



The **QUINTESSENTIAL PIC[®] MICROCONTROLLER** deals with the central 'intelligence' of most smart embedded digital systems and aims to give the reader the confidence to design, construct and program a real working system using the industrial standard and popular Microchip PIC family of devices as the exemplar.



Aimed specifically at a readership with no prior knowledge of software, electronics or logic design, this book is suitable for both industrial engineers and hobbyists. Students of Electronic Engineering and Computer Science, taking relevant courses or engaged in project work at either an under- or postgraduate level, will find it an ideal textbook.



The second edition has been extensively rewritten to both rationalize and clarify the introductory chapters and to update and extend the core material. Whilst the thrust remains with the Mid-range devices, an additional chapter has also been added on the Extended-range family.

Key features include:

- The use of current real-world hardware and software products to illustrate the material
- Numerous fully worked examples as well as self-assessment questions
- An associated web site providing solutions, further examples, listings and useful resources at <http://www.usnj.usfsc.us/click/quintessential>
- Clear and concise presentation of key points and underlying concepts.

ISBN 185233-942-X

springeronline.com



Katzen
The Quintessential PIC[®] Microcontroller

Second Edition



The Quintessential PIC[®] Microcontroller



Sidney J. Katzen

Second Edition

Springer

Contents

Preface to the Second Edition	IX
Preface to the First Edition	XI
<hr/>	
Part I The Fundamentals	
<hr/>	
1. Digital Representation	3
2. Logic Circuitry	17
3. Stored Program Processing	43
<hr/>	
Part II The Software	
<hr/>	
4. The PIC16F84 Microcontroller	71
5. The Instruction Set	95
6. Subroutines and Modules	145
7. Interrupt Handling	181
8. Assembly Language Code Building Tools	209
9. High-Level Language	243
<hr/>	
Part III The Outside World	
<hr/>	
10. The Real World	267
11. One Byte at a Time	289
12. One Bit at a Time	327

VIII Contents

13. Time Is of the Essence	397
14. Take the Rough with the Smooth	431
15. To Have and to Hold	479
16. Enhancing the Family	505
17. A Case Study	525

Appendices

A. Acronyms and Abbreviations	545
B. Special-Purpose Register Structure for the PIC16F87XA	551
C. C Instruction Set	553
Index	555

List of Figures

1.1	The NOT operation.....	12
1.2	The AND function.....	13
1.3	The inclusive-OR operation.....	13
1.4	The XOR operation.....	14
1.5	Detecting sign overflow.....	15
2.1	The 74LS00 quad 2-I/P NAND package.....	18
2.2	Output structures.....	19
2.3	Open-collector buffers driving a party line.....	20
2.4	Sharing a bus.....	20
2.5	The 74LS138 and 74LS139 MSI natural decoders.....	21
2.6	The 74LS148 highest-priority encoder.....	22
2.7	The 74LS688 octal equality detector.....	23
2.8	Addition.....	24
2.9	Implementing a programmable adder/subtractor.....	25
2.10	The 74LS382 ALU.....	25
2.11	A ROM-implemented 1-bit adder.....	26
2.12	The 2764 erasable PROM (EPROM).....	27
2.13	Floating-gate MOSFET link.....	28
2.14	The RS latch.....	29
2.15	Using a \overline{RS} latch to debounce a switch.....	31
2.16	The D latch and flip flop.....	31
2.17	The 74LS74 dual D flip flop.....	32
2.18	The 74LS377 octal D flip flop array.....	33
2.19	The 74LS373 octal D latch array.....	34
2.20	An 8-bit ALU-accumulator processor.....	35
2.21	A system look at our ALU-accumulator processor.....	36
2.22	The SISO shift register.....	37
2.23	The T flip flop.....	38
2.24	A modulo-16 ripple counter.....	39
2.25	Generating timing waveforms.....	40
2.26	The 6264 8196 \times 8 RAM.....	41
3.1	An elementary von Neumann computer.....	44
3.2	An elementary Harvard architecture computer.....	46

X List of Figures

3.3	A system look at a rudimentary Harvard computer.	48
3.4	Executing the 1st instruction whilst fetching down the 2nd. .	49
3.5	Machine code structure for direct instructions.	53
3.6	Machine code structure for literal instructions.	54
3.7	Parallel fetch and execute streams.	55
3.8	An example of a system based on a microcontroller.	62
3.9	A greenhouse environmental controller.	64
3.10	A flow chart showing the robot how to cross the road.	66
4.1	Architecture of the PIC16F84 microcontroller.	73
4.2	A close-up look at the PIC16F84's fetch unit.	74
4.3	A simplified look at the PIC16F84's Program store.	75
4.4	Internal clock sequencing waveforms.	76
4.5	A close-up view of the execute unit.	77
4.6	The PIC16F84 Status register.	78
4.7	A simplified look at the PIC16F84's Data store.	80
4.8	Showing how all of the PC are altered when writing to PCL. ..	86
5.1	Generating a 13-bit Program-store address.	98
5.2	Selecting the destination from the instruction <code>addwf h'26'</code> . .	99
5.3	The PIC16F84 Data store.	101
5.4	The PIC16F627/8 Data store.	102
5.5	General 14-bit core Status register.	103
5.6	The Indirect addressing mechanism.	104
5.7	Using a loop to clear an array of data.	105
5.8	Walking through an array.	106
5.9	The process.	113
5.10	Visualisation of the task.	114
5.11	Comparing the contents of W with <code>subwf h'26'</code>	119
5.12	Comparisons made in the fuel warning system.	120
5.13	Rotating the contents of a File once right.	126
5.14	Finding the leftmost 1 by continuous shifting.	127
5.15	Rotating the contents of a File once left.	129
5.16	Shifting a double-byte datum once to the left to multiply by two.	130
5.17	Jumping to instruction 1018.	131
5.18	Skipping whenever bit 7 of File <code>h'20'</code> is clear.	132
5.19	Pulse pin RA0 repeating 20 times.	133
5.20	Conversion of a byte up to 99 to BCD.	137
6.1	Modular hardware implementing a PC.	146
6.2	Subroutine calling.	148
6.3	Using the hardware stack to hold return addresses.	149
6.4	Nested subroutines.	150
6.5	Delaying by counting <i>N</i> times.	151

6.6	A nested loop delay algorithm.....	154
6.7	System view of $K \times 100$ ms delay subroutine.....	156
6.8	The 7-segment display.	158
6.9	System diagram for the byte multiplication subroutine.	161
6.10	The stack frame.....	165
6.11	Finding the square root of an integer.	172
6.12	A 7-bit pseudo-random number generator.	180
7.1	Detecting and measuring an external event.	182
7.2	Responding to an interrupt request.....	186
7.3	The PIC16F84's interrupt logic.	187
7.4	Monitoring customers entering the shop.....	188
7.5	The PIC16F627/8 MCU's interrupt logic.....	195
7.6	A register view of the PIC16F627/8 MCU's interrupt system..	196
7.7	Oven safety hardware.	200
7.8	Coin entry for a vending machine.	205
7.9	Echo sounding hardware.	207
8.1	Conversion from assembly-level source to machine code.....	210
8.2	Absolute assembly-level code translation.	215
8.3	Relocatable assembly-level code translation.	223
8.4	Linking three source files.....	225
8.5	Code building and testing tools.	232
8.6	MPLAB® IDE Project window.....	233
8.7	MPLAB IDE screen shot.	234
9.1	Conversion from high-level source code to machine code. ...	245
9.2	Pyramid view of the steps leading to executable code.	246
9.3	Simulating our example program in MPLAB Version 6.60.....	254
9.4	The active-low die patterns.	263
10.1	Architecture of the PIC16F874/77A devices.....	268
10.2	Pinout for a variety of PIC MCU family members.	269
10.3	Typical supply current versus clocking frequency.....	271
10.4	Equivalent output circuit.	272
10.5	Typical oscillator configurations.....	275
10.6	Configuring some representative mid-range PIC MCUs.	277
10.7	Configuration menu for the PICSTART® programmer.	278
10.8	Externally resetting the PIC MCU.	280
10.9	The PCON register for the mid-range PIC16XXXX family.....	281
10.10	The sequence of events leading to startup on Power-on.	282
10.11	A Brown-out reset due to a blip on the power supply.	284
10.12	Variable oscillator frequency.	286
10.13	An alternative oscillator frequency control circuit.	287

XII List of Figures

11.1	The Mid-range PIC16XXXX series Parallel Ports A and B.	290
11.2	Outputting data from Port B using a handshake transfer.	293
11.3	A simplified typical I/O port line.	296
11.4	Reading from and writing to a port bit.	298
11.5	Sinking and sourcing current.	299
11.6	Power dissipation model.	300
11.7	Output driver structures.	302
11.8	Interfacing switches to a port pin.	303
11.9	Port B's weak pull-up resistors.	304
11.10	Interfacing to a keypad.	305
11.11	The Port B change feature.	309
11.12	A multi-zone intruder alarm.	312
11.13	Source current against voltage.	315
11.14	The stepper motor.	318
11.15	Using port expansion to drive three 7-segment displays.	321
11.16	Scanning a multiplexed 3-digit 7-segment array.	322
11.17	Low-level output voltage against sink current.	326
12.1	The smart card.	327
12.2	Serial interface to a 3-digit 7-segment display.	329
12.3	Serially interfacing to a DAC digital to analog converter.	332
12.4	Serially interfacing to the multi-zone intruder alarm.	333
12.5	The MAXIM MAX549A SPI dual 8-bit DAC.	337
12.6	SPI waveforms for the MAX549A.	339
12.7	Multiple MAX549As on the one SPI circuit.	339
12.8	The basic Serial Synchronous Port.	340
12.9	The MSSP module's SPI-mode CONTROL and STATUS registers.	341
12.10	Clocking data in and out to remote Slave devices.	343
12.11	SSP SPI-mode Master waveforms.	347
12.12	A multidrop SPI communications network.	348
12.13	Data transfer on the I ² C bus.	350
12.14	Sharing the SCL and SDA bus lines.	352
12.15	A I ² C packet transmission.	353
12.16	The MAXIM MAX518 I ² C dual digital to analog converter.	354
12.17	Minimum timing relationships for the Fast I ² C mode.	355
12.18	Block diagram of a MSSP module set-up as an I ² C Slave device.	361
12.19	The Master I ² C SSP CONTROL and STATUS registers.	362
12.20	Transmitting the string "PIC" in the asynchronous mode ...	372
12.21	The generic UART.	376
12.22	The Serial Peripheral Interface module.	377
12.23	A local area network using an asynchronous serial protocol.	379
12.24	Some signaling configurations.	384
12.25	Communicating with a PC via an RS-232 link.	386
12.26	The 24XXX series of I ² C serial EEPROMs.	389
12.27	EEPROM Read and Write waveforms.	390

12.28	Interfacing the DS18S20 1-Wire digital thermometer.	393
13.1	The integral PIC MCU Watchdog timer with Postscaler.	400
13.2	The Option register.	401
13.3	Simplified equivalent circuit for Timer 0.	403
13.4	Counting cans of beans on a conveyer belt.....	405
13.5	Functional equivalent circuit for Timer 1	411
13.6	Capturing the time of an event.	415
13.7	The CCP1 module set to Compare mode.	418
13.8	A simplified equivalent circuit for Timer 2.	420
13.9	Pulse width modulation.	422
13.10	Timer 2 and the PWM CCP mode.	423
13.11	An event manifesting itself as a pulse duration.	430
14.1	Analog world - digital processing.	435
14.2	The quantizing process.	437
14.3	The analog-digital process.	440
14.4	Illustrating aliasing.....	441
14.5	Using an analog comparator to determine the ECG start.	442
14.6	The Comparator module.	443
14.7	The Comparator Voltage Reference module.....	445
14.8	Initializing the 8-4-2-1 capacitor network for a 4-bit converter.	449
14.9	Simplified view of a successive approximation A/D converter.	450
14.10	The successive approximation process.	452
14.11	The PIC16F87X 10-bit 8-channel ADC module.....	454
14.12	Configuring the analog inputs for Port A and Port E.	456
14.13	Aligning the 10-bit digital outcome in a 16-bit field.	458
14.14	Timeline for the conversion process.	459
14.15	R-2R digital-to-analog conversion.	467
14.16	The Maxim MAX506 quad 8-bit D/A converter.	468
14.17	Generating a continuous sawtooth using a MAX506 DAC.....	470
14.18	A level-shifting resistor network.	471
14.19	ECG detection strategy.	472
14.20	Measuring the discharge power for an ECG defibrillator.	476
14.21	A controllable external voltage circuit.	482
15.1	The PIC16F62X EEPROM Data module.	485
15.2	The PIC16F62X EECON1 register.	486
15.3	The first 32 bytes of EEPROM.	490
15.4	The PIC16F87X Flash and Data EEPROM storage system.	491
15.5	View of the Flash Program module.	494
15.6	Configuration word for the PIC16F87XA devices.	496
15.7	Writing to Flash memory with the PIC16F87XA group.	497
15.8	Watchdog timer period versus temperature.....	501

XIV List of Figures

16.1	Architecture of the 40-pin PIC18F442/52 devices.	510
16.2	Simplified look at the PIC18FX42 Program store.	512
16.3	The Enhanced-range Data store structure.	513
16.4	Machine-code structure of a typical 16-bit instruction.....	515
16.5	Indirect addressing via FRS0.....	516
16.6	A simplified typical Enhanced-range I/O port line.	518
17.1	The annunciator hardware.	530
17.2	The modular software structure.	533
17.3	The Main process.	543
17.4	Programming the PIC from the MPLAB Version 5 IDE.....	547
17.5	The Microchip PICSTART Plus programmer.	548

List of Tables

1.1	7-bit ASCII characters.	5
1.2	Some common bit groupings.	6
1.3	Different ways of representing the quantities decimal 0,...,20.	7
5.1	Move instructions.	109
5.2	Arithmetic.	112
5.3	Logic instructions.	121
5.4	Program Counter instructions.	130
6.1	Subroutine and interrupt handling instructions.	147
6.2	The 7-segment lookup table showing byte[k] being extracted.	159
8.1	Part of Microchip's file <code>p16f84a.inc</code>	213
8.2	The listing file <code>root.lst</code>	218
8.3	The absolute 8-bit Intel format object-code file <code>root.hex</code>	219
8.4	The error file.	220
8.5	The <code>rms.lkr</code> linker command file.	224
8.6	The output linker map file <code>rms.map</code>	230
8.7	The resulting absolute object file <code>rms.hex</code> in INHEX8 format.	231
9.1	Resulting assembly-level CCS compiler output after linking.	251
10.1	Power supply operating current.	270
10.2	Oscillator operation modes.	274
10.3	Power-on reset and <code>sleep</code> instruction timeouts.	283
10.4	Reset conditions.	285
11.1	Summary of Mid-range PIC MCU parallel I/O provision.	290
11.2	Energization pattern for the eight field directions.	318
14.1	Quantization parameters.	438
14.2	ADC clocking frequency versus device crystal frequency.	455
16.1	Enhanced-range instruction set.	523

List of Programs

4.1	Incrementing a packed BCD byte.	93
5.1	Clearing a block of Files the linear way.	104
5.2	Clearing a block of Files using a repeating loop.	107
5.3	The double-precision add program.	115
5.4	Scanning the File looking for the highest 1.....	128
5.5	Double-precision decrement.....	135
5.6	Bi-quinary error detection.	136
5.7	Binary to 2-digit BCD conversion.	137
5.8	Dividing by three.....	138
5.9	Multiplication by nine.	139
5.10	Average daily temperature.	141
6.1	A 1 ms delay subroutine.....	152
6.2	A 2 ms delay subroutine.....	153
6.3	A 100 ms delay subroutine.	155
6.4	A $K \times 100$ ms delay subroutine.	156
6.5	An alternative $K \times 100$ ms delay subroutine.	157
6.6	The software 7-segment decoder.	160
6.7	The byte multiplication subroutine.	162
6.8	Implementing a byte multiply using a stack model.....	168
6.9	Coding a 208 μ s delay.	169
6.10	A 1-min delay program.	170
6.11	Binary to 3-digit BCD conversion.	171
6.12	Coding the square root subroutine.	173
6.13	Using a software stack to pass parameters and to provide a workspace.	176
6.14	The software 7-segment decoder revisited.	176
6.15	A 30 s delay subroutine.	179
7.1	People counting.	189
7.2	Program for the pea-canning packer.	198
7.3	Foreground ISR for oven safety.....	201
7.4	Coding the real-time clock ISR.....	203
7.5	Incrementing a packed-BCD byte with maximum value of 99.	204
7.6	Interrupt handler for the vending machine.	206
8.1	Absolute assembly-level code for our square-root module....	212

XVIII List of Programs

8.2	The main relocatable source file <code>main.asm</code>	226
8.3	The relocatable source file <code>sqr.asm</code>	228
8.4	The relocatable source file <code>root.asm</code>	229
9.1	A simple function coded in C	248
9.2	Coding the square root function.	257
9.3	Linearizing a K-type thermocouple.	258
9.4	Generating the root-mean-square value of two variables.	259
9.5	A simple serial data transmitter.	260
9.6	Program in C for the pea-canning packer.	261
9.7	The electronic die.	263
11.1	Implementing a Parallel port handshake data transfer.	294
11.2	Scanning the keypad.	306
11.3	Noise filtered keypad scanning.	307
11.4	Scanning the keypad in C	308
11.5	Interacting with the intruder hardware.	313
11.6	A digital comparator with hysteresis.	316
11.7	Driving a stepper motor.	319
11.8	Debouncing the C keypad device driver.	320
11.9	Displaying the decimal equivalent of a binary byte.	323
11.10	Displaying a 3-digit decimal number on a scanning readout.	324
12.1	Displaying the decimal equivalent of a byte using in serial.	330
12.2	A C implementation of the <code>SPI_WRITE</code> subroutine.	331
12.3	Input serial byte subroutine.	335
12.4	A C implementation of a SPI input read.	335
12.5	Interacting with the MAX549A dual-channel SPI DAC.	338
12.6	Using the SSP for SPI data input and output.	346
12.7	Interfacing to the MAX549A in C	349
12.8	A crystal frequency-independent short delay macro.	357
12.9	Low-level I ² C subroutines.	358
12.10	Interacting with the MAX518 dual-channel I ² C DAC.	360
12.11	The I ² C temperature acquisition ISR.	368
12.12	The I ² C temperature acquisition handler subroutine.	369
12.13	Interfacing to the MAX518 in C	370
12.14	Asynchronous formatted input and output subroutines.	374
12.15	The USART-based I/O subroutines.	381
12.16	Using a duplex asynchronous channel in C	382
12.17	Updating Program 11.6's trip value.	387
12.18	Reading in a byte using the I ² C protocol.	388
12.19	Incrementing the non-volatile odometer count.	391
12.20	Reading and writing on a 1-Wire system.	395
13.1	The bean counter Interrupt Service Routine.	407
13.2	Measuring the ECG waveform period to a resolution of 1 ms.	409
13.3	Generating a 15 minute data logger timebase.	414
13.4	Capturing the instant of time an ECG R-point occurs.	417

13.5	Pulse-Width Modulation using Timer 0.....	426
13.6	Tachometer software.....	429
13.7	Measuring the duration of a pulse.	431
14.1	Scanning an 8-channel data acquisition system.	461
14.2	Scanning an 8-channel data acquisition system.	463
14.3	A digital/analog comparator with hysteresis.....	465
14.4	ECG peak picking.	474
14.5	An implementation of the ECG peak picker in C.	475
14.6	Gauging the defibrillator discharge power.	479
14.7	Sleep conversion in C.	480
15.1	Retrieving a byte from the EEPROM Data module.	486
15.2	Putting a byte into the EEPROM Data module.	488
15.3	Incrementing the non-volatile odometer count.....	489
15.4	Reading a word from the Flash Program store.	492
15.5	Squaring an integer.	493
15.6	Writing to Flash Program memory.	495
15.7	Block writing to a the PIC16F87XA Program store.....	498
15.8	C-based coding for the odometer.	499
15.9	The Sauna Power-on reset sequence and ISR.	503
15.10	Reading a new period count.	504
15.11	Updating the Sauna EEPROM.	505
15.12	Learning the baseline.....	506
16.1	Multiplying two byte arrays.	517
16.2	A triple-precision addition.	524
16.3	Packed 2-digit BCD incrementation using the daw instruction.	525
16.4	Toggling pin RA0.	526
17.1	The timebase software.	535
17.2	The data display function.	538
17.3	The initialization code.	539
17.4	The Diagnostic process.....	541
17.5	The SST-Time process.....	542
17.6	The Main process.	546

PART I

The Fundamentals

This book is about microcontrollers (MCUs). These are digital engines modeled after the architecture of a stored-program computer and integrated onto a single very largescale integrated circuit together with support circuitry, memories and peripheral interface devices. Although the microcontroller is often confused with its better-known cousin, the microprocessor, in its role as the driving force of the ubiquitous personal computer, the vast majority of both microprocessors and microcontrollers are embedded into a variety of other digital components. The first microprocessors in the early 1970s were marketed as an alternative way of implementing digital circuitry. Here the task would be determined by a series of instructions encoded as binary code groups in read-only memory. This is more flexible than the alternative approach of wiring hardware integrated circuits in the appropriate manner. The microcontroller is simply the embodiment of this original role of the integrated computer.

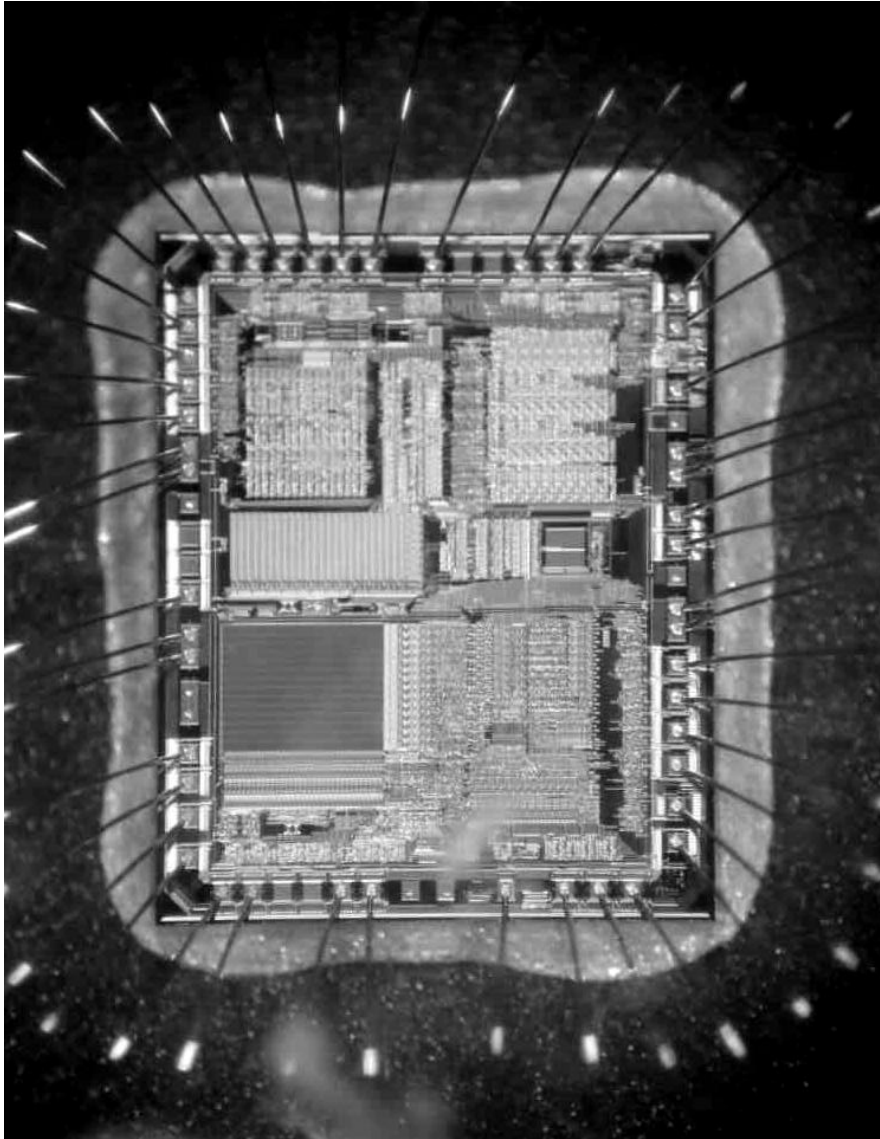
We will look at embedded microcontrollers in a general digital processing context in Parts II and III. Here our objective is to lay the foundation for this material. We will be covering:

- Digital code patterns.
- Binary arithmetic.
- Digital circuitry.
- Computer architecture and programming.

This will by no means be a comprehensive review of the subject, but there are many other excellent texts in this area¹ which will launch you into greater depths.

¹Such as S.J. Cahill's *Digital and Microprocessor Engineering*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ, 1993.

2 The Quintessential® PIC Microcontroller



Peeking into the silicon.