Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

PIC16F527

20-Pin, 8-Bit Flash Microcontroller

Processor Features:

- Interrupt Capability
- PIC16F527 Operating Speed:
  - DC – 20 MHz Crystal oscillator
  - DC – 200 ns Instruction cycle
- High-Endurance Program and Flash Data Memory Cells:
  - 1024 x 12 user execution memory
  - 64 x 8 self-writable data memory
  - 100,000 write program memory endurance
  - 1,000,000 write Flash data memory endurance
  - Program and Flash data retention: >40 years
- General Purpose Registers (SRAM):
  - 68 x 8 for PIC16F527
- Only 36 Single-Word Instructions to Learn:
  - Added `RETURN` and `RETFIE` instructions
  - Added `MOVLB` instruction
- All Instructions are Single-Cycle except for Program Branches which are Two-Cycle
- Four-Level Deep Hardware Stack
- Direct, Indirect and Relative Addressing modes for Data and Instructions

Peripheral Features:

- Device Features:
  - One Input-only pin
  - 17 I/Os
  - Individual direction control
  - High-current source/sink
- 8-Bit Real-Time Clock/Counter (TMR0) with 8-Bit Programmable Prescaler
- In-Circuit Serial Programming™ (ICSP™) via Two External Pin Connections
- Analog Comparator (CMP):
  - Two analog comparators
  - Absolute and programmable references
- Analog-to-Digital Converter (ADC):
  - 8-bit resolution
  - Eight external input channels
  - One internal channel to convert comparator
  - 0.6V reference input
- Operational Amplifiers (op amps):
  - Two operational amplifiers
  - Fully-accessible visibility

eXtreme Low-Power (XLP) Features

- Sleep mode 50 nA @ 2.0V, typical
- Watchdog Timer (WDT): 500 nA @ 2.0V, typical

Microcontroller Features:

- Brown-out Reset (BOR)
- Power-on Reset (POR)
- Device Reset Timer (DRT)
- Watchdog Timer (WDT) with a Dedicated RC Oscillator
- Programmable Code Protection (CP)
- Power-Saving Sleep mode with Wake-up on Change Feature
- Selectable Oscillator Options:
  - INTOSC: Precision 4 or 8 MHz internal oscillator
  - EXTRC: Low-cost external RC oscillator
  - LP: Power-saving, low-frequency crystal
  - XT: Standard crystal/resonator
  - HS: High-speed crystal/resonator
  - EC: High-speed external clock
- Variety of Packaging Options:
  - 20-Lead PDIP, SOIC, SSOP, QFN, UQFN

CMOS Technology:

- Low-Power, High-Speed CMOS Flash Technology
- Fully-Static Design
- Wide Operating Voltage and Temperature Range:
  - Industrial: 2.0V to 5.5V
  - Extended: 2.0V to 5.5V
- Operating Current:
  - 170 uA @ 2V, 4 MHz, typical
  - 15 uA @ 2V, 32 kHz, typical
- Standby Current:
  - 100 nA @ 2V, typical

TABLE 1: PIC16F527 AND PIC16F570 FAMILY TYPES

Device	Data Sheet Index	I/O Pins(1)	Flash	Data EE (B)	SRAM (B)	8-Bit ADC Channels	Op Amp	Comparator	8-Bit Timers	BOR	Stack Levels	Interrupts	8 MHz Int. Osc.	Interrupt-on-Change Pins	Weak Pull-up Pins	XLP
PIC16F527	(1)	18	1 KW	64	68	8	2	2	1	Y	4	Y	Y	4	4	Y
PIC16F570	(2)	25	2 KW	64	132	8	2	2	1	Y	4	Y	Y	8	8	Y

Note 1: One pin is input-only.

Data Sheet Index: (Unshaded devices are described in this document.)

1: DS40001652 PIC16F527 Data Sheet, 20-Pin, 8-bit Flash Microcontroller.

2: DS40001684 PIC16F570 Data Sheet, 28-Pin, 8-bit Flash Microcontroller.

FIGURE 1: 20-PIN DIAGRAM FOR PIC16F527

PDIP, SSOP, SOIC

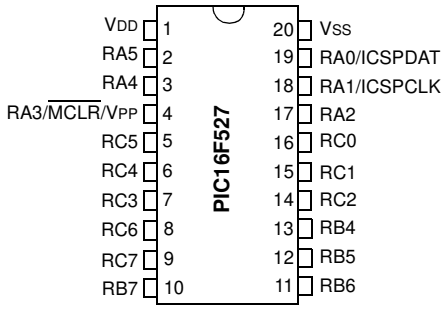


FIGURE 2: 20-PIN DIAGRAM FOR PIC16F527

QFN, UQFN

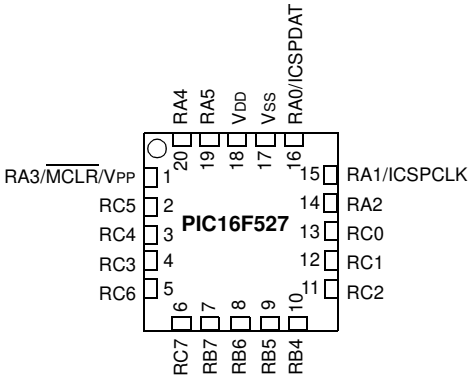


TABLE 2: 20-PIN ALLOCATION TABLE

I/O	20-Pin PDIP/SOIC/SSOP	20-Pin QFN/UQFN	Analog	Oscillator	Comparator	Reference	Timers	Op Amp	Clock Reference	ICSP™	Basic	Pull-up	Interrupt-on-Change
RA0	19	16	AN0	—	C1IN+	—	—	—	—	ICSPDAT	—	Y	Y
RA1	18	15	AN1	—	C1IN-	CVREF	—	—	—	ICSPCLK	—	Y	Y
RA2	17	14	AN2	—	C1OUT	—	T0CKI	—	—	—	—	—	—
RA3	4	1	—	—	—	—	—	—	—	—	$\overline{\text{MCLR}}$ VPP	Y	Y
RA4	3	20	AN3	OSC2	—	—	—	—	CLKOUT	—	—	Y	Y
RA5	2	19	—	OSC1	—	—	—	—	CLKIN	—	—	—	—
RB4	13	10	—	—	—	—	—	OP2-	—	—	—	—	—
RB5	12	9	—	—	—	—	—	OP2+	—	—	—	—	—
RB6	11	8	—	—	—	—	—	—	—	—	—	—	—
RB7	10	7	—	—	—	—	—	—	—	—	—	—	—
RC0	16	13	AN4	—	C2IN+	—	—	—	—	—	—	—	—
RC1	15	12	AN5	—	C2IN-	—	—	—	—	—	—	—	—
RC2	14	11	AN6	—	—	—	—	OP2	—	—	—	—	—
RC3	7	4	AN7	—	—	—	—	OP1	—	—	—	—	—
RC4	6	3	—	—	C2OUT	—	—	—	—	—	—	—	—
RC5	5	2	—	—	—	—	—	—	—	—	—	—	—
RC6	8	5	—	—	—	—	—	OP1-	—	—	—	—	—
RC7	9	6	—	—	—	—	—	OP1+	—	—	—	—	—
VDD	1	18	—	—	—	—	—	—	—	—	—	—	—
VSS	20	17	—	—	—	—	—	—	—	—	—	—	—

Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our website at **www.microchip.com** to receive the most current information on all of our products.

1.0 GENERAL DESCRIPTION

The PIC16F527 device from Microchip Technology is a low-cost, high-performance, 8-bit, fully-static, Flash-based CMOS microcontroller. It employs a RISC architecture with only 36 single-word/single-cycle instructions. All instructions are single cycle except for program branches, which take two cycles. The PIC16F527 device delivers performance an order of magnitude higher than its competitors in the same price category. The 12-bit wide instructions are highly symmetrical, resulting in a typical 2:1 code compression over other 8-bit microcontrollers in its class. The easy-to-use and easy to remember instruction set reduces development time significantly.

The PIC16F527 product is equipped with special features that reduce system cost and power requirements. The Power-on Reset (POR) and Device Reset Timer (DRT) eliminate the need for external Reset circuitry. There are several oscillator configurations to choose from, including INTRC Internal Oscillator mode and the power-saving LP (Low-Power) Oscillator mode. Power-Saving Sleep mode, Watchdog Timer and code protection features improve system cost, power and reliability.

The PIC16F527 device is available in the cost-effective Flash programmable version, which is suitable for production in any volume. The customer can take full advantage of Microchip's price leadership in Flash programmable microcontrollers, while benefiting from the Flash programmable flexibility.

The PIC16F527 product is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a 'C' compiler, a low-cost development programmer and a full-featured programmer. All the tools are supported on IBM® PC and compatible machines.

1.1 Applications

The PIC16F527 device fits in applications ranging from personal care appliances and security systems to low-power remote transmitters/receivers. The Flash technology makes customizing application programs (transmitter codes, appliance settings, receiver frequencies, etc.) extremely fast and convenient. The small footprint packages, for through hole or surface mounting, make these microcontrollers perfect for applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16F527 device very versatile, even in areas where no microcontroller use has been considered before (e.g., timer functions, logic and PLDs in larger systems and coprocessor applications).

TABLE 1-1: FEATURES AND MEMORY OF PIC16F527

		PIC16F527
Clock	Maximum Frequency of Operation (MHz)	20
Memory	Flash Program Memory	1024
	SRAM Data Memory (bytes)	68
	Flash Data Memory (bytes)	64
Peripherals	Timer Module(s)	TMR0
	Wake-up from Sleep on Pin Change	Yes
Features	I/O Pins	17
	Input Pins	1
	Internal Pull-ups	Yes
	In-Circuit Serial Programming™	Yes
	Number of Instructions	36
	Packages	20-pin PDIP, SOIC, SSOP, QFN, UQFN
	Interrupts	Yes

2.0 PIC16F527 DEVICE VARIETIES

A variety of packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in this section. When placing orders, please use the PIC16F527 Product Identification System at the back of this data sheet to specify the correct part number.

2.1 Quick Turn Programming (QTP) Devices

Microchip offers a QTP programming service for factory production orders. This service is made available for users who choose not to program medium-to-high quantity units and whose code patterns have stabilized. The devices are identical to the Flash devices but with all Flash locations and fuse options already programmed by the factory. Certain code and prototype verification procedures do apply before production shipments are available. Please contact your local Microchip Technology sales office for more details.

2.2 Serialized Quick Turn Programming^{SM} (SQTP^{SM}) Devices

Microchip offers a unique programming service, where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number, which can serve as an entry code, password or ID number.

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16F527 device can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16F527 device uses a Harvard architecture in which program and data are accessed on separate buses. This improves bandwidth over traditional von Neumann architectures where program and data are fetched on the same bus. Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. Instruction opcodes are 12 bits wide, making it possible to have all single-word instructions. A 12-bit wide program memory access bus fetches a 12-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (200 ns @ 20 MHz, 1 μs @ 4 MHz) except for program branches.

Table 3-1 below lists memory supported by the PIC16F527 device.

TABLE 3-1: PIC16F527 MEMORY

Device	Program Memory	Data Memory	
	Flash (words)	SRAM (bytes)	Flash (bytes)
PIC16F527	1024	68	64

The PIC16F527 device can directly or indirectly address its register files and data memory. All Special Function Registers (SFR), including the PC, are mapped in the data memory. The PIC16F527 device has a highly orthogonal (symmetrical) instruction set that makes it possible to carry out any operation, on any register, using any Addressing mode. This symmetrical nature and lack of “special optimal situations” make programming with the PIC16F527 device simple, yet efficient. In addition, the learning curve is reduced significantly.

The PIC16F527 device contains an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is eight bits wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two’s complement in nature. In two-operand instructions, one operand is typically the W (working) register. The other operand is either a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC) and Zero (Z) bits in the STATUS register. The C and DC bits operate as a borrow and digit borrow out bit, respectively, in subtraction. See the `SUBWF` and `ADDWF` instructions for examples.

A simplified block diagram is shown in Figure 3-2, with the corresponding device pins described in Table 3-2.

FIGURE 3-1: PIC16F527 BLOCK DIAGRAM

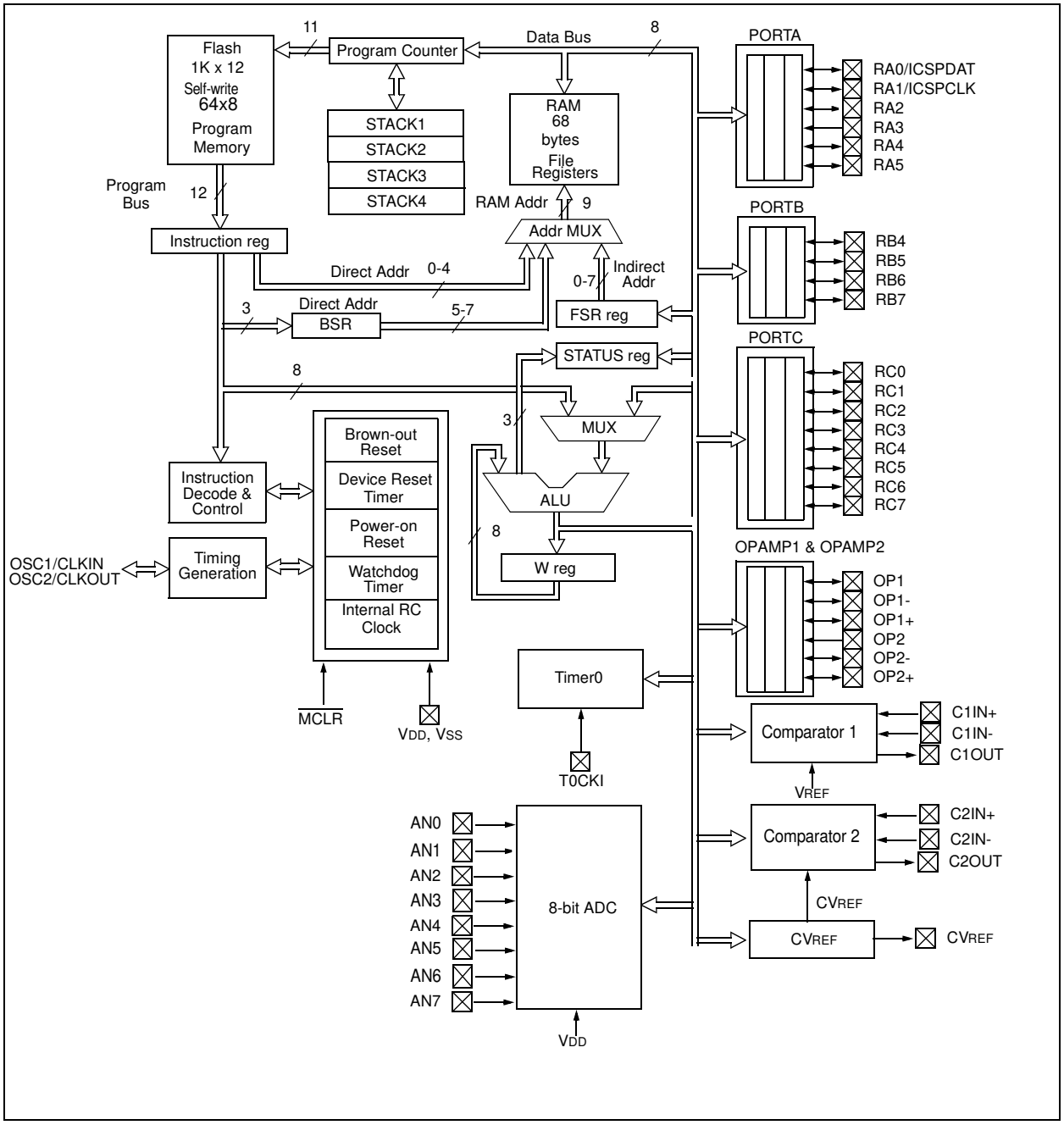


TABLE 3-2: PIC16F527 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/C1IN+/ICSPDAT	RA0	TTL	CMOS	Bidirectional I/O pin. It can be software programmed for internal weak pull-up and wake-up from Sleep on pin change.
	ICSPDAT	ST	CMOS	ICSP™ mode Schmitt Trigger.
	C1IN+	AN	—	Comparator 1 input.
	AN0	AN	—	ADC channel input.
RA1/AN1/C1IN-/CVREF/ ICSPCLK	RA1	TTL	CMOS	Bidirectional I/O pin. It can be software programmed for internal weak pull-up and wake-up from Sleep on pin change.
	ICSPCLK	ST	—	ICSP™ mode Schmitt Trigger.
	C1IN-	AN	—	Comparator 1 input.
	CVREF	—	AN	Programmable Voltage Reference output.
	AN1	AN	—	ADC channel input.
RA2/AN2/C1OUT/T0CKI	RA2	TTL	CMOS	Bidirectional I/O port.
	C1OUT	—	CMOS	Comparator 1 output.
	AN2	AN	—	ADC channel input.
	T0CKI	ST	—	Timer0 Schmitt Trigger input pin.
RA3/$\overline{\text{MCLR}}$/VPP	RA3	TTL	—	Standard TTL input with weak pull-up.
	$\overline{\text{MCLR}}$	ST	—	Master Clear (Reset). When configured as $\overline{\text{MCLR}}$, this pin is an active-low Reset to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD during normal device operation or the device will enter Programming mode. Weak pull-up is always on if configured as $\overline{\text{MCLR}}$.
	VPP	HV	—	Test mode high-voltage pin.
RA4/AN3/OSC2/CLKOUT	RA4	TTL	CMOS	Bidirectional I/O pin. It can be software programmed for internal weak pull-up and wake-up from Sleep on pin change.
	OSC2	—	XTAL	Oscillator crystal output. Connections to crystal or resonator in Crystal Oscillator mode (XT, HS and LP modes only, PORTB in other modes).
	CLKOUT	—	CMOS	EXTRC/INTRC CLKOUT pin (FOSC/4).
	AN3	AN	—	ADC channel input.
RA5/OSC1/CLKIN	RA5	TTL	CMOS	Bidirectional I/O port.
	OSC1	XTAL	—	XTAL oscillator input pin.
	CLKIN	ST	—	EXTRC Schmitt Trigger input.
RB4/OP2-	RB4	TTL	CMOS	Bidirectional I/O port.
	OP2-	AN	—	Op amp 2 inverting input.
RB5/OP2+	RB5	TTL	CMOS	Bidirectional I/O port.
	OP2+	AN	—	Op amp 2 non-inverting input.
RB6	RB6	TTL	CMOS	Bidirectional I/O port.
RB7	RB7	TTL	CMOS	Bidirectional I/O port.
RC0/AN4/C2IN+	RC0	ST	CMOS	Bidirectional I/O port.
	AN4	AN	—	ADC channel input.
	C2IN+	AN	—	Comparator 2 input.
RC1/AN5/C2IN-	RC1	ST	CMOS	Bidirectional I/O port.
	AN5	AN	—	ADC channel input.
	C2IN-	AN	—	Comparator 2 input.

Legend: I = Input, O = Output, I/O = Input/Output, P = Power, — = Not Used, TTL = TTL input, ST = Schmitt Trigger input, HV = High Voltage, AN = Analog Voltage

TABLE 3-2: PIC16F527 PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RC2/AN6/OP2	RC2	ST	CMOS	Bidirectional I/O port.
	AN6	AN	—	ADC channel input.
	OP2	—	AN	Op amp 2 output.
RC3/AN7/OP1	RC3	ST	CMOS	Bidirectional I/O port.
	AN7	AN	—	ADC channel input.
	OP1	—	AN	Op amp 1 output.
RC4/C2OUT	RC4	ST	CMOS	Bidirectional I/O port.
	C2OUT	—	CMOS	Comparator 2 output.
RC5	RC5	ST	CMOS	Bidirectional I/O port.
RC6/OP1-	RC6	ST	CMOS	Bidirectional I/O port.
	OP1-	AN	—	Op amp 1 inverting input.
RC7/OP1+	RC7	ST	CMOS	Bidirectional I/O port.
	OP1+	AN	—	Op amp 1 non-inverting input.
V_{DD}	V_{DD}	—	P	Positive supply for logic and I/O pins.
V_{SS}	V_{SS}	—	P	Ground reference for logic and I/O pins.

Legend: I = Input, O = Output, I/O = Input/Output, P = Power, — = Not Used, TTL = TTL input, ST = Schmitt Trigger input, HV = High Voltage, AN = Analog Voltage

3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN pin) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the PC is incremented every Q1 and the instruction is fetched from program memory and latched into the instruction register in Q4. It is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2 and Example 3-1.

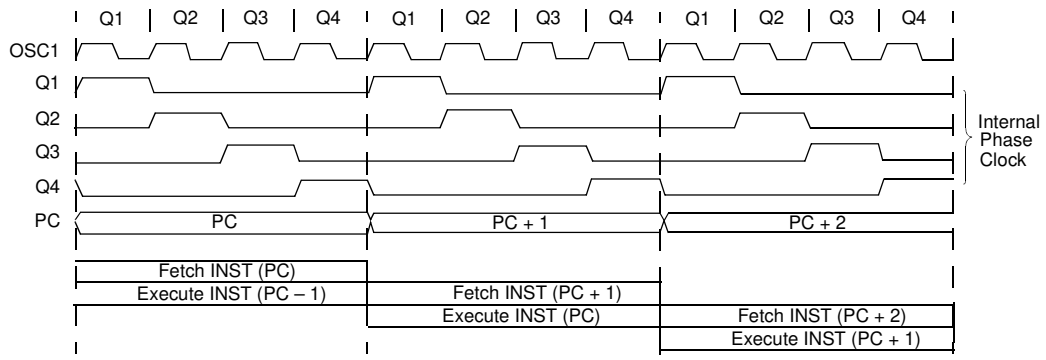
3.2 Instruction Flow/Pipelining

An instruction cycle consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the PC to change (e.g., `GOTO` or an interrupt), then two cycles are required to complete the instruction (see Example 3-1).

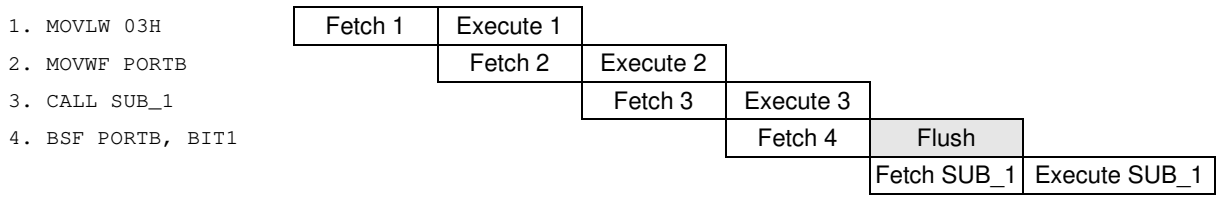
A fetch cycle begins with the PC incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW



All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is “flushed” from the pipeline, while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

The PIC16F527 memories are organized into program memory and data memory (SRAM).The self-writable portion of the program memory called self-writable Flash data memory is located at addresses 400h-43Fh. All program mode commands that work on the normal Flash memory, work on the Flash data memory. This includes bulk erase, row/column/cycling toggles, Load and Read data commands (Refer to **Section 5.0 "Self-Writable Flash Data Memory Control"** for more details). For devices with more than 512 bytes of program memory, a paging scheme is used. Program memory pages are accessed using one STATUS register bit. For the PIC16F527, with data memory register files of more than 32 registers, a banking scheme is used. Data memory banks are accessed using the File Select Register (FSR).

4.1 Program Memory Organization for PIC16F527

The PIC16F527 device has an 11-bit Program Counter (PC) capable of addressing a 2K x 12 program memory space. Program memory is partitioned into user memory, data memory and configuration memory spaces.

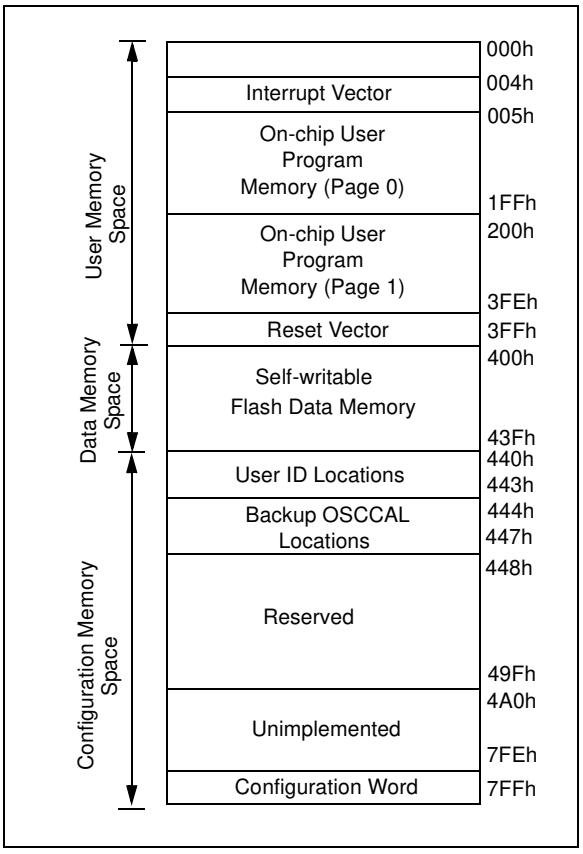
The user memory space is the on-chip user program memory. As shown in Figure 4-1, it extends from 0x000 to 0x3FF and partitions into pages, including an Interrupt vector at address 0x004 and a Reset vector at address 0x3FF.

The data memory space is the self-writable Flash data memory block and is located at addresses PC = 400h-43Fh. All program mode commands that work on the normal Flash memory, work on the Flash data memory block. This includes bulk erase, Load and Read data commands.

The configuration memory space extends from 0x440 to 0x7FF. Locations from 0x448 through 0x49F are reserved. The user ID locations extend from 0x440 through 0x443. The Backup OSCCAL locations extend from 0x444 through 0x447. The Configuration Word is physically located at 0x7FF.

Refer to "*PIC16F527 Memory Programming Specification*" (DS41640) for more details.

FIGURE 4-1: MEMORY MAP



4.2 Data Memory (SRAM and SFRs)

Data memory is composed of registers or bytes of SRAM. Therefore, data memory for a device is specified by its register file. The register file is divided into two functional groups: Special Function Registers (SFR) and General Purpose Registers (GPR).

The Special Function Registers are registers used by the CPU and peripheral functions for controlling desired operations of the PIC16F527. See **Section 4.3 "STATUS Register"** for details.

4.2.1 GENERAL PURPOSE REGISTER FILE

The General Purpose Register file is accessed directly or indirectly. See **Section 4.8 "Direct and Indirect Addressing"**.

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral functions to control the operation of the device (see **Section 4.3 "STATUS Register"**).

The Special Function Registers can be classified into two sets. The Special Function Registers associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section for each peripheral feature.

FIGURE 4-2: PIC16F527 REGISTER FILE MAP

BSR<1:0> →	00	01	10	11
File Address ↓		20h	40h	60h
00h	INDF(1)	INDF(1)	INDF(1)	INDF(1)
01h	TMR0	EECON	TMR0	IW
02h	PCL	PCL	PCL	PCL
03h	STATUS	STATUS	STATUS	STATUS
04h	FSR	FSR	FSR	FSR
05h	OSCCAL	EEDATA	OSCCAL	INTCON1
06h	PORTA	EEADR	PORTA	ISTATUS
07h	PORTB	CM1CON0	PORTB	IFSR
08h	PORTC	CM2CON0	PORTC	IBSR
09h	ADCON0	VRCON	ADCON0	OPACON
0Ah	ADRES	ANSEL	ADRES	ANSEL
0Bh	INTCON0	INTCON0	INTCON0	INTCON0
0Ch	General Purpose Registers	2Ch Addresses map back to addresses in Bank 0.	4Ch	6Ch
0Fh		2Fh	4Fh	6Fh
10h	General Purpose Registers	30h General Purpose Registers	50h General Purpose Registers	70h General Purpose Registers
1Fh		3Fh	5Fh	7Fh
	Bank 0	Bank 1	Bank 2	Bank 3

Note 1: Not a physical register. See **Section 4.8 "Direct and Indirect Addressing"**.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR/BOR	Value on all other Resets
Bank 0											
N/A	W(2)	Working Register (W)								xxxx xxxx	xxxx xxxx
N/A	TRIS	I/O Control Registers (TRISA, TRISB, TRISC)								1111 1111	1111 1111
N/A	OPTION	Contains control bits to configure Timer0 and Timer0/WDT prescaler								1111 1111	1111 1111
N/A	BSR(2)	—	—	—	—	—	—	BSR1	BSR0	---- -000	---- -0uu
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
01h	TMR0	Timer0 module Register								xxxx xxxx	uuuu uuuu
02h	PCL(1)	Low-order eight bits of PC								1111 1111	1111 1111
03h	STATUS(2)	Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	-001 1xxx	-00q qqqq
04h	FSR(2)	—	Indirect data memory address pointer							0xxx xxxx	0uuu uuuu
05h	OSCCAL	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—	1111 111-	uuuu uuu-
06h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--uu uuuu
07h	PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	uuuu ----
08h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
09h	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	1111 1100	1111 1100
0Ah	ADRES	ADC Conversion Result								xxxx xxxx	uuuu uuuu
0Bh	INTCON0	ADIF	CWIF	T0IF	RAIF	—	—	—	GIE	0000 ---0	0000 ---0
Bank 1											
N/A	W(2)	Working Register (W)								xxxx xxxx	xxxx xxxx
N/A	TRIS	I/O Control Registers (TRISA, TRISB, TRISC)								1111 1111	1111 1111
N/A	OPTION	Contains control bits to configure Timer0 and Timer0/WDT prescaler								1111 1111	1111 1111
N/A	BSR(2)	—	—	—	—	—	—	BSR1	BSR0	---- -000	---- -0uu
20h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
21h	EECON	—	—	—	FREE	WRERR	WREN	WR	RD	---0 0000	---0 0000
22h	PCL(1)	Low-order eight bits of PC								1111 1111	1111 1111
23h	STATUS(2)	Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	-001 1xxx	-00q qqqq
24h	FSR(2)	—	Indirect data memory address pointer							0xxx xxxx	0uuu uuuu
25h	EEDATA	Self Read/Write Data								xxxx xxxx	uuuu uuuu
26h	EEADR	—	—	Self Read/Write Address						--xx xxxx	--uu uuuu
27h	CM1CON0	C1OUT	$\overline{C1OUTEN}$	C1POL	$\overline{C1T0CS}$	C1ON	C1NREF	C1PREF	$\overline{C1WU}$	1111 1111	quuu uuuu
28h	CM2CON0	C2OUT	$\overline{C2OUTEN}$	C2POL	C2PREF2	C2ON	C2NREF	C2PREF1	$\overline{C2WU}$	1111 1111	quuu uuuu
29h	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	001- 0000	uuu- uuuu
2Ah	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	1111 1111	1111 1111
2Bh	INTCON0	ADIF	CWIF	T0IF	RAIF	—	—	—	GIE	0000 ---0	0000 ---0

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0' (if applicable), q = value depends on condition.
Shaded cells = unimplemented or unused

Note 1: The upper byte of the Program Counter is not directly accessible. See **Section 4.6 "Program Counter"** for an explanation of how to access these bits.

2: Registers are implemented as two physical registers. When executing from within an ISR, a secondary register is used at the same logical location. Both registers are persistent. See **Section 8.11 "Interrupts"**.

3: These registers show the contents of the registers in the other context: ISR or main line code. See **Section 8.11 "Interrupts"**.

TABLE 4-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR/BOR	Value on all other Resets
Bank 2											
N/A	W(2)	Working Register (W)								xxxx xxxx	xxxx xxxx
N/A	TRIS	I/O Control Registers (TRISA, TRISB, TRISC)								1111 1111	1111 1111
N/A	OPTION	Contains control bits to configure Timer0 and Timer0/WDT prescaler								1111 1111	1111 1111
N/A	BSR(2)	—	—	—	—	—	—	BSR1	BSR0	---- -000	---- -0uu
40h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
41h	TMR0	Timer0 module Register								xxxx xxxx	uuuu uuuu
42h	PCL(1)	Low-order eight bits of PC								1111 1111	1111 1111
43h	STATUS(2)	Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	-001 1xxx	-00q qqqq
44h	FSR(2)	—	Indirect data memory address pointer							0xxx xxxx	0uuu uuuu
45h	OSCCAL	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—	1111 111-	uuuu uuu-
46h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--uu uuuu
47h	PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	uuuu ----
48h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
49h	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	1111 1100	1111 1100
4Ah	ADRES	ADC Conversion Result								xxxx xxxx	uuuu uuuu
4Bh	INTCON0	ADIF	CWIF	T0IF	RAIF	—	—	—	GIE	0000 ---0	0000 ---0
Bank 3											
N/A	W(2)	Working Register (W)								xxxx xxxx	xxxx xxxx
N/A	TRIS	I/O Control Registers (TRISA, TRISB, TRISC)								1111 1111	1111 1111
N/A	OPTION	Contains control bits to configure Timer0 and Timer0/WDT prescaler								1111 1111	1111 1111
N/A	BSR(2)	—	—	—	—	—	—	BSR1	BSR0	---- -000	---- -0uu
60h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
61h	IW(3)	Interrupt Working Register. (Addressed also as W register when within ISR)								xxxx xxxx	xxxx xxxx
62h	PCL(1)	Low-order eight bits of PC								1111 1111	1111 1111
63h	STATUS(2)	Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	-001 1xxx	-00q qqqq
64h	FSR(2)	—	Indirect data memory address pointer							0xxx xxxx	0uuu uuuu
65h	INTCON1	ADIE	CWIE	T0IE	RAIE	—	—	—	WUR	0000 ---0	0000 ---0
66h	ISTATUS(3)	Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	-xxx xxxx	-00q qqqq
67h	IFSR(3)	—	Indirect data memory address pointer							0xxx xxxx	0uuu uuuu
68h	IBSR(3)	—	—	—	—	—	—	BSR1	BSR0	---- -0xx	---- -0uu
69h	OPACON	—	—	—	—	—	—	OPA2ON	OPA1ON	---- --00	---- --00
6Bh	INTCON0	ADIF	CWIF	T0IF	RAIF	—	—	—	GIE	0000 ---0	0000 ---0

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0' (if applicable), q = value depends on condition.
Shaded cells = unimplemented or unused

Note 1: The upper byte of the Program Counter is not directly accessible. See **Section 4.6 "Program Counter"** for an explanation of how to access these bits.

2: Registers are implemented as two physical registers. When executing from within an ISR, a secondary register is used at the same logical location. Both registers are persistent. See **Section 8.11 "Interrupts"**.

3: These registers show the contents of the registers in the other context: ISR or main line code. See **Section 8.11 "Interrupts"**.

4.3 STATUS Register

This register contains the arithmetic status of the ALU, the Reset status and the page preselect bit.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS,` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

Therefore, it is recommended that only `BCF`, `BSF` and `MOVWF` instructions be used to alter the STATUS register. These instructions do not affect the Z, DC or C bits from the STATUS register. For other instructions which do affect Status bits, see **Section 13.0 "Instruction Set Summary"**.

REGISTER 4-1: STATUS: STATUS REGISTER

R-0	R-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
Reserved	Reserved	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **Reserved**: Read as '0'

bit 5 **PA0**: Program Page Preselect bit
1 = Page 1 (200h-3FFh)
0 = Page 0 (000h-1FFh)

bit 4 **$\overline{TO}$**: Time-Out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-Down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (for `ADDWF` and `SUBWF` instructions)
`ADDWF:`
1 = A carry from the 4th low-order bit of the result occurred
0 = A carry from the 4th low-order bit of the result did not occur
`SUBWF:`
1 = A borrow from the 4th low-order bit of the result did not occur
0 = A borrow from the 4th low-order bit of the result occurred

bit 0 **C**: Carry/$\overline{borrow}$ bit (for `ADDWF`, `SUBWF` and `RRF`, `RLF` instructions)
`ADDWF: SUBWF: RRF` or `RLF:`
1 = A carry occurred 1 = A borrow did not occur; Load bit with LSb or MSb, respectively
0 = A carry did not occur 0 = A borrow occurred

4.4 OPTION Register

The OPTION register is a 8-bit wide, write-only register, which contains various control bits to configure the Timer0/WDT prescaler and Timer0.

By executing the OPTION instruction, the contents of the W register will be transferred to the OPTION register. A Reset sets the OPTION <7:0> bits.

Note: If TRIS bit is set to '0', the wake-up on change and pull-up functions are disabled for that pin (i.e., note that TRIS overrides Option control of $\overline{RAPU}$ and $\overline{RAWU}$).

REGISTER 4-2: OPTION: OPTION REGISTER

W-1	W-1	W-1	W-1	W-1	W-1	W-1	W-1
$\overline{RAWU}$(2)	$\overline{RAPU}$	T0CS(1)	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **$\overline{RAWU}$:** Enable PORTA Interrupt Flag on Pin Change bit(2)
1 = Disabled
0 = Enabled

bit 6 **$\overline{RAPU}$:** Enable PORTA Weak Pull-Ups bit
1 = Disabled
0 = Enabled

bit 5 **T0CS:** Timer0 Clock Source Select bit(1)
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to Timer0

bit 2-0 **PS<2:0>:** Prescaler Rate Select bits

Bit Value	Timer0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Note 1: If the T0CS bit is set to '1', it will override the TRIS function on the T0CKI pin.
2: The $\overline{RAWU}$ bit of the OPTION register must be cleared to enable the RAIF function in the INTCON0 register.

4.5 OSCCAL Register

The Oscillator Calibration (OSCCAL) register is used to calibrate the 8 MHz internal oscillator macro. It contains seven bits of calibration that uses a two's complement scheme for controlling the oscillator speed. See Register 4-3 for details.

REGISTER 4-3: OSCCAL: OSCILLATOR CALIBRATION REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-1 **CAL<6:0>**: Oscillator Calibration bits

0111111 = Maximum frequency
•
•
•
0000001
0000000 = Center frequency
1111111
•
•
•
1000000 = Minimum frequency

bit 0 **Unimplemented**: Read as '0'

4.6 Program Counter

As a program instruction is executed, the Program Counter (PC) will contain the address of the next program instruction to be executed. The PC value is increased by one every instruction cycle, unless an instruction changes the PC.

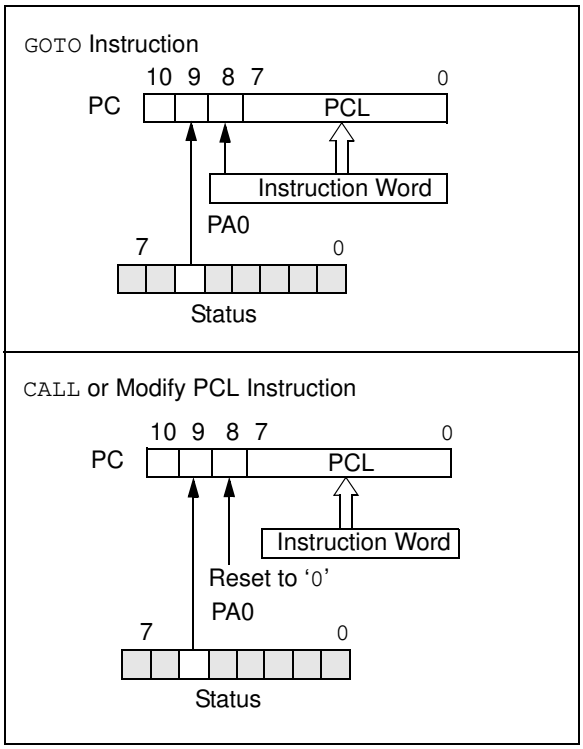
For a GOTO instruction, bits <8:0> of the PC are provided by the GOTO instruction word. The Program Counter (PCL) is mapped to PC<7:0>. Bit 5 of the STATUS register provides page information to bit 9 of the PC (see Figure 4-3).

For a CALL instruction, or any instruction where the PCL is the destination, bits <7:0> of the PC again are provided by the instruction word. However, PC<8> does not come from the instruction word, but is always cleared (see Figure 4-3).

Instructions where the PCL is the destination, or modify PCL instructions, include MOVWF PCL, ADDWF PCL and BSF PCL,5.

Note:	Because bit 8 of the PC is cleared in the CALL instruction or any modify PCL instruction, all subroutine calls or computed jumps are limited to the first 256 locations of any program memory page (512 words long).

FIGURE 4-3: LOADING OF PC BRANCH INSTRUCTIONS



4.6.1 EFFECTS OF RESET

The PC is set upon a Reset, which means that the PC addresses the last location in the last page (i.e., the oscillator calibration instruction). After executing MOVLW XX, the PC will roll over to location 00h and begin executing user code.

The STATUS register page preselect bits are cleared upon a Reset, which means that page 0 is pre-selected.

Therefore, upon a Reset, a GOTO instruction will automatically cause the program to jump to page 0 until the value of the page bits is altered.

4.7 Stack

The PIC16F527 device has a 4-deep, 12-bit wide hardware PUSH/POP stack.

A CALL instruction or an interrupt will PUSH the current PC value, incremented by one, into Stack Level 1. If there was a previous value in the Stack 1 location, it will be pushed into the Stack 2 location. This process will be continued throughout the remaining stack locations populated with values. If more than four sequential CALLs are executed, only the most recent four return addresses are stored.

A RETLW, RETURN or RETFIE instruction will POP the contents of Stack Level 1 into the PC. If there was a previous value in the Stack 2 location, it will be copied into the Stack Level 1 location. This process will be continued throughout the remaining stack locations populated with values. If more than four sequential RETLWs are executed, the stack will be filled with the address previously stored in Stack Level 4. Note that the W register will be loaded with the literal value specified in the instruction. This is particularly useful for the implementation of data look-up tables within the program memory.

Note 1:	There are no Status bits to indicate Stack Overflows or Stack Underflow conditions.
2:	There are no instruction mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETFIE and RETLW instructions.

4.8 Direct and Indirect Addressing

4.8.1 DIRECT DATA ADDRESSING: BSR REGISTER

Traditional data memory addressing is performed in the Direct Addressing mode. In Direct Addressing, the Bank Select Register bits BSR<1:0>, in the new BSR register, are used to select the data memory bank. The address location within that bank comes directly from the opcode being executed.

BSR<1:0> are the bank select bits and are used to select the bank to be addressed (00 = Bank 0, 01 = Bank 1, 10 = Bank 2, 11 = Bank 3).

A new instruction supports the addition of the BSR register, called the MOVLB instruction. See **Section 13.0 "Instruction Set Summary"** for more information.

4.8.2 INDIRECT DATA ADDRESSING: INDF AND FSR REGISTERS

The INDF Register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR Register (FSR is a *pointer*). This is indirect addressing.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF Register indirectly results in a no-operation (although Status bits may be affected).

The FSR is an 8-bit wide register. It is used in conjunction with the INDF Register to indirectly address the data memory area.

The FSR<6:0> bits are used to select data memory addresses 00h to 1Fh.

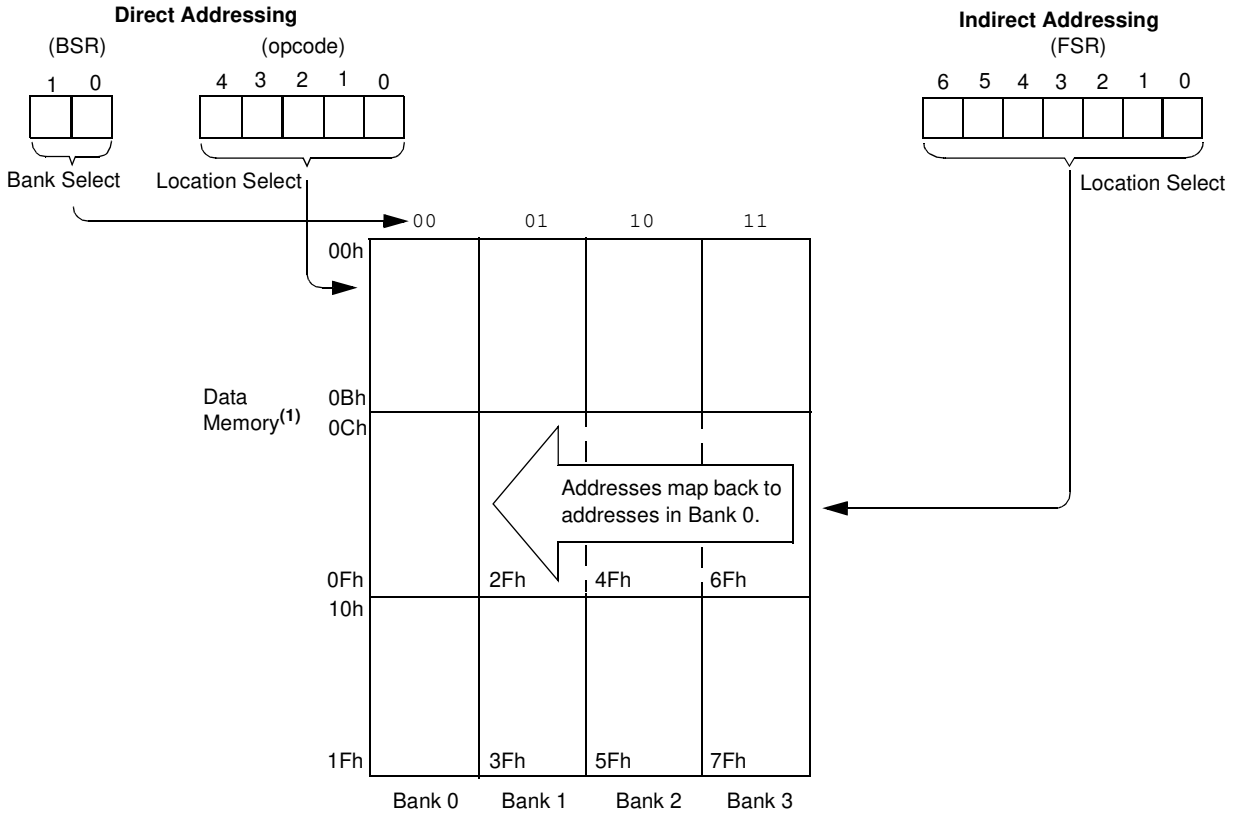
FSR<7> is unimplemented and read as '0'.

A simple program to clear RAM locations 10h-1Fh using indirect addressing is shown in Example 4-1.

EXAMPLE 4-1: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
        MOVLW   0x10    ;initialize pointer
        MOVWF   FSR     ;to RAM
NEXT    CLRF    INDF    ;clear INDF
                        ;register
        INCF    FSR,F   ;inc pointer
        BTFSC   FSR,4   ;all done?
        GOTO    NEXT    ;NO, clear next
CONTINUE
            :           ;YES, continue
            :
```

FIGURE 4-4: DIRECT/INDIRECT ADDRESSING



Note 1: For register map detail see Figure 4-2.

5.0 SELF-WRITABLE FLASH DATA MEMORY CONTROL

Flash Data memory consists of 64 bytes of self-writable memory and supports a self-write capability that can write a single byte of memory at one time. Data to be written to the self-writable data memory is first written into a write latch before writing the data to Flash memory.

Although each Flash data memory location is 12 bits wide, access is limited to the lower eight bits. The upper four bits will automatically default to '1' in any self-write procedure. The lower eight bits are fully readable and writable during normal operation and throughout the full VDD range.

The self-writable Flash data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers, EECON, EEDATA and EEADR.

5.1 Reading Flash Data Memory

To read a Flash data memory location the user must:

- Write the EEADR register
- Set the RD bit of the EECON register

The value written to the EEADR register determines which Flash data memory location is read. Setting the RD bit of the EECON register initiates the read. Data from the Flash data memory read is available in the EEDATA register immediately. The EEDATA register will hold this value until another read is initiated or it is modified by a write operation. Program execution is suspended while the read cycle is in progress. Execution will continue with the instruction following the one that sets the WR bit. See Example 5-1 for sample code.

EXAMPLE 5-1: READING FROM FLASH DATA MEMORY

```
MOVLB     0x01           ; Switch to Bank 1
MOVF      DATA_EE_ADDR,W;
MOVWF     EEADR          ; Data Memory
                         ; Address to read
BSF       EECON, RD      ; EE Read
MOVF      EEDATA, W      ; W = EEDATA
```

Note: Only a BSF command will work to enable the Flash data memory read documented in Example 5-1. No other sequence of commands will work, no exceptions.

Note 1: To prevent accidental corruption of the Flash data memory, an unlock sequence is required to initiate a write or erase cycle. This sequence requires that the bit set instructions used to configure the EECON register happen exactly as shown in Example 5-2 and Example 5-3, depending on the operation requested.

2: In order to prevent any disruptions of self-writes or row erases performed on the self-writable Flash data memory, interrupts should be disabled prior to executing those routines.

5.1.1 ERASING FLASH DATA MEMORY

A row must be manually erased before writing new data. The following sequence must be performed for a single row erase.

1. Load EEADR with an address in the row to be erased.
2. Set the FREE bit to enable the erase.
3. Set the WREN bit to enable write access to the array.
4. Disable interrupts.
5. Set the WR bit to initiate the erase cycle.

If the WREN bit is not set in the instruction cycle after the FREE bit is set, the FREE bit will be cleared in hardware.

If the WR bit is not set in the instruction cycle after the WREN bit is set, the WREN bit will be cleared in hardware.

Sample code that follows this procedure is included in Example 5-2.

Program execution is suspended while the erase cycle is in progress. Execution will continue with the instruction following the one that sets the WR bit.

EXAMPLE 5-2: ERASING A FLASH DATA MEMORY ROW

```
MOVLB     0x01           ; Switch to Bank 1
MOVLW     EE_ADR_ERASE   ; LOAD ADDRESS OF ROW TO
                         ; ERASE
MOVWF     EEADR          ;
BSF       EECON,FREE     ; SELECT ERASE
BSF       EECON,WREN     ; ENABLE WRITES
BSF       EECON,WR       ; INITITATE ERASE
```

Note 1: The FREE bit may be set by any command normally used by the core. However, the WREN and WR bits can only be set using a series of BSF commands, as documented in Example 5-1. No other sequence of commands will work, no exceptions.

2: Bits <5:3> of the EEADR register indicate which row is to be erased.

5.1.2 WRITING TO FLASH DATA MEMORY

Once a cell is erased, new data can be written. Program execution is suspended during the write cycle.

The self-write operation writes one byte of data at one time. The data must first be loaded into a write latch. Once the write latch is loaded, the data will be written to Flash data memory.

The self-write sequence is shown below.

1. Load EEADR with the address.
2. Load EEDATA with the data to be written.
3. Set the WREN bit to enable write access to the array.
4. Disable interrupts.
5. Set the WR bit to load the data into the write latch.

Once the WR bit is set and the processor recognizes that the write latch is loaded, it will immediately perform the Flash data memory write of that byte.

The specific sequence of setting the WREN bit and setting the WR bit must be executed to properly initiate loading of the write latches and the write to Flash data memory.

If the WR bit is not set in the instruction cycle after the WREN bit is set, the WREN bit will be cleared in hardware.

Sample code that follows this procedure is included in Example 5-3.

EXAMPLE 5-3: WRITING TO FLASH DATA MEMORY

```
MOVLW   EE_ADR_WRITE    ;LOAD ADDRESS
MOVWF   EEADR           ;INTO EEADR
                        ;REGISTER
MOVLW   EE_DATA_WRITE   ;LOAD DATA
MOVWF   EEDATA          ;INTO EEDATA
                        ;REGISTER
BSF     EECON, WREN     ;ENABLE WRITES
BCF     INTCON, GIE     ;DISABLE INTERRUPTS
BSF     EECON,WR        ;LOAD WRITE LATCH
                        ;AND PERFORM DATA
                        ;MEMORY WRITE
```

Note 1: Only a series of BSF commands will work to enable the memory write sequence documented in Example 5-3. No other sequence of commands will work, no exceptions.

2: For reads, erases and writes to the Flash data memory, there is no need to insert a NOP into the user code as is done on mid-range devices. The instruction immediately following the "BSF EECON,WR/RD" will be fetched and executed properly.

5.2 Write/Verify

Depending on the application, good programming practice may dictate that data written to the Flash data memory be verified. Example 5-4 is an example of a write/verify.

EXAMPLE 5-4: WRITE/VERIFY OF FLASH DATA MEMORY

```
MOVF    EEDATA, W   ;EEDATA has not changed
                    ;from previous write
BSF     EECON, RD   ;Read the value written
XORWF   EEDATA, W   ;
BTFSS   STATUS, Z   ;Is data the same
GOTO    WRITE_ERR   ;No, handle error
                    ;Yes, continue
```

5.3 Register Definitions — Memory Control

REGISTER 5-1: EEDATA: FLASH DATA REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EEDATA7	EEDATA6	EEDATA5	EEDATA4	EEDATA3	EEDATA2	EEDATA1	EEDATA0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **EEDATA<7:0>**: Eight bits of data to be read from/written to data Flash

REGISTER 5-2: EEADR: FLASH ADDRESS REGISTER

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'.

bit 5-0 **EEADR<5:0>**: Six bits of data to be read from/written to data Flash

REGISTER 5-3: EECON: FLASH CONTROL REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Legend:			
S = Bit can only be set			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'.

bit 4 **FREE:** Flash Data Memory Row Erase Enable bit
- 1 = Program memory row being pointed to by EEADR will be erased on the next write cycle. No write will be performed. This bit is cleared at the completion of the erase operation.
- 0 = Perform write only

bit 3 **WRERR:** Write Error Flag bit
- 1 = A write operation terminated prematurely (by device Reset)
- 0 = Write operation completed successfully

bit 2 **WREN:** Write Enable bit
- 1 = Allows write cycle to Flash data memory
- 0 = Inhibits write cycle to Flash data memory

bit 1 **WR:** Write Control bit
- 1 = Initiate a erase or write cycle
- 0 = Write/Erase cycle is complete

bit 0 **RD:** Read Control bit
- 1 = Initiate a read of Flash data memory
- 0 = Do not read Flash data memory

5.4 Code Protection

Code protection does not prevent the CPU from performing read or write operations on the Flash data memory. Refer to the code protection chapter for more information.