction following the TBLWT instruction is shifted in during the execution of the second cycle (see Figure 2-9).

The TBLWT instruction is used in ICSP mode to program the EPROM array. When writing a 16-bit value to the EPROM, ID locations, or configuration locations, the device, RB6 must be held high for the appropriate programming time during the TBLWT instruction, as specified by parameter P9.

When RB6 is asserted low, the device will cease programming the specified location.

After RB6 is asserted low, RB6 is held low for the time specified by parameter P10, to allow high voltage discharge of the program memory array.

FIGURE 2-9: TBLWT INSTRUCTION SEQUENCE

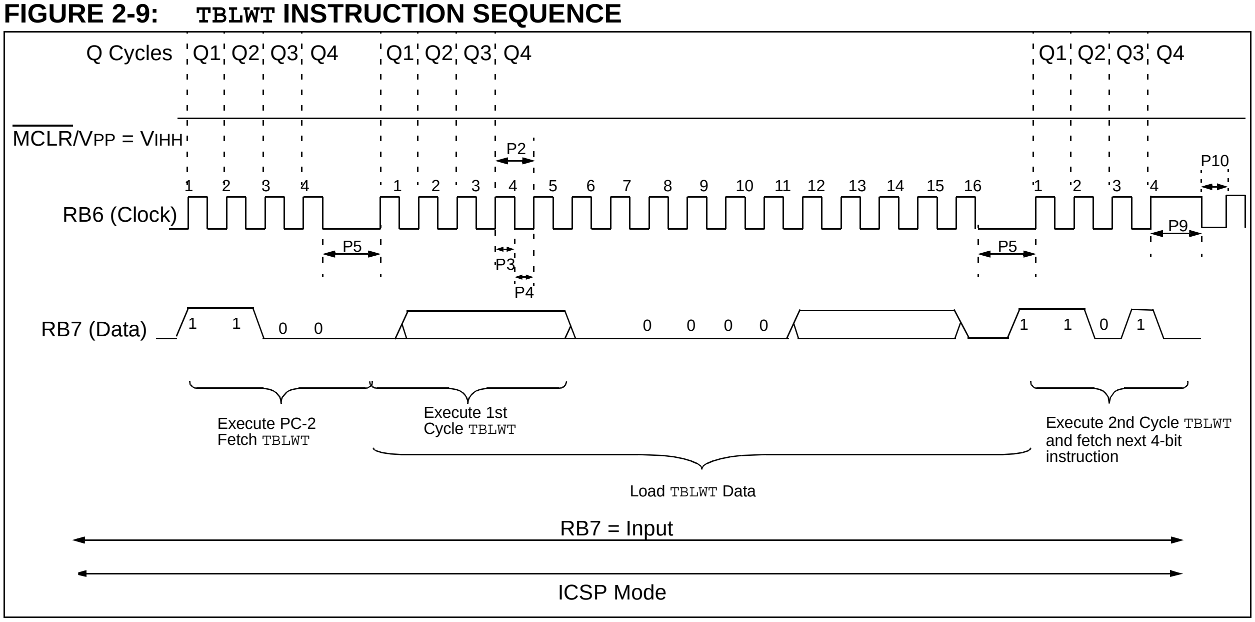


FIGURE 2-10: `TBLWT` SERIAL INSTRUCTION FLOW AFTER RESET

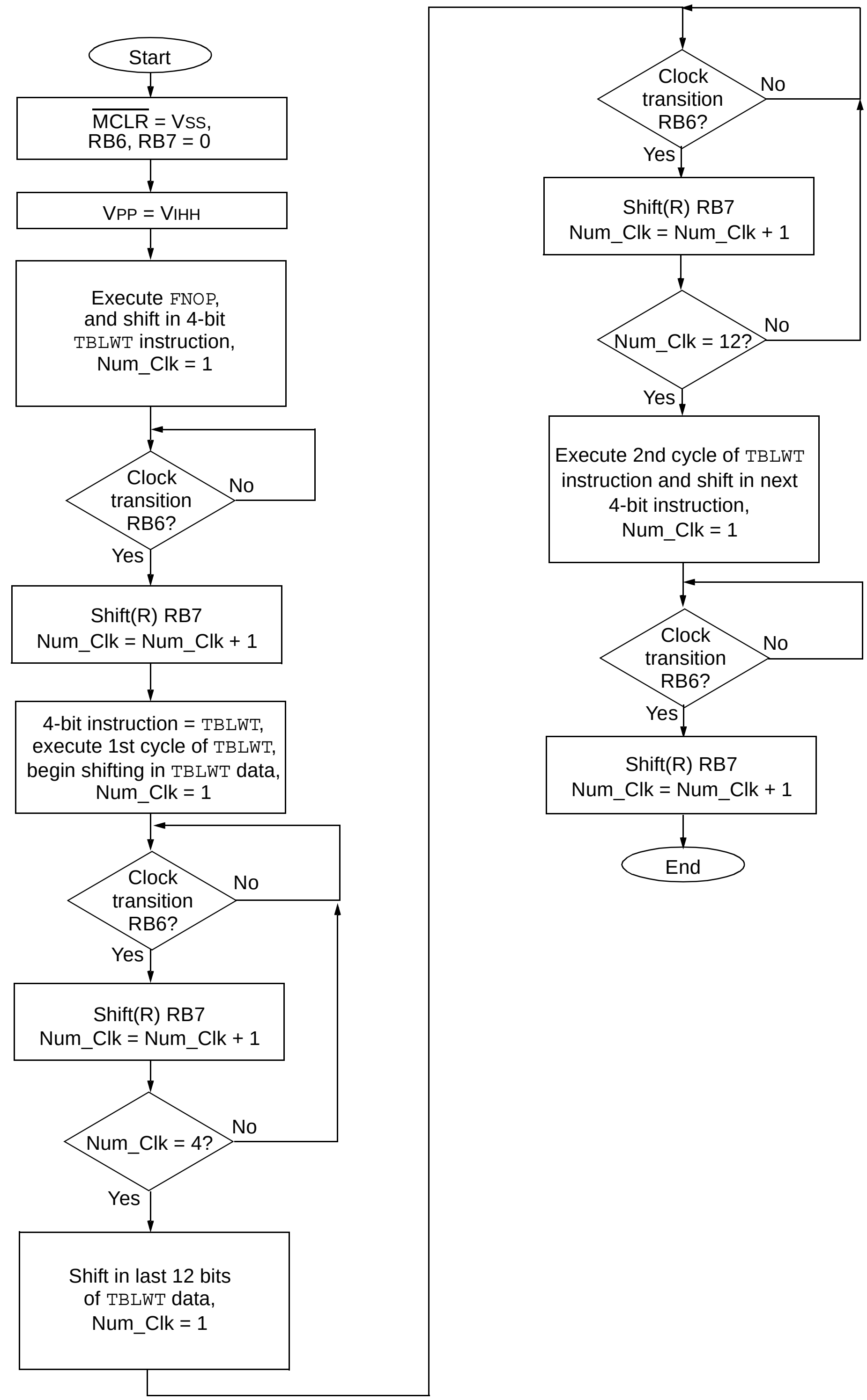
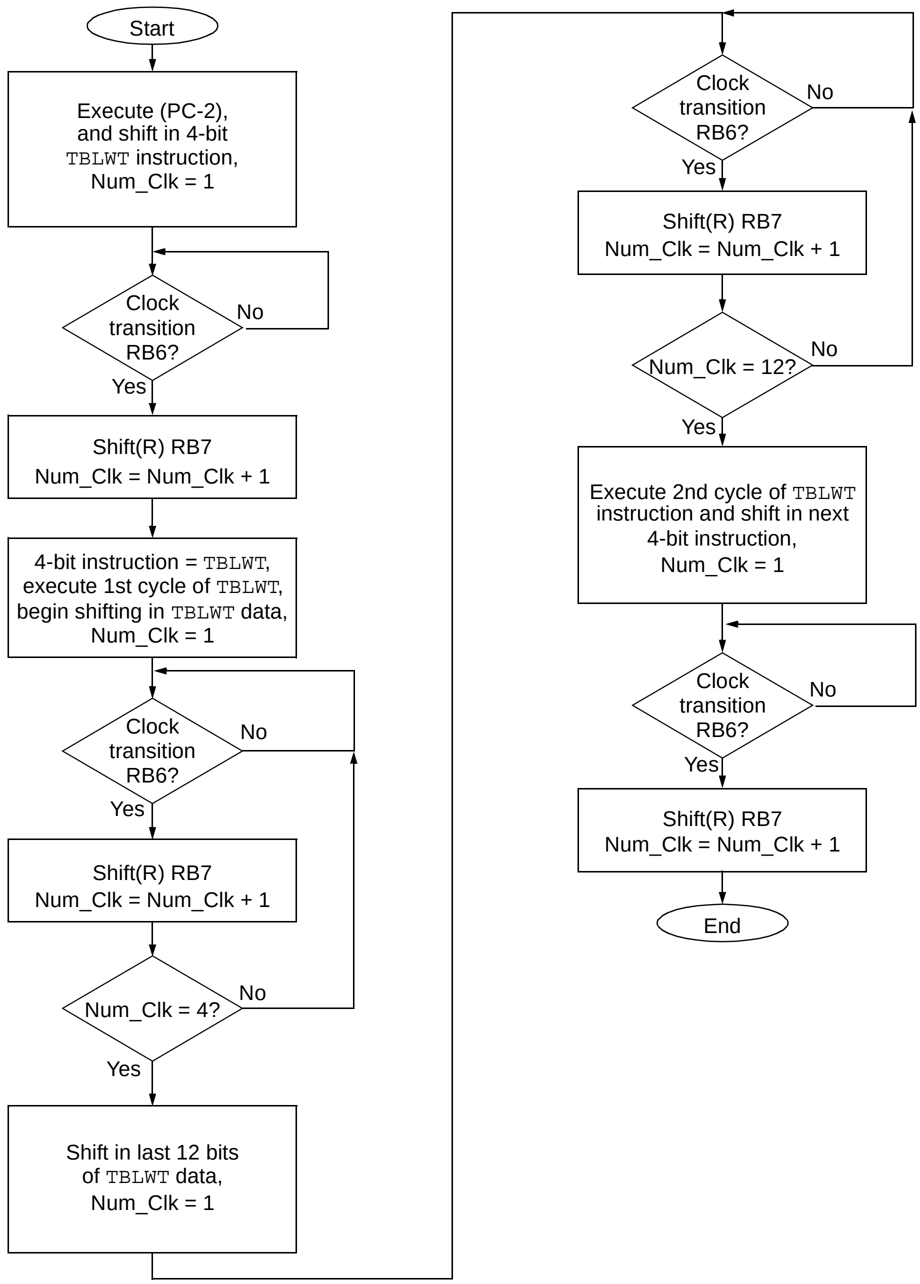


FIGURE 2-11: TBLWT SERIAL INSTRUCTION FLOW



2.6 TBLRD Instruction

The TBLRD instruction is another special two-cycle instruction.

All forms of TBLRD instructions (post/pre-increment, post-decrement, etc.) are encoded as 4-bit special instructions. This is useful as TBLRD instructions are used repeatedly in ICSP mode. A 4-bit instruction will minimize the total number of clock cycles required to perform programming algorithms.

The TBLRD instruction sequence operates as follows:

1. The 4-bit TBLRD instruction is read in by the state machine on RB7 during the four clock cycle execution of the instruction fetched previous to the TBLRD (which is an FNOP if the TBLRD is executed following a RESET).
2. Once the state machine recognizes that the instruction fetched is a TBLRD, the state machine releases the CPU and allows execution of the first and second cycles of the TBLRD instruction for eight clock cycles. When the TBLRD is performed, the contents of the program memory byte pointed to by the TBLPTR is loaded into the TABLAT register.
3. After eight clock cycles have transitioned on RB6, and the TBLRD instruction has completed, the state machine will suspend the CPU for eight clock cycles. During these eight clock cycles, the state machine configures RB7 as an output, and will shift out the contents of the TABLAT register onto RB7 LSb first.
4. When the state machine has shifted out all eight bits of data, the state machine suspends the CPU to allow an instruction pre-fetch. Four clock cycles are required on RB6 to shift in the next 4-bit instruction.

FIGURE 2-12: TBLRD INSTRUCTION SEQUENCE

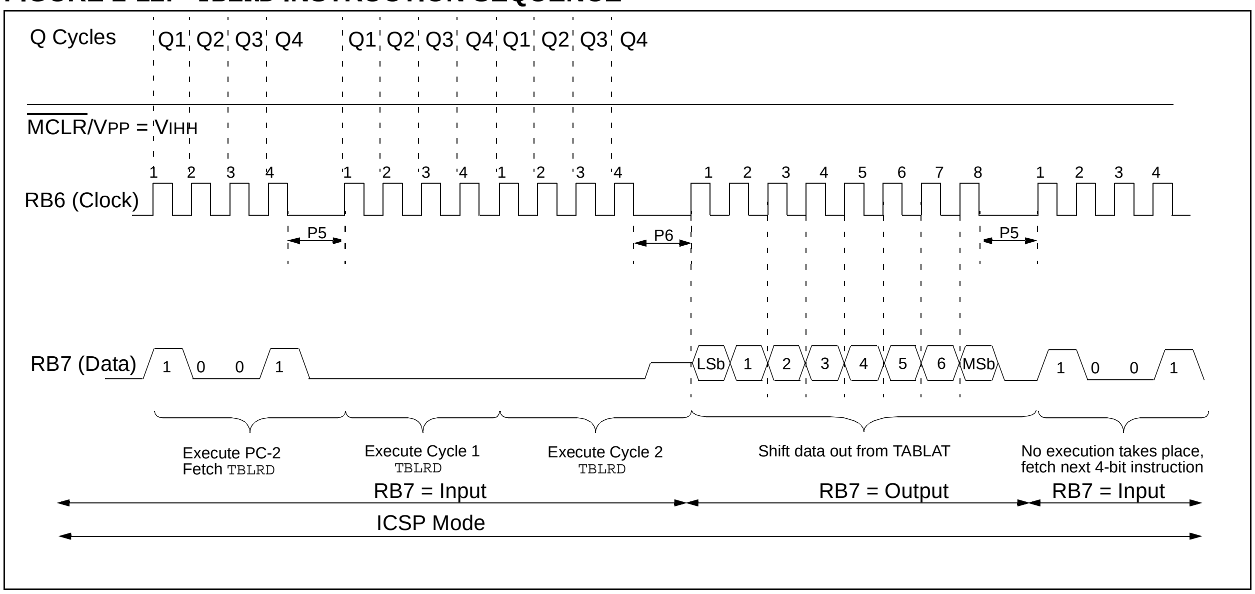


FIGURE 2-13: TBLRD SERIAL INSTRUCTION FLOW AFTER RESET

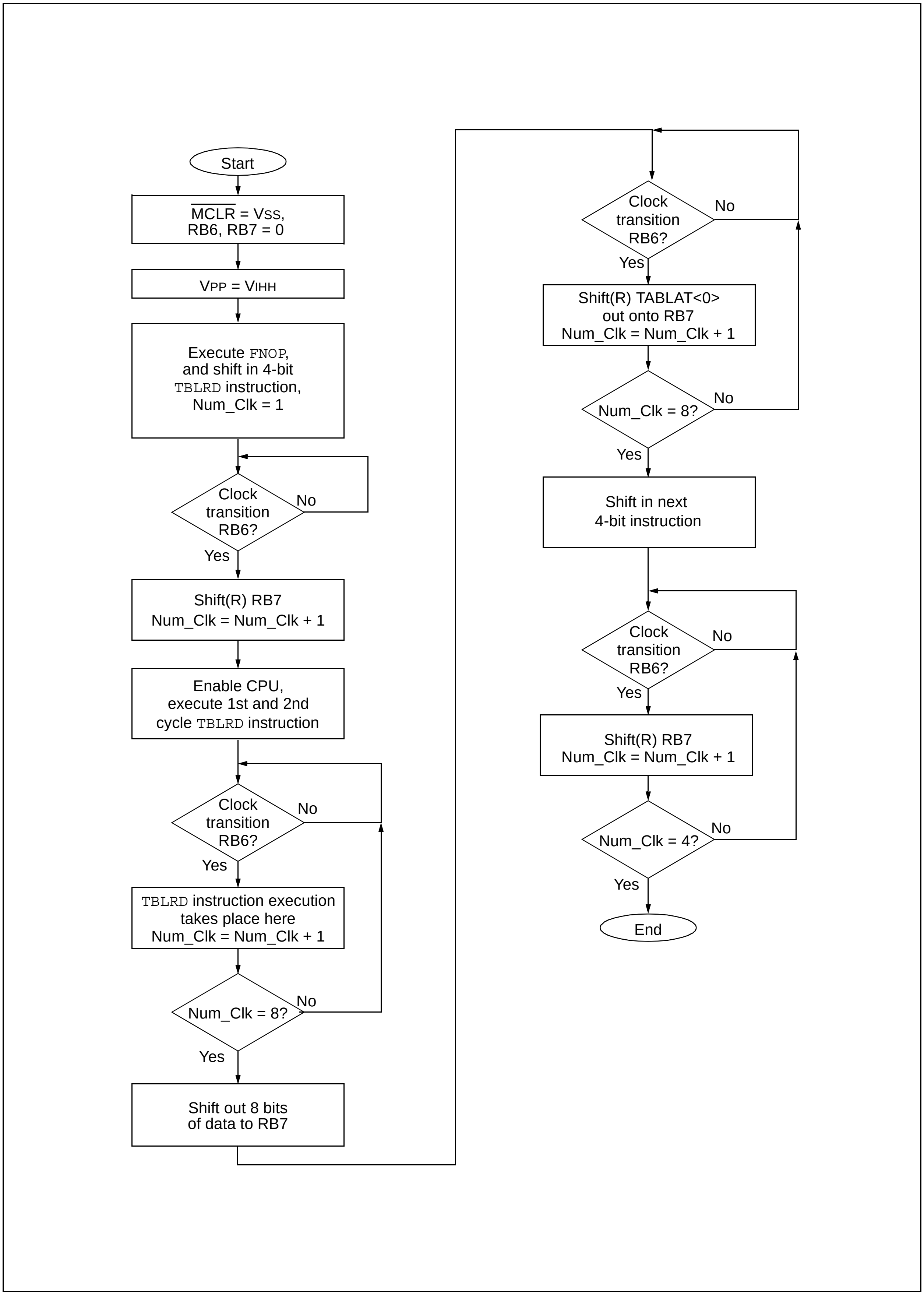
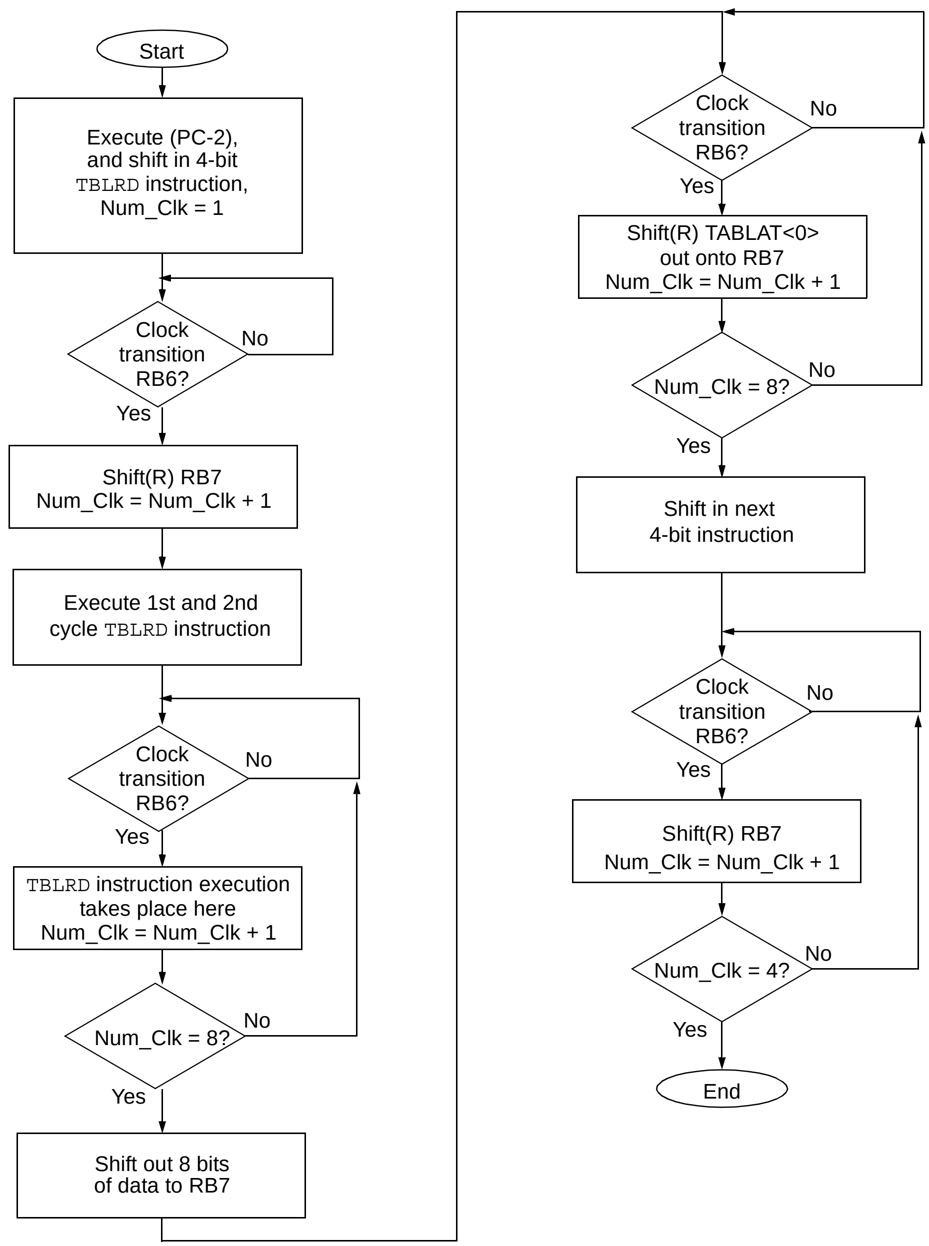


FIGURE 2-14: TBLRD SERIAL INSTRUCTION FLOW



2.6.1 SOFTWARE COMMANDS

ICSP commands of the PICmicro MCU are supported in the PIC18CXXX family by simply combining CPU instructions. Once in the ICSP mode, instructions are loaded into a shift register, and the device waits for a command to be received. The ICSP commands for the PIC18CXXX family are now “pseudo-commands” and are shown in Table 2-2. The following sections describe how to implement the pseudo-commands using CPU instructions.

TABLE 2-2: ICSP PSEUDO COMMAND MAPPING

ICSP™ Command	Golden Gate Instructions	
Load Configuration	MOVLW	#Address1
	MOVWF	TBLPTRL
	MOVLW	#Address2
	MOVWF	TBLPTRH
	MOVLW	#Address3
	MOVWF	TBLPTRU
Load Data	Not needed. Data encoded in 4-bit TBLWT instruction sequence.	
Read Data	TBLRD instruction	
Increment Address	Not needed. Use TBLWT with increment/decrement (TBLWT *+/*-).	
Load Address	MOVLW	#Addr_low
	MOVWF	TBLPTRL
	MOVLW	#Addr_high
	MOVWF	TBLPTRH
	MOVLW	#Addr_upper
	MOVWF	TBLPTRU
RESET Address	MOVLW	#Data
	MOVWF	TBLPTRH
	MOVWF	TBLPTRL
	MOVWF	TBLPTRU
Begin Programming	TBLWT	
End Programming	Not needed. Programming will cease at the end of TBLWT execution.	

2.6.2 RESET ADDRESS

A reset of the program memory pointer is a write to the upper, high, and low bytes of the TBLPTR. To reset the program memory pointer, the following instruction sequence is used.

```
NOP                         ;(4-BIT INSTRUCTION)
MOVLW   00h
NOP                         ;(4-BIT INSTRUCTION)
MOVWF   TBLPTRU, 0
NOP                         ;(4-BIT INSTRUCTION)
MOVWF   TBLPTRH, 0
NOP                         ;(4-BIT INSTRUCTION)
MOVWF   TBLPTRL, 0
```


FIGURE 2-15: RESET ADDRESS SERIAL INSTRUCTION SEQUENCE

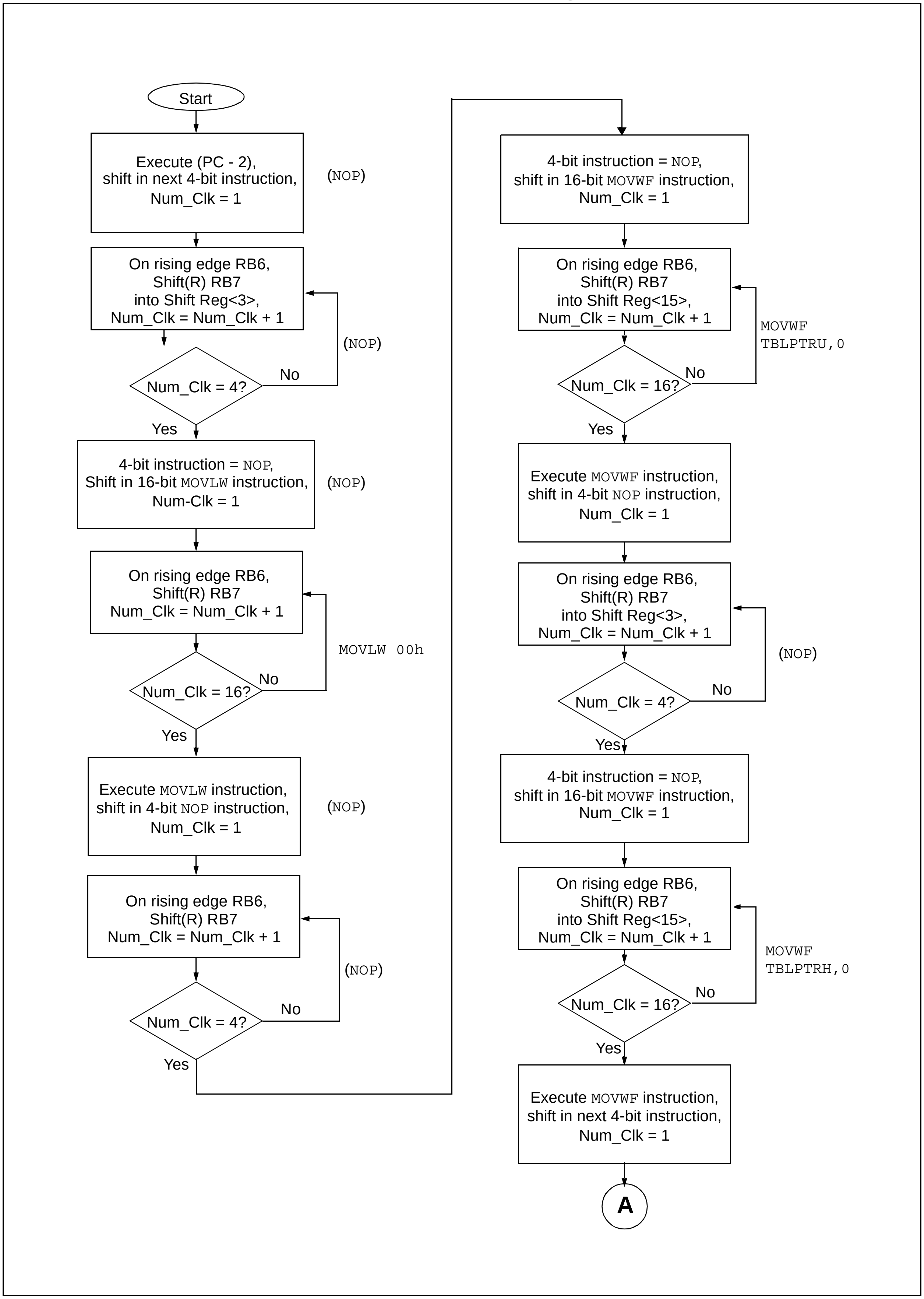
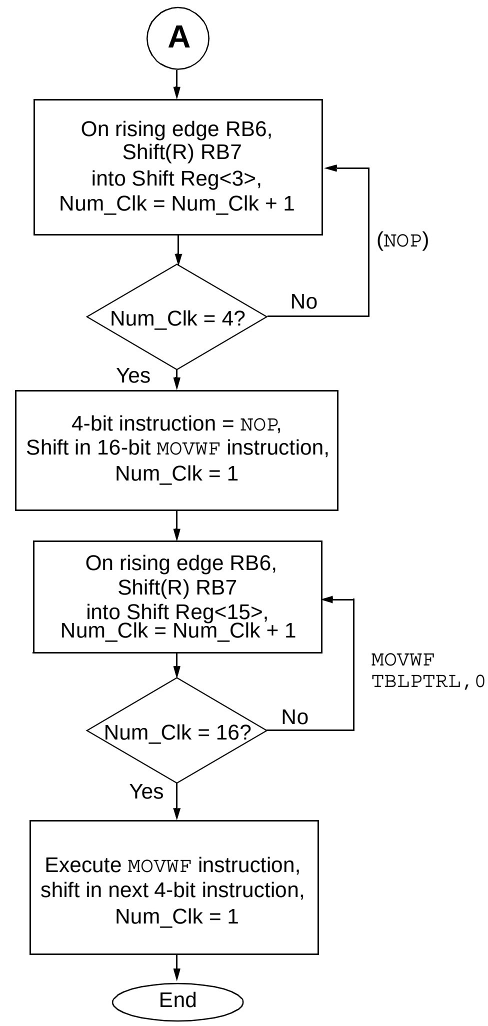


FIGURE 2-15: RESET ADDRESS SERIAL INSTRUCTION SEQUENCE (CONTINUED)



2.6.3 LOAD ADDRESS

This is used to load the address pointer to the Program Memory with a specific 22-bit value, and is useful when a specific range of locations are to be accessed. To load the address into the table pointer, the following commands must be used:

```
NOP                  ; 4-bit instruction
MOVLW   Low_Address
NOP                  ; 4-bit instruction
MOVWF   TBLPTRL, 0
NOP                  ; 4-bit instruction
MOVLW   High_Address
NOP                  ; 4-bit instruction
MOVWF   TBLPTRH, 0
NOP                  ; 4-bit instruction
MOVLW   Upper_Address
NOP                  ; 4-bit instruction
MOVWF   TBLPTRU, 0
```

FIGURE 2-16: LOAD ADDRESS SERIAL INSTRUCTION SEQUENCE

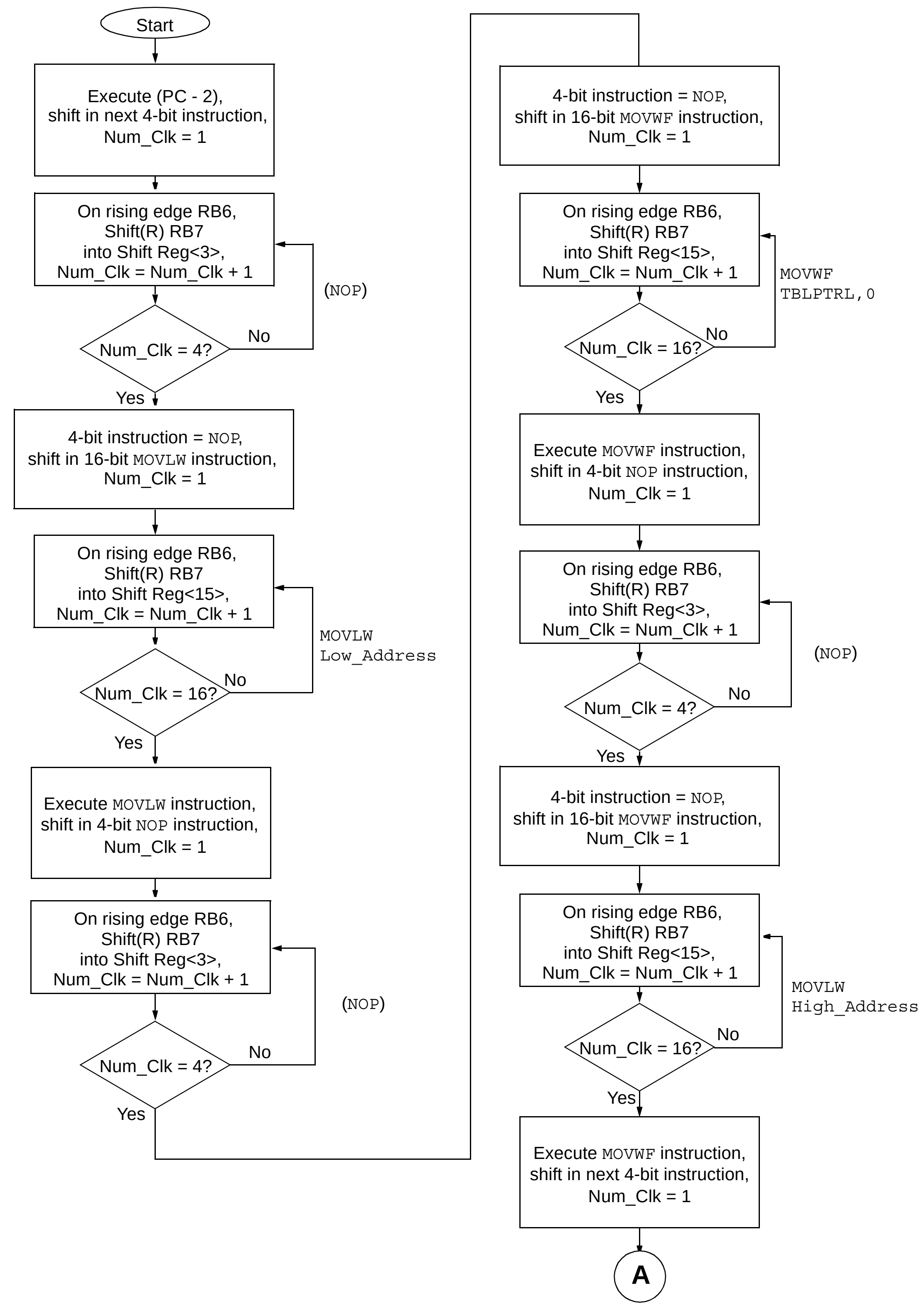
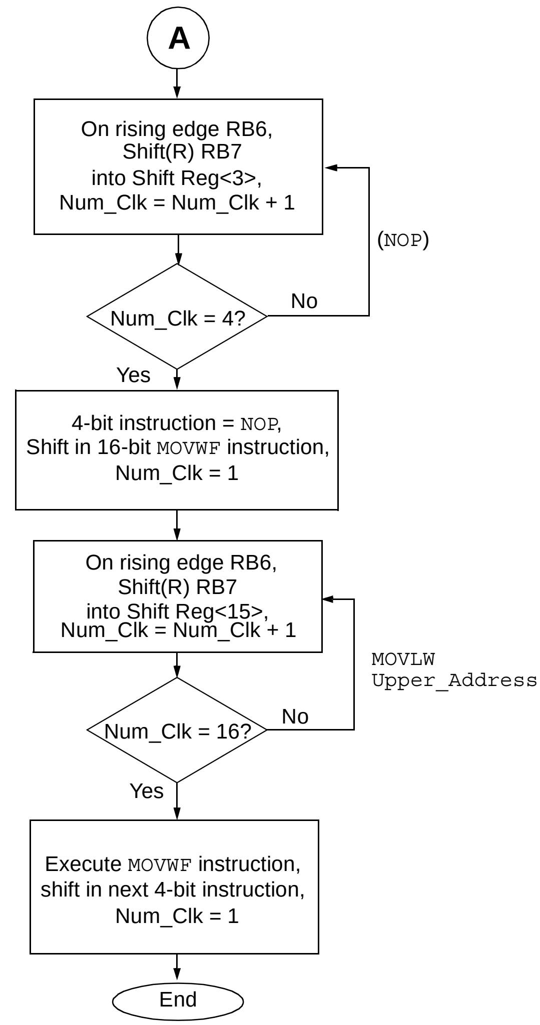


FIGURE 2-16: LOAD ADDRESS SERIAL INSTRUCTION SEQUENCE (CONTINUED)



2.6.4 ICSP BEGIN PROGRAMMING

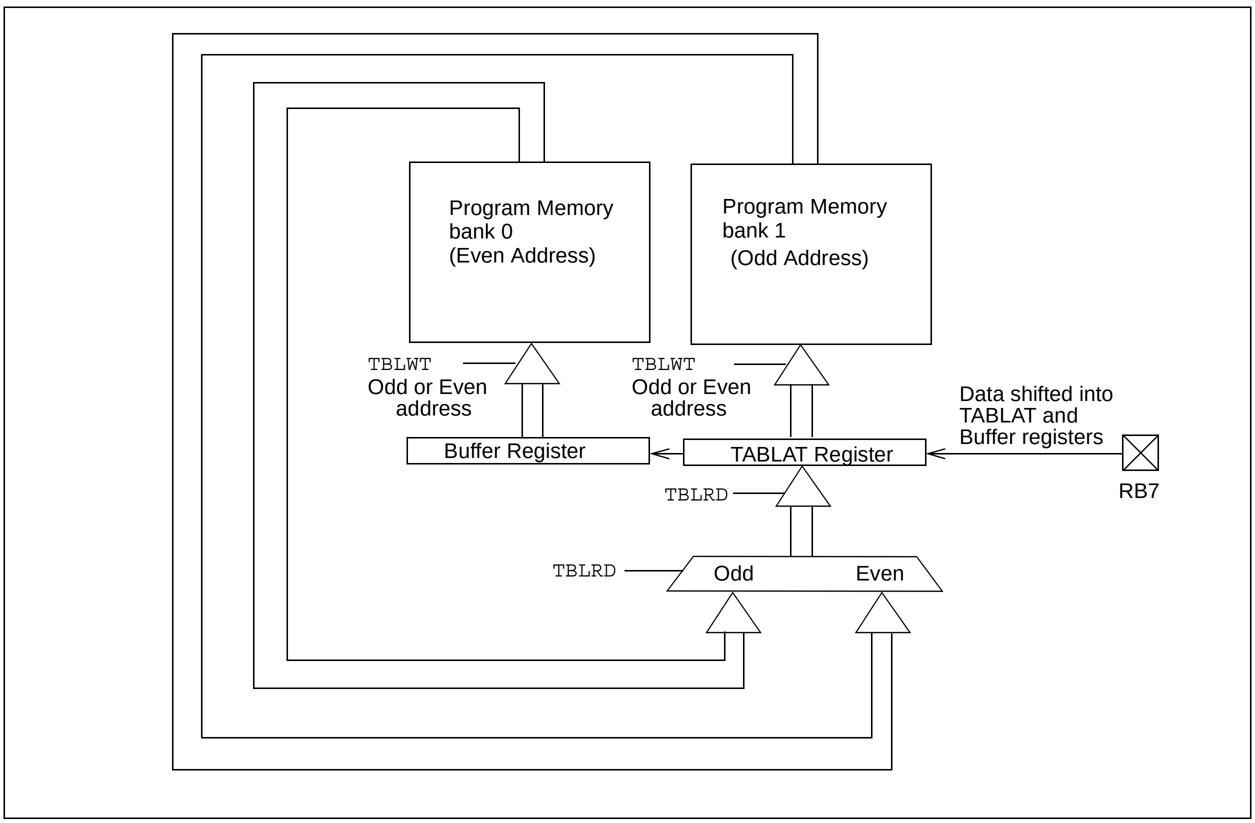
Programming is performed by executing a TBLWT instruction. In ICSP mode, the TBLWT instruction sequence will include 16 bits of data shifted into a data buffer, and then written to the word location addressed by the TBLPTR. Although the TBLPTR addresses the program memory on a byte wide boundary, all 16 bits of data shifted in during the TBLWT sequence are written at once. The 16 bits are shifted into the TABLAT and buffer registers. The TBLPTR points to the word that will be programmed; it can point to either the high or the low byte (see Figure 2-17).

The sequence for programming a location could occur as follows:

1. Set up the TLBPTR with the first address to be programmed (even or odd byte).
2. Shift in a 4-bit TBLWT instruction.
3. 16 bits of data are shifted in for programming both high and low byte of the first programmed location.
4. Execute TBLWT instruction to program location.
5. Verify high byte (odd address) by executing TLBRD*- (post-decrement). (TBLPTR points at odd address.)
6. Verify low byte (even address) by executing TLBRD*+ (post-increment). (TBLPTR points at odd address again.)
7. If location doesn't verify, go back to step 4.
8. If location does verify, begin 3x over-programming (see Section 2.6.7).

The TBLWT instruction offers flexibility with multiple addressing modes: pre-increment, post-increment, post-decrement, and no change of the TBLPTR. These modes eliminate the need for the increment address command sequence.

FIGURE 2-17: DATA BUFFERING SCHEME FOR ICSP



2.6.5 PROGRAMMING INSTRUCTION SEQUENCE

The instructions needed to execute a programming sequence are shown in the following example. Many of the instruction sequences are also shown in previous sections.

```
NOP                         ; 4-bit instruction
                            ; Set up low byte
                            ; of program address
MOVLW  Low_Byte_Address     ; = 00
NOP                         ; 4-bit instruction
MOVWF  TBLPTRL, 0
NOP                         ; 4-bit instruction
                            ; Set up high byte
                            ; of program
                            ; address
MOVLW  High_Byte_Address    ; = 00
NOP                         ; 4-bit instruction
MOVWF  TBLPTRH, 0
NOP                         ; 4-bit instruction
                            ; Set up upper byte
                            ; of program
                            ; address
MOVLW  Upper_Byte_Address; = 00
NOP                         ; 4-bit instruction
MOVWF  TBLPTRU, 0           ; Program data byte
                            ; included in TBLWT
                            ; instruction
                            ; sequence

TBLWT+*                     ; TBLPTR = 000000h
```

A write of a program memory location with an odd or an even address causes a long write cycle in ICSP mode. The 16-bit data is encoded in the TBLWT sequence and is loaded into the temporary buffer register for word wide writes.

2.6.6 VERIFY SEQUENCE

The table pointer = 000001h in the last example. A TBLRD will then read the odd address byte of the current program word address location first. The verify sequence will be as follows:

```
; Read/verify high byte first
      TBLRD*-
; TBLPTR = 0000 post-dec
; Read/verify low byte
      TBLRD*
```

The first TBLRD decrements the table pointer to point to the even address byte of the current program word. After the first and second cycle of the TBLRD are performed, all eight bits of data are shifted out on RB7. The fetch of the second TBLRD occurs on the next four clock cycles. The second TBLRD does not modify the table pointer address. This allows another programming cycle (TBLWT+*) to take place if the verify doesn't match the program data, without having to update the table pointer.

If the contents of the verify do not match the intended program data word, then the TBLWT instruction must be repeated with the correct contents of the current program word. Therefore, only one instruction needs to be performed to repeat the programming cycle:

```
      TBLWT+*
```

2.6.7 3X OVER-PROGRAMMING

Once a location has been both programmed and verified over the range of voltages, 3x over-programming should be applied. In other words, apply three times the number of programming pulses required to program a location in memory to ensure solid programming margin.

This means that every location will be programmed a minimum of four times (1 + 3x over-programming).

FIGURE 2-18: DETAILED PROGRAMMING FLOW CHART – PROGRAM MEMORY

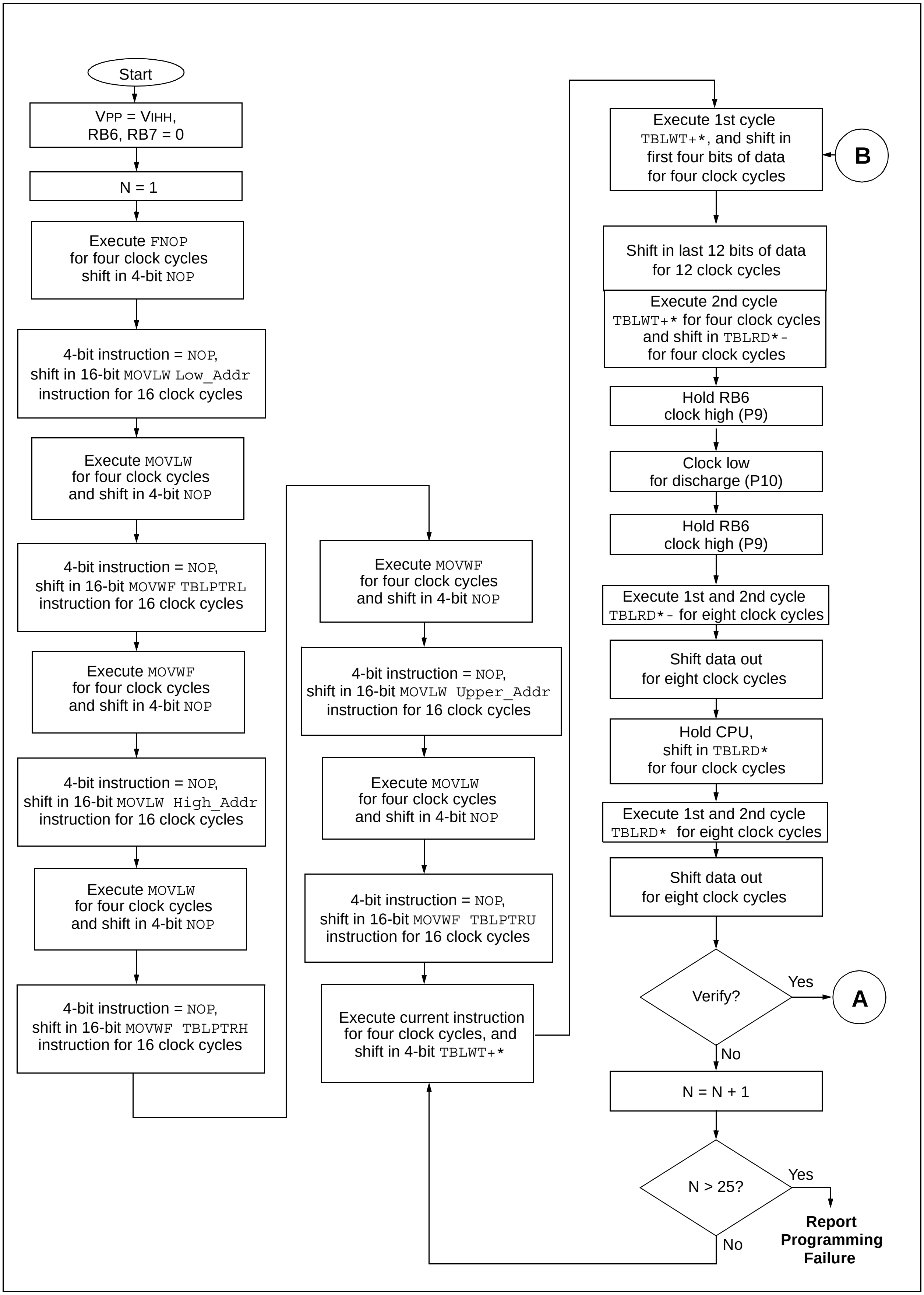
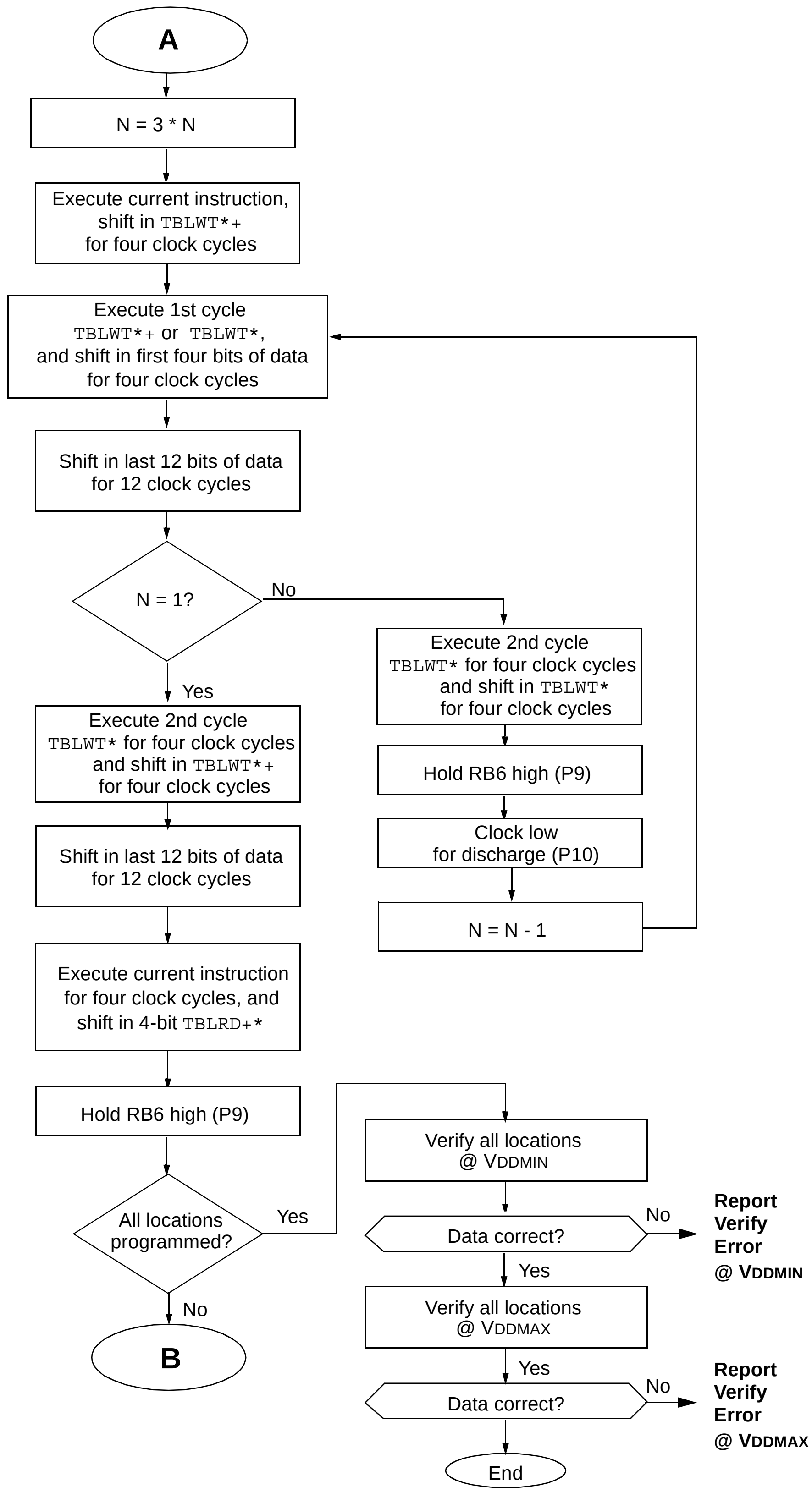


FIGURE 2-18: DETAILED PROGRAMMING FLOW CHART – PROGRAM MEMORY (CONTINUED)



2.6.8 LOAD CONFIGURATION

The Configuration registers are located in test memory, and are only addressable when the high address bit of the TBLPTR (bit 21) is set. Test program memory contains test memory, configuration registers, calibration registers, and ID locations. The desired address must be loaded into all three bytes of the table pointer to program specific ID locations or the configuration bits. To program the configuration registers, the following sequence must be followed:

```
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVLW instruction
MOVLW   03h
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVWF instruction
                      ; Enable Test memory
MOVWF   TBLPTRU, 0
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVLW instruction
MOVLW   Low_Config_Address
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVWF instruction
MOVWF   TBLPTRL, 0
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVLW instruction
MOVLW   High_Config_Address
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVWF instruction
MOVWF   TBLPTRH, 0
NOP                   ; 4-bit instruction
                      ; shift in 16-bit
                      ; MOVLW instruction
TBLWT   *+
                      ; 16-bits of data are
                      ; shifted in for write
                      ; of config1L and
                      ; config1H TBLWT is a
                      ; 4-bit special
                      ; instruction Wait
                      ; 100 µsec for programming
```

2.6.9 END PROGRAMMING

When programming occurs, 16 bits of data are programmed into memory. The 16 bits of data are shifted in during the `TBLWT` sequence. After the programming command (`TBLWT`) has been executed, the user must wait 100 µs until programming is complete, before another command can be executed by the CPU. There is no command to end programming.

RB6 must remain high for as long as programming is desired. When RB6 is lowered, programming will cease.

After the falling edge occurs on RB6, RB6 must be held low for a period of time (Parameter 10) so a high voltage discharge can be performed. This ensures the program array isn’t stressed at high voltage during execution of the next instruction. The high voltage discharge will occur while RB6 is low following the programming time.

FIGURE 2-19: DETAILED PROGRAMMING FLOW CHART – CONFIG WORD

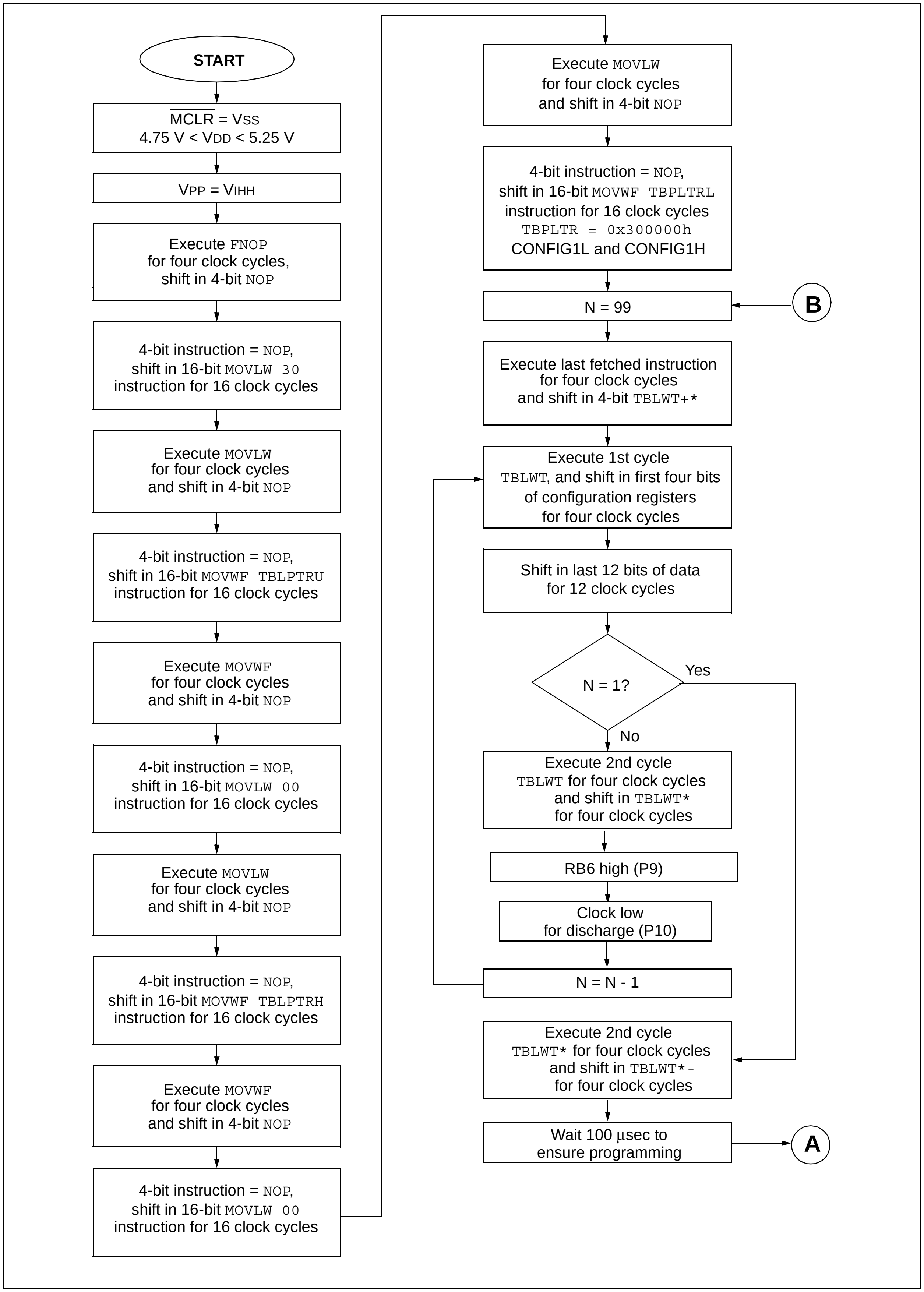


FIGURE 2-20: DETAILED PROGRAMMING FLOW CHART – CONFIG WORD

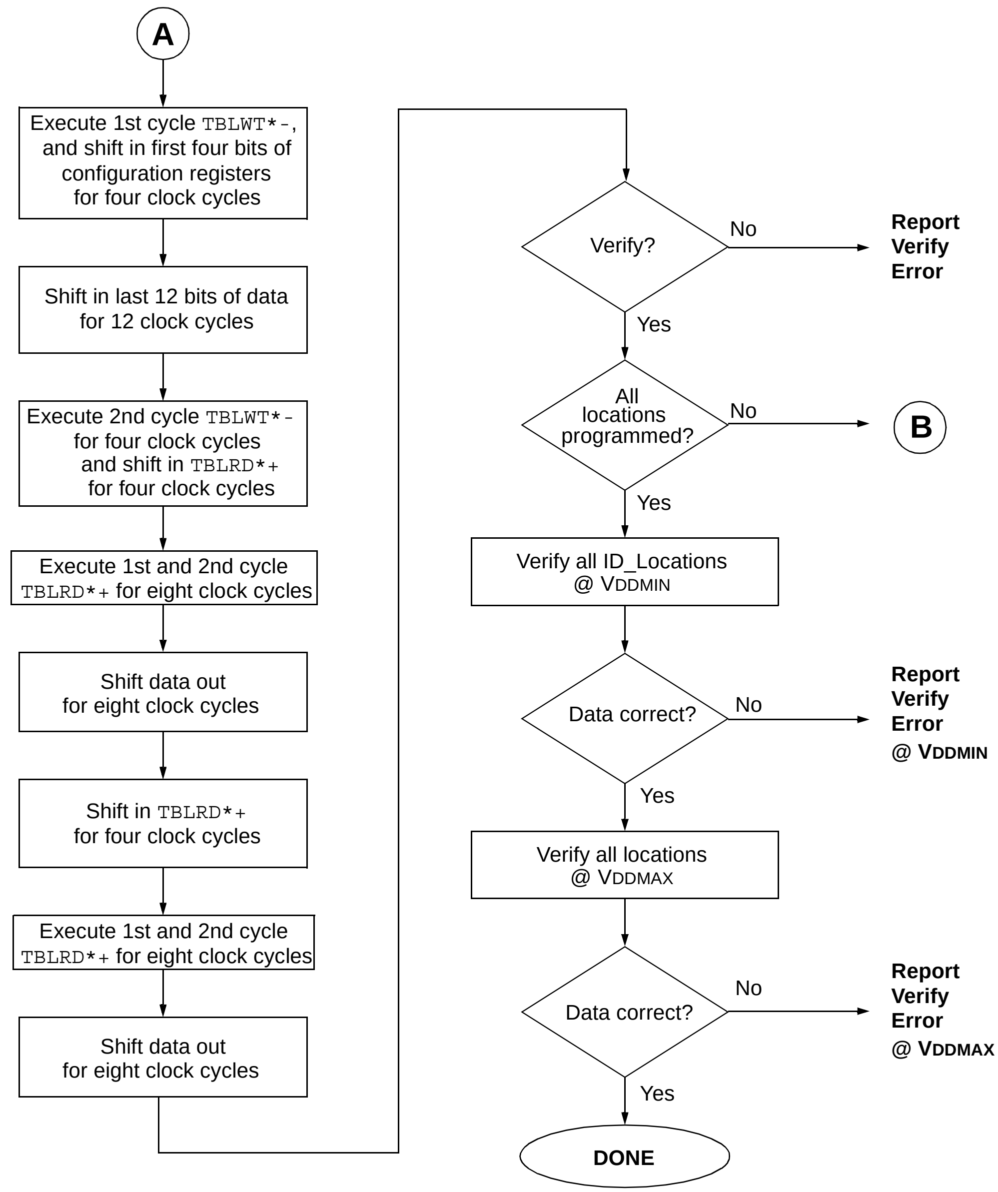


FIGURE 2-21: DETAILED PROGRAMMING FLOW CHART – ID LOCATION

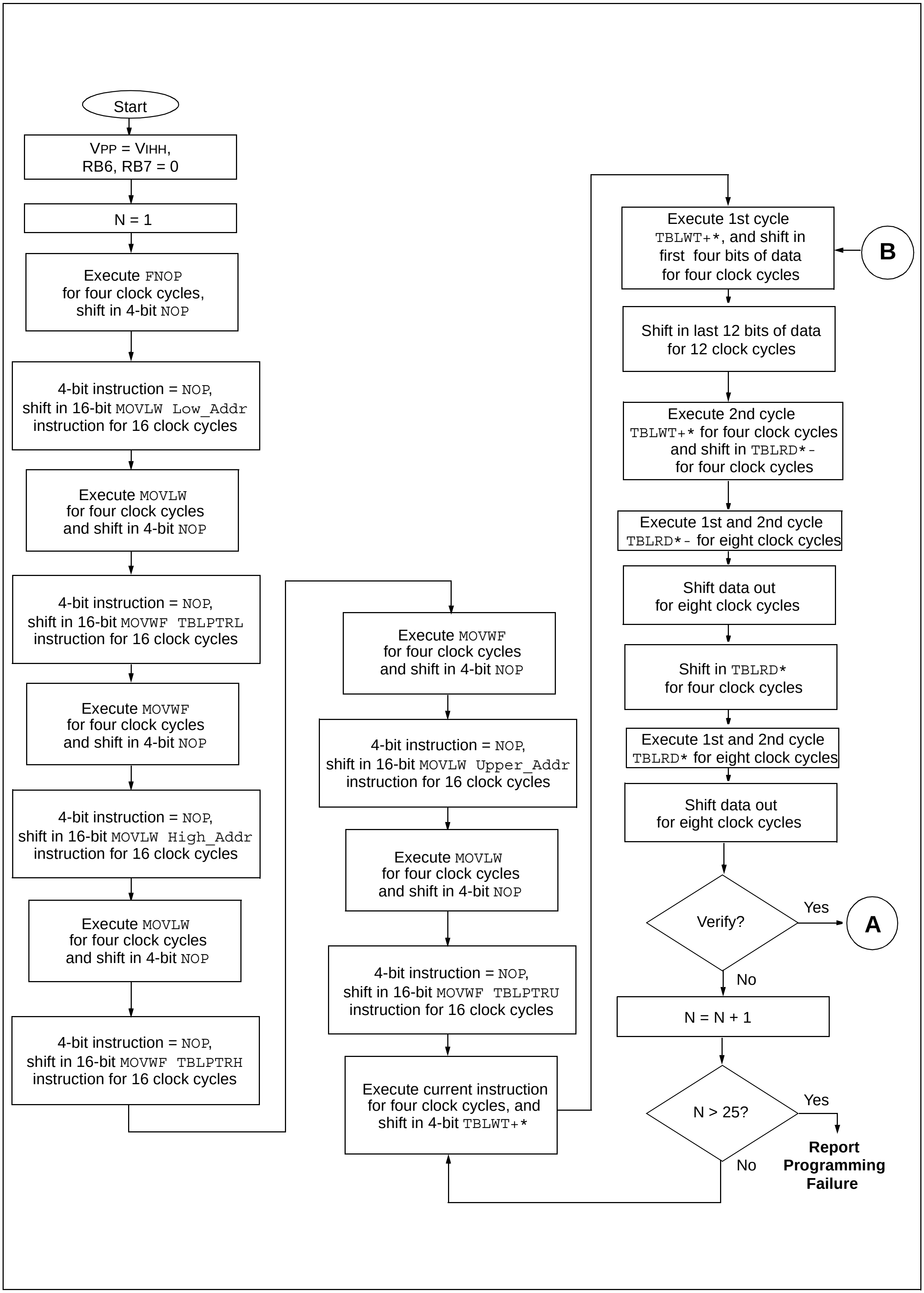
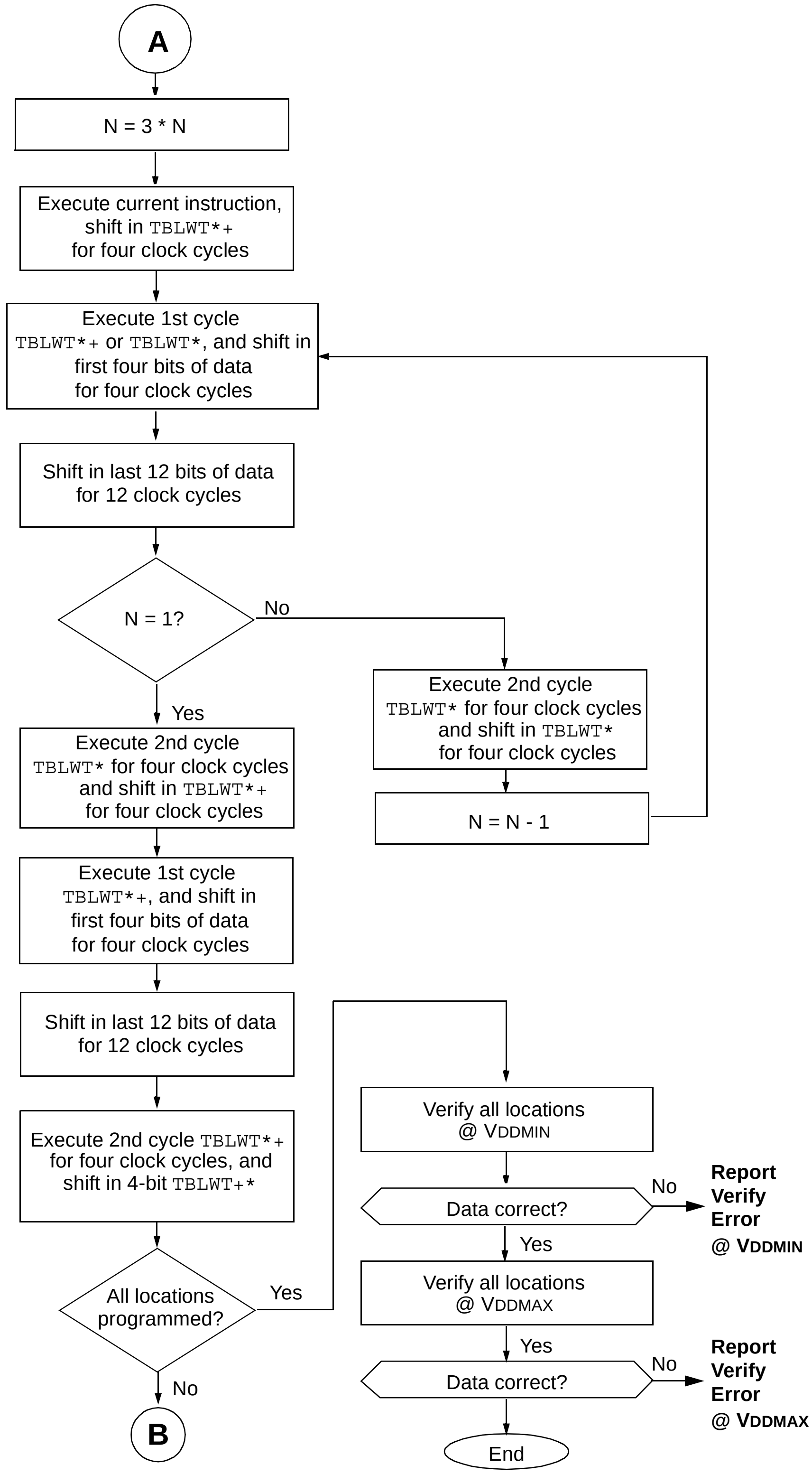


FIGURE 2-21: DETAILED PROGRAMMING FLOW CHART – ID LOCATION (CONTINUED)



3.0 CONFIGURATION WORD

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h – 3FFFFFh).

3.1 ID Locations

A user may store identification information (ID) in eight ID locations mapped in [0x200000:0x200007]. It is recommended that the user use only the four least significant bits of each ID location.

The ID locations do not read out in a scrambled fashion after code protection is enabled. For all devices, it is recommended to write ID locations as '1111 bbbb' where 'bbbb' is the ID information.

TABLE 3-1: 18CXX2 CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300000h	CONFIG1L	CP	CP	CP	CP	CP	CP	CP	CP	1111 1111
300001h	CONFIG1H	r	r	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0	111- -111
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$	---- 1111
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	---- 1111
300005h	CONFIG3H	—	—	—	—	—	—	—	CCP2MX	---- ---1
300006h	CONFIG4L	—	—	—	—	—	—	r	STVREN	---- --11
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	0000 0000
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0010

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved.
Grayed cells are unimplemented, read as 0.

TABLE 3-2: 18CXX8 CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300000h	CONFIG1L	CP	CP	CP	CP	CP	CP	CP	CP	1111 1111
300001h	CONFIG1H	r	r	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0	111- -111
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$	---- 1111
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	---- 1111
300006h	CONFIG4L	—	—	—	—	—	—	r	STVREN	---- --11
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	0000 0000
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0010

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved.
Grayed cells are unimplemented, read as 0.

TABLE 3-3: 18CXX8 BIT DESCRIPTION

Bit Name	Bit Type	File Name/Devices	Description
CP	R/P – 1	CONFIG1L/18CXXX	Code Protection bits 1 = Program memory code protection off 0 = All of program memory code protected
$\overline{\text{OSCSEN}}$	R/P – 1	CONFIG1H/18CXXX	Oscillator System Clock Switch Enable bit 1 = Oscillator system clock switch option is disabled (Main oscillator is source) 0 = Oscillator system clock switch option is enabled (Oscillator switching is enabled)
FOSC2:FOSC0	R/P – 1	CONFIG1H/18CXXX	Oscillator Selection bits 111 = RC oscillator w/ OSC2 configured as RA6 110 = HS oscillator with PLL enabled/Clock frequency = (4 X Fosc) 101 = EC oscillator w/ OSC2 configured as RA6 100 = EC oscillator w/ OSC2 configured as divide by 4 clock output 011 = RC oscillator 010 = HS oscillator 001 = XT oscillator 000 = LP oscillator
BORV1:BORV0	R/P – 1	CONFIG2L/18CXXX	Brown-out Reset Voltage bits 11 = V_{BOR} set to 2.5 V 10 = V_{BOR} set to 2.7 V 01 = V_{BOR} set to 4.2 V 00 = V_{BOR} set to 4.5 V
BOREN	R/P – 1	CONFIG2L/18CXXX	Brown-out Reset Enable bit 1 = Brown-out Reset enabled 0 = Brown-out Reset disabled
$\overline{\text{PWRTEN}}$	R/P – 1	CONFIG2L/18CXXX	Power-up Timer Enable bit 1 = PWRT disabled 0 = PWRT enabled Enabling Brown-out Reset automatically enables the Power-up Timer (PWRT), regardless of the value of bit $\overline{\text{PWRTEN}}$. Ensure Power-up Timer is enabled when Brown-out Reset is enabled.
WDTPS2:WDTPS0	R/P – 1	CONFIG2H/18CXXX	Watchdog Timer Postscale Select bits 111 = 1 : 128 110 = 1 : 64 101 = 1 : 32 100 = 1: 16 011 = 1: 8 010 = 1 : 4 001 = 1 : 2 000 = 1 : 1
WDTEN	R/P – 1	CONFIG2H/18CXXX	Watchdog Timer Enable bit 1 = WDT enabled 0 = WDT disabled (control is placed on SWDTEN bit)
CCP2MX	R/P – 1	CONFIG3H/18CXX2	CCP2 Mux bit 1 = CCP2 input/output is multiplexed with RC1 0 = CCP2 input/output is multiplexed with RB3
STVREN	R/P – 1	CONFIG4L/18CXXX	Stack Overflow/Underflow Reset Enable bit 1 = Stack Overflow/Underflow will cause RESET 0 = Stack Overflow/Underflow will not cause RESET
DEV10:DEV3	R	DEVID2/18CXXX	Device ID bits These bits are used with the DEV2:DEV0 bits in the DEVID1 register to identify part number.
DEV2:DEV0	R	DEVID1/18CXXX	Device ID bits These bits are used with the DEV10:DEV3 bits in the DEVID2 register to identify part number.
REV4:REV0	R	DEVID1/18CXXX	These bits are used to indicate the revision of the device.

Legend: R = readable, P = programmable, U = unimplemented, read as '0', - n = value when device is unprogrammed, u = unchanged.

3.2 Embedding Configuration Word Information in the Hex File

To allow portability of code, a PIC18CXXX 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.

3.3 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.

Table 3-4 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 3-4: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	0xAA at 0 and max address
PIC18C242	Disable	SUM[0x0000:0x3FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01	0xC146	0xC09C
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01 + SUM_ID	0x005E	0x0068
PIC18C252	Disable	SUM[0x0000:0x7FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01	0x8146	0x809C
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01 + SUM_ID	0x005A	0x0064
PIC18C442	Disable	SUM[0x0000:0x3FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01+ CONFIG4L & 0x01	0xC146	0xC09C
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01 + SUM_ID	0x005E	0x0068
PIC18C452	Disable	SUM[0x0000:0x7FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01	0x8146	0x809C
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0xF + CONFIG2H & 0x0F + CONFIG3H & 0x01 + CONFIG4L & 0x01 + SUM_ID	0x005A	0x0064
PIC18C658	Disabled	SUM[0x0000: 0x7FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG4L & 0x01	0x8145	0x809B
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG4L & 0x01 + SUM_ID	0x0058	0x0062
PIC18C858	Disabled	SUM[0x0000: 0x7FFF] + CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG4L & 0x01	0x8145	0x809B
	Enabled	CONFIG1L & 0xFF + CONFIG1H & 0x27 + CONFIG2L & 0x0F + CONFIG2H & 0x0F + CONFIG4L & 0x01 + SUM_ID	0x0058	0x0062

Legend:

Item	Description
CFGW =	Configuration Word
SUM[a:b] =	Sum of locations a to b inclusive
SUM_ID =	Byte-wise sum of lower four bits of all customer ID locations
+ =	Addition
& =	Bitwise AND

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

Standard Operating Conditions
Operating Temperature: -40°C ≤ TA ≤ +40°C, unless otherwise stated, (25°C is recommended).
Operating Voltage: 4.75 V ≤ VDD ≤ 5.25 V, unless otherwise stated.

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	VIHH	Programming Voltage on VPP/MCLR pin	12.75	—	13.25	V	
	IPP	Programming current on MCLR pin		25	50	mA	
P1	TSER	Serial setup time	20	—	—	ns	
P2	TSCLK	Serial Clock period	100	—	—	ns	
P3	TSET1	Input Data Setup Time to serial clock ↓	15	—	—	ns	
P4	THLD1	Input Data Hold Time from serial clock ↓	15	—	—	ns	
P5	TDLY1	Delay between last clock ↓ to first clock ↑ of next command	20	—	—	ns	
P6	TDLY2	Delay between last clock ↓ of command byte to first clock ↑ of read of data word	20	—	—	ns	
P8	TDLY4	Data input not driven to next clock input	1	—	—	ns	
P9	TDLY5	RB6 high time (minimum programming time)	100	—	—	µs	
P10	TDLY6	RB6 low time after programming (high voltage discharge time)	100	—	—	ns	

† Data in “Typ” column is at 5 V, 25°C unless otherwise stated.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

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

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3838 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

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

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

China - Beijing
Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai
Microchip Technology Shanghai Office
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

Hong Kong
Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

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

ASIA/PACIFIC (continued)

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
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 Munich, 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: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

10/01/00

DNV Certification, Inc. USA
DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.