Designing Embedded Systems with PIC^{TM} Microcontrollers: Principles and Applications

Table of Contents

Introduction

SECTION 0 – Getting Started with Embedded Systems
This introductory chapter introduces embedded systems and the microcontroller, leading to a survey of the Microchip range of PIC^{TM} microcontrollers.

1. Tiny Computers, Hidden Control
1. The Main Idea - Embedded Systems in Today's World
1.1 What is an Embedded System?
2. Some Example Embedded Systems
2.1 The Domestic Refrigerator
2.2. A Car Door Mechanism
2.3 The Electronic "Ping-pong"
2.4. The Derbot Autonomous Guided Vehicle
3. Some Computer Essentials
3.1. Elements of a Computer
3.2. Instruction Sets – CISC and RISC
3.3. Memory Types
3.4. Organising Memory
4. Microprocessors and Microcontrollers
4.1. Microprocessors
4.2. Microcontrollers
4.3. Microcontroller Families
4.4. Microcontroller Packaging and Appearance
5. Microchip, and the PIC Microcontroller
5.1 The Background
5.2 PIC Microcontrollers Today
6. An Introduction to PIC Microcontrollers Using the 12 Series
6.1. The 12F508 Architecture
7. What Others Do - a Freescale Microcontroller

SECTION 1 – Minimum Systems, and the PIC^{TM} 16F84A
This section, of 5 chapters, introduces the main concepts of a microcontroller, using a "small" 16 Series PIC microcontroller. Emphasis is placed on understanding the core architecture, and using simple peripherals. Programming is in Assembler, as this allows the closest possible contact with the underlying harcware.

2. Introducing the PIC^{TM} 16 Series, and the 16F84A
1. The Main Idea - The PIC 16 Series Family
1.1. A Family Overview
1.2. The 16F84A
1.3. A Caution on Upgrades
2. An Architecture Overview of the 16F84A
2.1. The Status Register
3. A Review of Memory Technologies
3.1. Static RAM (SRAM)
3.2. EPROM (Erasable Programmable Read-Only Memory)
3.3. EEPROM (Electrically Erasable Programmable Read-Only Memory)
3.4. Flash
4. The 16F84A Memory

4.1. The 16F84A Program Memory
4.2. The 16F84A Data and Special Function Register Memory (“RAM”)
4.3. The Configuration Word
4.4. EEPROM Memory
5. Some Issues of Timing
5.1. Clock Oscillator and Instruction Cycle
5.2. Pipelining
6. Power-up and Reset
7. What Others Do - the Atmel AT89C2051
8. Taking Things Further – the 16F84A On-Chip Reset

3. Parallel Ports, Power Supply, and the Clock Oscillator
1. The Main Idea - Parallel Input/Output
2. The Technical Challenge of Parallel Input/Output
2.1. Building a Parallel Interface
2.2. Port Electrical Characteristics
2.3. Some Special Cases
3. Connecting to the Parallel Port
3.1 Switches
3.2. Light Emitting Diodes
3.3 Simple Resistive DC Switching
4. The PIC 16F84A Parallel Ports
4.1 The 16F84A Port B
4.2 The 16F84A Port A
4.3 Port Output Characteristics
5. The Clock Oscillator
5.1. Clock Oscillator Types
5.2. Practical Oscillator Considerations
5.3. The 16F84A Clock Oscillator
6. Power Supply
6.1. The Need for Power, and its Sources
6.2. 16F84A Operating Conditions
7. The Hardware Design of the Electronic Ping-Pong

4. Starting to Program – an Introduction to Assembler
1. The Main Idea - What Programs Do, and How We Develop Them
1.1. The Problem of Programming, and the Assembler Compromise
1.2. The Process of Writing in Assembler
1.3. The Program Development Process
2. The PIC 16 Series Instruction Set, with a Little More on the ALU
2.1 More on the PIC 16 Series ALU
2.2. The PIC 16 Series Instruction Set – an Introduction
3. Assemblers and Assembler Format
3.1. Introducing Assemblers, and the Microchip MPASM Assembler
3.2. Assembler Format
3.3. Assembler Directives
3.4. Number Representation
4. Creating Simple Programs
4.1. A Simple Data Transfer Program
5. Adopting a Development Environment
5.1. Introducing MPLAB
5.2. The Elements of MPLAB
5.3. The MPLAB File Structure
6. An Introductory MPLAB Tutorial
6.1. Creating a Project
6.2. Entering Source Code
6.3. Assembling the Project
7. An Introduction to Simulation
7.1. Getting Started
7.2. Generating Port Inputs

7.3. Viewing Microcontroller Features
7.4. Resetting and Running the Program.
8. Downloading the Program to a Microcontroller
9. What Others Do: A Brief Comparison of CISC and RISC Instruction Sets.
10. Taking Things Further: The 16 Series Instruction Set Format

5. Building Assembler Programs
1. The Main Idea – Building Structured Programs
1.1. Flow Diagrams
1.2. State Diagrams
1.3 Moving from Diagram to Code
2. Flow Control – Branching and Subroutines
2.1. Conditional Branching, and Working with Bits
2.2. Subroutines & the Stack
3. Generating Time Delays and Intervals
4. Dealing with Data
4.1 Indirect Addressing, and the File Select Register
4.2 Look-up Tables
4.3 Example Program with Delays and Look-up Table
5. Introducing Logical Instructions
6. Introducing Arithmetic Instructions, and the Carry Flag
6.1 Using Add Instructions
6.2 Using Subtract Instructions
6.3 An Arithmetic Program Example
6.4. Using Indirect Addressing to Save the Fibonacci Series
7. Taming Assembler Complexity
7.1. Include Files
7.2. Macros
7.3 MPLAB Special Instructions
8. More Use of the MPLAB Simulator
8.1. Breakpoints
8.2. Stopwatch
8.3. Trace
9. The Ping-pong Program
9.1. A Structure for the Ping-pong Program
9.2. Exploring the Ping-pong Program Code
10. Simulating the Ping-pong Program - Tutorial

6. Working with Time: Interrupts, Counters and Timers
1. The Main Idea - Interrupts –
1.1. Interrupt Structures
1.2. The 16F84A Interrupt Structure
1.3. The CPU Response to an Interrupt
2. Working with Interrupts
2.1. Programming with a Single Interrupt
2.2. Moving to Multiple Interrupts – Identifying the Source
2.3. Stopping Interrupts from Wrecking Your Program 1 - Context Saving
2.4 Stopping Interrupts from Wrecking Your Program 2 - Critical Regions and Masking
3. The Main Idea - Counters and Timers
3.1. The Digital Counter Reviewed
3.2. The Counter as Timer
3.3. The 16F84A TIMER0 Module
4. Applying the 16F84A Timer 0, with Examples Using the Electronic Ping-Pong
4.1. Object or Event Counting
4.2. Hardware-Generated Delays
5. The Watchdog Timer
6. Sleep Mode
7. What Others Do
8. Taking Things Further - Interrupt Latency

SECTION 2 – Larger Systems, and the PIC^{TM} 16F873A

This section, also of 5 chapters, focuses on developing a good understanding of microcontroller peripherals and their underlying principles. It makes use of the same microcontroller core as the earlier chapters, but applies a "large" PIC 16 Series microcontroller. All peripherals introduced are also directly applicable to the 18 Series. Emphasis is placed on understanding the peripherals, and using them in increasingly sophisticated applications. Program examples are in Assembler.

7. Larger Systems, and the PIC^{TM} 16F873A

1. The Main Idea - The PIC 16F87XA
2. The 16F873A Block Diagram and CPU
2.1. Overview of CPU and Core
2.2. Overview of Memory
2.3. Overview of Peripherals
3. 16F873A Memory, and Memory Maps
3.1. The 16F873A Program Memory
3.2. The 16F873A Data Memory, and Special Function Registers
3.3. The Configuration Word
4. "Special" Memory Operations
4.1. Accessing EEPROM and Program Memory
4.2. In-Circuit Serial Programming (ICSP)
5. The 16F873A Interrupts
5.1. The Interrupt Structure
5.2. The Interrupt Registers
5.3. Interrupt Identification and Context Saving
6. The 16F873A Oscillator, Reset, and Power Supply
6.1. The Clock Oscillator
6.2. Reset and Power Supply
7. The 16F873A Parallel Ports
7.1. The 16F873A Port A
7.2. The 16F873A Port B
7.3. The 16F873A Port C
8. Test, Commission and Diagnostic Tools
8.1. The Challenge of Testing an Embedded System
8.2. Oscilloscopes and Logic Analysers
8.3. In Circuit Emulators
8.4. On-Chip Debuggers
9. The Microchip In-Circuit Debugger (ICD 2)
10. Applying the 16F873A: the Derbot AGV
10.1. Power Supply, Oscillator and Reset
10.2. Use of the Parallel Ports
10.3. Assembling the Hardware
11. Downloading, Testing and Running a Simple Program, with ICD 2
11.1. A First Derbot Program
11.2. Applying the ICD 2
11.3 Setting the Configuration Bits within the Program
12. Taking Things Further – the 16F874A/16F877A Ports D and E

8. The Human and Physical Interface

1. The Main Idea – the Human Interface
2. From Switches to Keypads
2.1. The Keypad
2.2. Design Example: Use of Keypad in Derbot Hand Controller
3. LED Displays
3.1. LED Arrays: Seven-Segment Displays
3.2. Design Example: The Derbot Hand Controller 7-Segment Display
4. Liquid Crystal Displays
4.1. The HD44780 LCD Driver, and its Derivatives

4.2. Design Example: Use of LCD Display in Derbot Hand Controller
5. The Main Idea – Interfacing to the Physical World
6. Some Simple Sensors
6.1. The Microswitch
6.2. Light Dependent Resistors
6.3. Optical Object Sensing
6.4. The Opto-Sensor Applied as a Shaft Encoder
6.5. Ultrasonic Object Sensor
7. More on Digital Input
7.1. 16F873A Input Characteristics
7.2. Ensuring Legal Logic Levels, and Input Protection
7.3. Switch Debouncing
8. Actuators: Motors and Servos
8.1. DC and Stepper Motors
8.2. Angular Positioning: The "Servo"
9. Interfacing to Actuators
9.1. Simple DC Switching
9.2. Simple Switching on the Derbot
9.3. Reversible Switching - the H Bridge
9.4. Motor Switching on the Derbot
10. Building up the Derbot
11. Applying Sensors and Actuators – a "Blind" Navigation Derbot Program

9. Taking Timing Further
1. The Main Ideas – Taking Counting and Timing Further
2. The 16F87XA Timer 0 and Timer 1
2.1. Timer 0
2.2. Timer 1
2.3. Application of Timer 0 and Timer 1 for Derbot Odometry
2.4. Using Timer 0 and Timer 1 to Generate Repetitive Interrupts
3. The 16F87XA Timer 2, Comparator and PR2 Register
3.1. Timer 2
3.2. The PR2 Register, Comparator and Postscaler
4. The Capture/Compare/PWM Modules
4.1. A Capture/Compare/PWM Overview
4.2. Capture Mode
4.3. Compare Mode
5. Pulse Width Modulation
5.1. The Principle of PWM
5.2. Generating PWM Signals in Hardware - the 16F87XA PWM
5.3. PWM Applied in the Derbot for Motor Control
6. Generating PWM in Software
6.1. An Example of Software-Generated PWM
6.2. Further Assembler Directives, for Memory Definition and Branching
7. PWM used for Digital to Analog Conversion
7.1. An Example of PWM used for Digital to Analog Conversion
8. Frequency Measurement
8.1. The Principle of Frequency Measurement
8.2: Frequency (Speed) Measurement in the Derbot
9. Speed Control Applied to the Derbot
10. Where There is No Timer
11. Sleep Mode
12. Where Do We Go From Here?
13. Building up the Derbot

10. Starting with Serial
1. The Main Idea - Introducing Serial
2. Simple Serial Links - Synchronous Data Communication
2.1. Synchronous Basics
2.2. Implementing Synchronous Serial I/O in the Microcontroller

2.3. Microwire and SPI (Serial Peripheral Interface)
2.4. Introducing Multiple Nodes
3. The 16F87XA Master Synchronous Serial Port (MSSP) Module in SPI Mode
3.1. Port Overview
3.2. Port Configuration
3.3. Setting the Clock
3.4. Managing Data Transfer
4. A Simple SPI Example
5. The Limitations of Microwire and SPI, and of Simple Synchronous Serial Transfer
6. Enhancing Synchronous Serial, and the Inter-Integrated Circuit Bus
6.1. Main I^2C Features, and Physical Interconnection
6.2. The Pull-up Resistor
6.3. I^2C Signal Characteristics
7. The MSSP Configured for I^2C
7.1. The MSSP I^2C Registers, and their Preliminary Use
7.2. The MSSP in I^2C Slave Mode
7.3. The MSSP in I^2C Master Mode
8. I^2C Applied in the Derbot AGV
8.1: The Derbot Hand Controller as a Serial Node
8.2: The AGV as an I^2C Master
8.3: The Hand Controller as an I^2C Slave
8.4: Evaluation of the Derbot I^2C Programs
9. Evaluation of Synchronous Serial Data Communication, and an Introduction to Asynchronous
9.1: Asynchronous Principles
9.2: Synchronising Serial Data – without an Incoming Clock
10. The 16F87XA Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART)
10.1. Port Overview
10.2.The USART Asynchronous Transmitter
10.3. The USART Baud Rate Generator
10.4. The USART Asynchronous Receiver
10.5. An Asynchronous Example
10.6. Using Address Detection with the USART Receive Mode
10.7. The USART in Synchronous Mode
11. Implementing Serial without a Serial Port – "Bit-Banging"
12. Building up the Derbot

11. Data Acquisition and Manipulation
1. The Main Idea - Analog and Digital Quantities, their Acquisition and Use
2. The Data Acquisition System
2.1. The Analog to Digital Converter
2.2. Signal Conditioning – Amplification and Filtering
2.3. The Analog Multiplexer
2.4. Sample and Hold, and Acquisition Time
2.5. Timing, and Microprocessor Control
2.6. Data Acquisition in the Microcontroller Environment
3. The PIC 16F87XA ADC Module
3.1. Overview and Block Diagram
3.2. Controlling the ADC
3.3. The Analog Input Model
3.4. Calculating Acquisition Time
3.5. Repeated Conversions
3.6. Trading off Conversion Speed and Resolution
4. Applying the ADC in the Derbot Light Meter Program
4.1. Configuration of the ADC
4.2. Acquisition Time
4.3. Data Conversion
5. Some Simple Data Manipulation Techniques

5.1. Fixed and Floating Point Arithmetic
5.2. Binary to Binary Coded Decimal Conversion
5.3. Multiplication
5.4. Scaling, and the Derbot Light Meter Example
5.5. Using the Voltage Reference for Scaling
6. The Derbot Light-Seeking Program
7. The Comparator Module
7.1 Review of Comparator Action
7.2. The 16F87XA Comparators and Voltage Reference
8. Applying the Derbot Circuit for Measurement Purposes
8.1. The Electronic Tape Measure
8.2. The Light Meter
8.3. The Voltmeter
8.4. Other Measurement Systems
9. Configuring the Derbot AGV as a Light Seeking Robot

SECTION 3 – Smarter Systems, and the PIC^{TM} 18FXX2

This section moves to more sophisticated processing power, and matching programming techniques. The PIC 18 Series family is introduced, and used as the underlying hardware. The section is however mainly concerned with software. The C programming language is introduced and applied. The issues of multi-tasking and real time, essential in the embedded world, are introduced, and the Salvo Real Time Operating System is applied.

12. Smarter Systems, and the PIC 18FXX2
1. The Main Idea - The PIC 18 Series, and the 18FXX2
2. The 18F2X2 Block Diagram and Status Register
3. The 18 Series Instruction Set
4. Data Memory and Special Function Registers
4.1. The Data Memory Map
4.2. Access RAM
4.3. Indirect Addressing, and Accessing Tables in Data Memory
5. Program Memory
5.1. The Program Memory Map
5.2. The Program Counter
5.3. Upgrading from the 16 Series, and Computed **goto** Instructions
5.4. The Configuration Registers
6. The Stacks
6.1. Automatic Stack Operations
6.2. Programmer Access to the Stack
6.3. The Fast Register Stack
7. The Interrupts
7.1. An Interrupt Structure Overview
7.2. The Interrupt Sources, their Enabling and Prioritisation
7.3. Overall Interrupt Prioritisation Enabling
7.4. Global Enabling
7.5. Other Aspects of the Interrupt Logic
7.6. The Interrupt Registers
7.7. Context Saving with Interrupts
8. Power Supply and Reset
8.1. Power Supply
8.2. Power-up and Reset
9. The Oscillator Sources
9.1. LP, XT, HS and RC Oscillator Modes
9.2. EC, ECIO and RCIO Oscillator Modes
9.3. HS+PLL Oscillator Mode
9.4. Clock Source Switching
10. Introductory Programming with the 18F242
10.1. Using the MPLAB IDE for the 18 Series
10.2. The Fibonacci Program

13. The PIC 18FXX2 Peripherals
1. The Main Idea – The 18FXX2 Peripherals
2. The Parallel Ports
2.1. The 18FXX2 Port A
2.2. The 18FXX2 Port B
2.3. The 18FXX2 Port C
2.4. The Parallel Slave Port
3. The Timers
3.1. Timer 0
3.2. Timer 1
3.3. Timer 2
3.4. Timer 3
3.5. The Watchdog Timer
4. The Capture/Compare/PWM (CCP) Registers
4.1. The Control Registers
4.2. Capture Mode
4.3. Compare Mode
4.4. Pulse Width Modulation
5. The Serial Ports
5.1. The MSSP in SPI Mode
5.2. The MSSP in I^2C Mode
5.3 The USART
6. The Analog to Digital Converter (ADC)
7. Low Voltage Detect
8. Applying the 18 Series in the Derbot-18
9. The 18F2420, and the Extended Instruction Set
9.1. NanoWatt Technology
9.2. The Extended Instruction Set
9.3. Enhanced Peripherals

14. Introducing C
1. The Main Idea – Why C?
2. An Introduction to C
2.1. A Little History
2.2. A First Program
2.3. Laying out the Program - Declarations, Statements, Comments and Space
2.4. C Key Words
2.5. The C Function
2.6. Data Type and Storage
2.7. C Operators
2.8. Control of Program Flow, and the **while** Keyword
2.9. The C Preprocessor, and its Directives
2.10. Use of Libraries, and the Standard Library
3. Compiling the C Program
4. The MPLAB C18 Compiler
4.1. Specification of Radix
4.2. Arithmetic Operations
5. A C18 Tutorial
5.1. The Linker, and Linker Scripts
5.2. Linking Header and Library Files
5.3. Building the Project
5.4. Project Files
6. Simulating a C Program
7. A Second C Example - the Fibonacci Program
7.1. Program Preliminaries – more on Declaring Variables
7.2. The **do-while** Construct
7.3. Labels, and the **goto** Keyword
7.4. Simulating the Fibonacci Program
8. The MPLAB C18 Libraries

8.1. Hardware Peripheral Functions
8.2. The Software Peripheral Library
8.3. The General Software Library
8.4. The Maths Library
9. Reading Further

15. C and the Embedded Environment
1. The Main Idea – Adapting C to the Embedded Environment
2. Controlling and Branching on Bit Values
2.1. Controlling Individual Bits
2.2. The **if,** and **if-else** Conditional Branch Structures
2.3. Simulating and Running the Example Program
3. More on Functions
3.1. The Function Prototype
3.2. The Function Definition
3.3. Function Calls, and Data Passing
3.4. Library Delay Functions, and **Delay10KTCYx** ( )
4. More Branching and Looping
4.1. Using the **break** Keyword
4.2. Using the **for** Keyword
5. Using Timer and PWM Peripherals
5.1. Using the Timer Peripherals
5.2. Using PWM
5.3. The Main Example Program Loop

16. Acquiring and Using Data with C
1. The Main Idea – Using C for Data Manipulation
2. Using the 18FXX2 ADC
2.1. The Light Seek Program Structure
2.2. Use of the ADC
2.3. Further Use of **if...else**
2.4. Simulating the Light Seek Program
3. Pointers, Arrays and Strings
3.1. Pointers
3.2 Arrays
3.3. Using Pointers with Arrays
3.4. Strings
3.5. An Example Program: Using Pointers, Arrays and Strings
3.6. A Word on Evaluating the **while** Condition
3.7. Simulating the Program Example
4. Using the I^{2}C Peripheral
4.1. An Example I^{2}C Program
4.2. Use of ++ and – Operators
5. Formatting Data for Display
5.1. Overview of Example Program
5.2. Using Library Functions for Data Formatting

17. More C, and the Wider C Environment
1. The Main Idea - More C, and the Wider C Environment
2. Assembler Inserts
3. Controlling Memory Allocation
3.1. Memory Allocation Pragmas
3.2. Setting the Configuration Words
4. Interrupts
4.1. The Interrupt Service Routine
4.2. Locating and Identifying the ISR
5. Example with Interrupt on Overflow – Flashing leds on the Derbot
5.1. Using Timer 0
5.2. Using Interrupts and the ISR Action
5.3. Simulating the Flashing LEDs Program

6. Storage Classes and their Application
6.1. Storage Classes
6.2. Scope
6.3. Duration
6.4. Linkage
6.5. Working with the 18 Series Memory
6.6. Storage Class Examples
7. Start-up Code: c018i.c
7.1. The C18 Start-up Files
7.2. The **c018i.c** Structure
7.3. Simulating **c018i.c**
8. Structures, Unions and Bit-Fields
9. Processor Specific Header Files
9.1. SFR Definitions
9.2. Assembler Utilities in the Header File
10. Taking Things Further - The MPLAB Linker
10.1. What the Linker Does
10.2. The Linker Script
10.3. The .map File

18. Multi-Tasking, and the Real Time Operating System
1. The Main Idea – The Challenge of Multi-Tasking and Real Time
1.1. Multi-tasking - Tasks, Priorities, and Deadlines
1.2. So What is "Real-Time"?
2. Achieving Multi-Tasking with Sequential Programming
2.1. Evaluating the Super Loop
2.2. Time-Triggered and Event-Triggered Tasks
2.3. Using Interrupts for Prioritisation - the Foreground/Background Structure
2.4. Introducing a “Clock Tick” to Synchronise Program Activity
2.5. A General Purpose “Operating System”
2.6. The Limits of Sequential Programming when Multitasking
3. The Real Time Operating System (RTOS)
4. Scheduling, and the Scheduler
4.1. Cyclic Scheduling
4.2. Round Robin Scheduling, and Context Switching
4.3. Task States
4.4. Prioritised Pre-emptive Scheduling
4.5. Cooperative Scheduling
4.6. The Role of Interrupts in Scheduling
5. Developing Tasks
5.1. Defining Tasks
5.2. Writing Tasks and Setting Priority
6. Data and Resource Protection - the Semaphore
7. Where Do We Go from Here?

19. The Salvo Real Time Operating System
1. The Main Idea - Salvo, an Example RTOS
1.1. Basic Salvo Features
1.2. Salvo Versions and References
2. Configuring the Salvo Application
2.1. Building Salvo Applications – the Library Build
2.2. Salvo Libraries
2.3. Using Salvo with C18
3. Writing Salvo Programs
3.1. Initialising and Scheduling
3.2. Writing Salvo Tasks
4. A First Salvo Example
4.1. Program Overview, and the **main** Function
4.2. Tasks and Scheduling
4.3. Creating a Salvo/C18 Project

4.4. Setting the Configuration File
4.5. Building the Salvo Example
4.6. Simulating the Salvo Program
5. Using Interrupts, Delays and Semaphores with Salvo
5.1. A Demo Program, Using an Interrupt-Based Clock Tick
5.2. Selecting the Library and Configuration
5.3. Using Interrupts, and Establishing the Clock Tick
5.4. Using Delays
5.5. Using a Binary Semaphore
5.6. Simulating the Program
5.7. Running the Program
6. Using Salvo Messages, and Increasing RTOS Complexity
7. A Program Example with Messages
7.1. Selecting the Library and Configuration
7.2. The Task: **USnd_Task**
7.3. The Task: **Motor_Task**
7.4. The Use of Messages
7.5. The Use of Interrupts, and the ISRs
7.6. Simulating or Running the Program

This closing chapter surveys techniques of connectivity and networking, an essential field of activity in the current world of embedded systems.

20. Connectivity and Networks
1. The Main Idea – Networking and Connectivity
1.1. A Word on Protocols
2. Infrared Connectivity
2.1. IrDA and the PIC Microcontroller
3. Radio Connectivity
3.1. Bluetooth
3.2. Zigbee
3.3. Zigbee and the PIC Microcontroller
4. Controller Area Network (CAN) and Local Interconnect Network (LIN)
4.1. Controller Area Network (CAN)
4.2. CAN and the PIC Microcontroller
4.3. Local Interconnect Network (LIN)
4.4. LIN and the PIC Microcontroller
5. Embedded Systems and the Internet
5.1. Connecting to the Internet with the PIC Microcontroller
6. A Conclusion

Appendix 1: The PIC 16 Series Instruction Set
Appendix 2: The Electronic Pingpong, Circuit Diagram and Program Listing
Appendix 3: The Derbot AGV, Hardware design details
Appendix 4: Basics of AGVs
Appendix 5: The PIC 18 Series Instruction Set
Appendix 6: Essentials of C