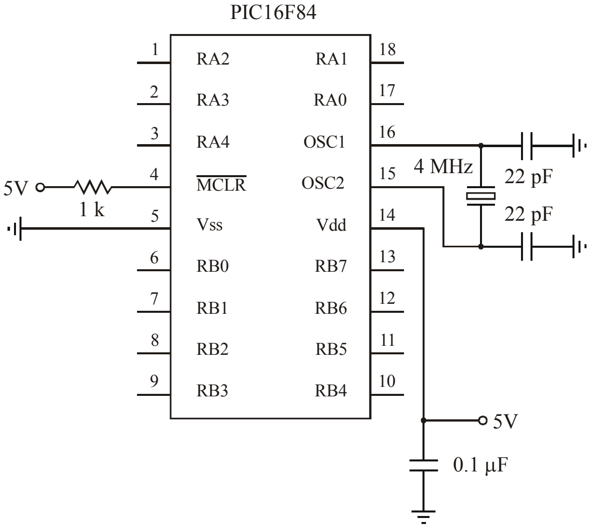
Chapter 7 Summary
Microcontroller Programming and Interfacing



PIC16F84 pin-out and required external components

PIC16F84 instruction set

Mnemonic and Operands	Description
ADDLW k	add literal and W
ADDWF f, d	add W and f
ANDLW k	AND literal with W
ANDWF f, d	AND W with f
BCF f, b	bit clear f
BSF f, b	bit set f
BTFSC f, b	bit test f, skip if clear
BTFSS f, b	bit test f, skip if set
CALL k	call subroutine
CLRF f	clear f
CLRW	clear W
CLRWDT	clear Watchdog Timer

Mnemonic and Operands	Description
COMF f, d	complement f
DECF f, d	decrement f
DECFSZ f, d	decrement f, Skip if 0
GOTO k	go to address
INCF f, d	increment f
INCFSZ f, d	increment f, skip if 0
IORLW k	inclusive OR literal with W
IORWF f, d	inclusive OR W with f
MOVF f, d	move f
MOVLW k	move literal to W
MOVWF f	move W to f
NOP	no operation
RETFIE	return from interrupt
RETLW k	return with literal in W
RETURN	return from subroutine
RLF f, d	rotate f left 1 bit
RRF f, d	rotate f right 1 bit
SLEEP	go into standby mode
SUBLW k	subtract W from literal
SUBWF f, d	subtract W from f
SWAPF f, d	Swap nibbles in f
XORLW k	Exclusive OR literal with W
XORWF f, d	Exclusive OR W with f

Selected PicBasic Pro math operators and functions

math operator or function	description
A + B	add A and B
A – B	subtract B from A
A * B	multiply A and B
A / B	divide A by B

math operator or function	description
A << n	shift A n bits to the left
A >> n	shift A n bits to the right
COS A	return the cosine of A
A MAX B	return the maximum of A and B
A MIN B	return the minimum of A and B
SIN A	return the sine of A
SQR A	return the square root of A
A & B	return the bitwise AND of A and B
A | B	return the bitwise OR of A and B
A ^ B	return the bitwise exclusive OR of A and B
~A	return the bitwise NOT of A

PicBasic Pro logical comparison operators

operator	description
= or ==	equal
<> or !=	not equal
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

PicBasic Pro statement summary

Statement	Description
@ assembly statement	insert one line of assembly language code
ADCIN channel, var	read the on-chip analog to digital converter (if there is one)
ASM ... ENDASM	insert an assembly language code section consisting of one or more statements
BRANCH index, [label1{, label2, ...}]	computed GOTO that jumps to a label based on index

Statement	Description
BRANCHL index, [label1{, label2, ...}]	BRANCH to a label that can be outside of the current page of code memory (for PICs with more than 2k of program ROM)
BUTTON pin, down_state, auto_repeat_delay, auto_repeat_rate, countdown_variable, action_state, label	read the state of a pin and perform debounce (by use of a delay) and auto-repeat (if used within a loop)
CALL assembly_label	call an assembly language subroutine
CLEAR	zero all variables
CLEARWDT	clear the Watchdog Timer
COUNT pin, period, var	count the number of pulses occurring on a pin during a period
DATA {@ location,} constant1{, constant2, ...}	define initial contents of the on-chip EEPROM (same as the EEPROM statement)
DEBUG item1{, item2, ...}	asynchronous serial output to a pin at a fixed baud rate
DEBUGIN {timeout, label,} [item1{,{item2, ...}]	asynchronous serial input from a pin at a fixed baud rate
DISABLE	disable ON INTERRUPT and ON DEBUG processing
DISABLE DEBUG	disable ON DEBUG processing
DISABLE INTERRUPT	disable ON INTERRUPT processing
DTMFOUT pin, {on_ms, off_ms,} [tone1{, tone2, ...}]	produce touch-tones on a pin
{EEPROM {@ location,} constant1{, constant2, ...}}	define initial contents of on-chip EEPROM (same as DATA statement)
ENABLE	enable ON INTERRUPT and ON DEBUG processing
ENABLE DEBUG	enable ON DEBUG processing
ENABLE INTERRUPT	enable ON INTERRUPT processing
END	stop execution and enter low power mode

Statement	Description

FOR count = start TO end {STEP {-} inc}
{body statements}
NEXT {count}
	repeatedly execute statements as count goes from start to end in fixed increment

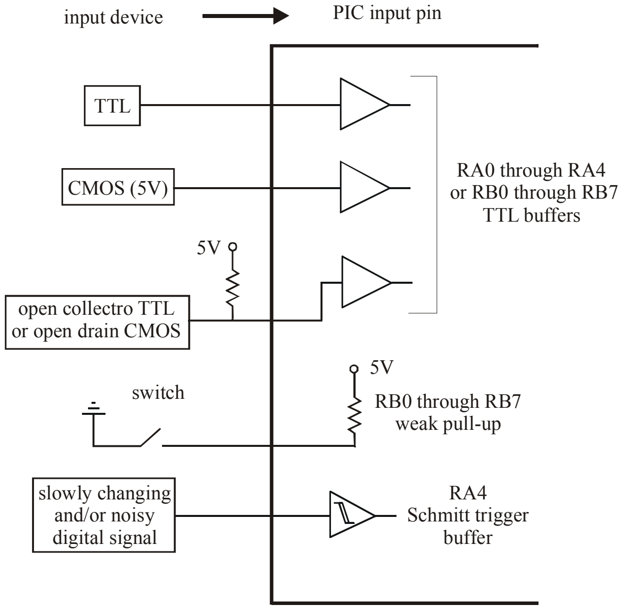
FREQOUT pin, on_ms, freq1{, freq2}	produce up to two frequencies on a pin
GOSUB label	call a PicBasic subroutine at the specified label
GOTO label	continue execution at the specified label
HIGH pin	make pin output high
HSERIN {parity_label,} {time_out, label,} [item1{, item2, ...}]	hardware asynchronous serial input (if there is a hardware serial port)
HSEROUT [item1{, item2, ...}]	hardware asynchronous serial output (if there is a hardware serial port)
I2CREAD data_pin, clock_pin, control,{ address,} [var1{, var2, ...}]{, label}	read bytes from an external I^{2}C serial EEPROM device
I2CWRITE data_pin, clock_pin, control,{ address,} [var1{, var2, ...}]{, label}	write bytes to an external I^{2}C serial EEPROM device
IF log_comp THEN label	conditionally jump to a label

IF log_comp THEN
true_statements
ELSE
false_statements
ENDIF
	conditional execution of statements

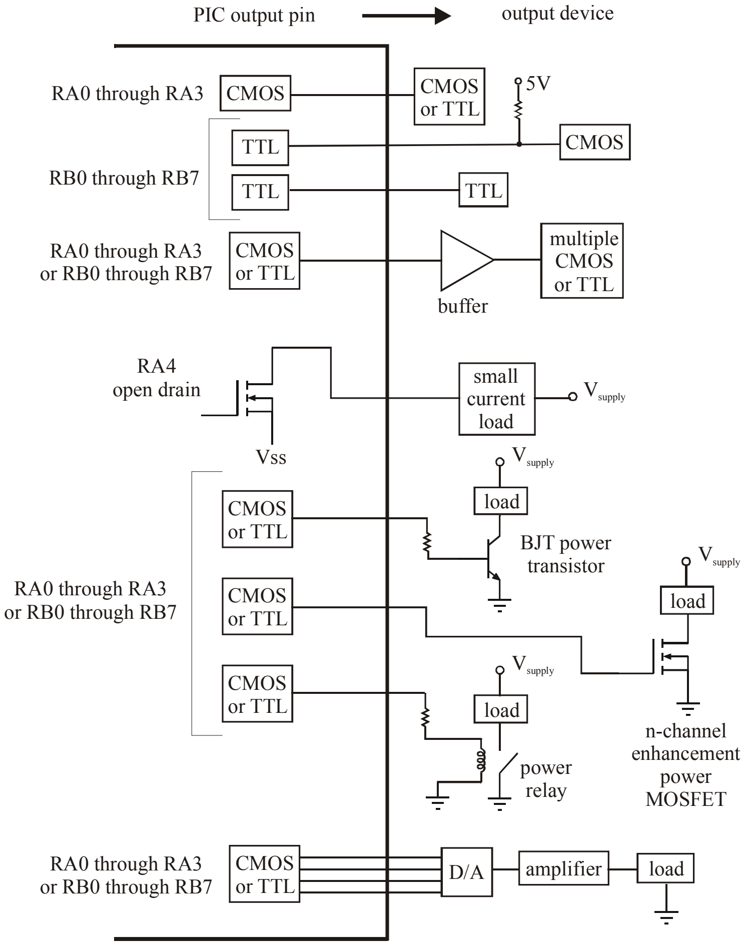
INPUT pin	make pin an input
LCDIN {address,} [var1{, var2, ...}]	read RAM on a liquid crystal display (LCD)
LCDOUT item1{, item2, ...}	display characters on a liquid crystal display (LCD)
{LET} var = value	assignment statement (assigns a value to a variable)
LOOKDOWN value, [const1{, const2, ...}], var	search constant table for a value
LOOKDOWN2 value,{ test} [value1{, value2, ...}], var	search constant / variable table for a value

Statement	Description
LOOKUP index, [const1{, const2, ...}], var	fetch constant value from a table
LOOKUP2 index, [value1{, value2, ...}], var	fetch constant / variable value from a table
LOW pin	make pin output low
NAP period	power down processor for a selected period of time
ON DEBUG GOTO label	execute PicBasic debug subroutine at label after every statement if debug is enabled
ON INTERRUPT GOTO label	execute PicBasic subroutine at label when an interrupt is detected
OUTPUT pin	make pin an output
PAUSE period	delay a given number of milliseconds
PAUSEUS period	delay a given number of microseconds
{PEEK address, var}	read byte from a register
{POKE address, var}	write byte to a register
POT pin, scale, var	read resistance of a potentiometer, or other variable resistance device, connected to a pin with a series capacitor to ground
PULSIN pin, state, var	measure the width of a pulse on a pin
PULSOUT pin, period	generate a pulse on a pin
PWM pin, duty, cycles	output a pulse width modulated (PWM) pulse train to pin
RANDOM var	generate a pseudo-random number
RCTIME pin, state, var	measure pulse width on a pin
READ address, var	read a byte from on-chip EEPROM
READCODE address, var	read a word from code memory
RESUME {label}	continue execution after interrupt handling
RETURN	continue execution at the statement following last executed GOSUB
REVERSE pin	make output pin an input or an input pin an output

Statement	Description
SERIN pin, mode,{ timeout, label,} {[qual1, qual2, ...],}{ item1{, item2, ...}}	asynchronous serial input (Basic Stamp 1 style)
SERIN2 data_pin{\flow_pin}, mode, {parity_label,} {timeout, label,} [item1{, item2, ...}]	asynchronous serial input (Basic Stamp 2 style)
SEROUT pin, mode, [item1{, item2, ...}]	asynchronous serial output (Basic Stamp 1 style)
SEROUT2 data_pin{\flow_pin}, mode, {pace,} {timeout, label,} [item1{, item2, ...}]	asynchronous serial output (Basic Stamp 2 style)
SHIFTIN data_pin, clock_pin, mode, [var1{\bits1}{, var2{\bits2}, ...}]	synchronous serial input
SHIFTOUT data_pin, clock_pin, mode, [var1{\bits1}{, var2{\bits2}, ...}]	synchronous serial output
SLEEP period	power down the processor for a given number of seconds
SOUND pin, [note1, duration1{, note2, duration2, ...}]	generate a tone or white-noise on a specified pin
STOP	stop program execution
SWAP var1, var2	exchange the values of two variables
TOGGLE pin	change the state of an output pin
WHILE logical_comp statements WEND	execute code while condition is true
WRITE address, value	write a byte to on-chip EEPROM
WRITECODE address, value	write a word to code memory
XIN data_pin, zero_pin, {timeout, label,} [var1{, var2, ...}]]	receive data from an external X-10 type device
XOUT data_pin, zero_pin, [house_code1\key_code1{\repeat1}{, house_code2\key_code2{\repeat2, ...}]	send data to an external X-10 type device



Interface circuits for input devices



Interface circuits for output devices

Method to Design a Microcontroller-based System:

1. define the problem
2. select an appropriate microcontroller model
3. identify necessary interface circuits
4. decide on a programming language
5. draw the schematic
6. draw a program flowchart
7. write the code
8. build and test the system