Embedded Controller Programming 2

Section 1:

Introduction and Getting Started

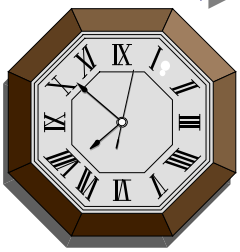
- Ken Arnold
ecp2@hte.com

Welcome!

- ECP II Course Overview
- Instructor & Student Introductions
- Review of the C language
- Basic Functions and Operators
- Introduction to the C Compiler
- Homework #1

 2

Administrative Stuff



- Fill Out Student Forms Please !
- Course Format, Policy
  - Lecture, Demo, Homework, Project
- Web Page, Discussion Group:
  http://www.hte.com/uconline/ecp
  http://webct.ucsd.edu recordings & discussion board
- Grading
  - Project, Final Exam 100 pts each
  - Extra Credit 10 pts each, Presentation, Group
- Be Here So We Can Start (and Finish) On Time !

 3

Course Objectives

- Familiarity with Tools
- Hands-on Exposure Required
- Low Level Programming, Interfacing
- Develop Embedded Applications in C
- Polite, Invisible computing!

 4

Overview - ECP 2

- Focus on Applications
- Learning to use C for Microcontrollers
- Focusing on Modular Programming
- Handling Basic Peripherals
  - Displays
  - Switches/Keypads
  - Motors/Controls
- Student Projects !

 5

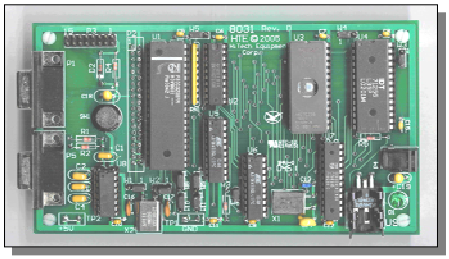
Course Format

- In-class:
  - Lecture and demonstrations
  - 3 hours * 6 meetings
  - Please Ask Questions!!
- Outside of Class:
  - Software Development Kit (SDK),
  - Development Setup:
    - SDK, Prototyping, and Test Equipment

 6

Resources

- Hardware
  - SDK, Prototyping Board
  - Component kit
- Software Tools
  - SDCC Freeware Compiler
  - Keil IDE including C Compiler
- Support Web Sites
  - C compilers:
    - http://sdcc.sourceforge.net http://http://www.opcube.com
    - http://www.keil.com
  - SDK:
    - http://www.hte.com/uconline/ecp
    - http://www.hte.com/uconline/ecp/sdkinfo.htm



 7

Resources - Hardware

- SDK
- Prototyping Board
- Components
  - Component Kit, or
  - Order from:
    - www.jameco.com
    - www.digikey.com
    - Or Fry's, Radio Shack, etc.

 8

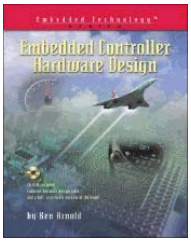
Resources - Software Tools

- SDCC Compiler (Free)
  - Command line or IDE version
  - Command line: sdcc.sourceforge.net
  - IDE: www.opcube.com
  - Unlimited code size
- Keil 8051 Commercial C Compiler
  - Keil 8051 uVision2 C Compiler/IDE ($$)
  - http://www.keil.com
  - Limited to 2KB, current eval won't run on SDK

 9

Instructor

- Father of 5
  - Preemptive Multi-tasking, Dynamic Priorities!
- HiTech Enterprises
  - Product Development and Manufacturing
- UCSD Extended Studies
  - Instructor, Program Coordinator
  - Embedded Certificate Programs



 10

Student Introduction

- Your Name and Background
- What Do You Do ?
  - *(i.e. - EE at XYZ Corp., etc..)*
- What Do You Want to Get out of This Class ?

 11

C Based 8051 Development

- Early 8051 developers used assembly.
- It was the only thing available initially!
- The thinking at the time was:
  - Assembly programming was the only choice for efficient, time-critical code.
  - Assembly programming was needed because of limited on-chip space for code and data.

 12

C Based 8051 Development

- C variants for 8051 date back to 1985
- At least a dozen companies currently make C compilers for the 8051
- Vendors Claim Code Efficiency As Good As Assembly -- don't bet on it!
- Write >>90% of Applications in C, Reserve Assembler for "Special Cases"

 13

Why use C (v Assembler)?

- Huge Amount of Existing Code!
- Assembly is hard to read and maintain
- Assembly language has it's place
  - Like a hand tool...
- C programs can be clearer, easier to read
  - Like a power tool...
- Use the right tool!

 14

Why use C? (cont'd)

- High Level Languages: More Cost-Effective
- Standardized Syntax: ANSI C Standard
  - *But* embedded compilers really are different
- C is More Portable than Assembly
- C Allows Access to Low and High Level
- There are More C Programmers!

 15

Limitations of C

- Extensions: Incompatibility with ANSI C
- Lots of Flexibility == Lots of Rope...
  - It depends on how you use it!
- Can be Cryptic: Lots of Short Hand
- Everything is a Function...
- Precedence is Not Intuitively Obvious
- Less Efficient Use of Memory and Time

 16

C Background

- First Appeared in 1970s - to Write an OS (Unix)
- Quickly became language of choice for Pro "Systems" Programmers
- December 1989 ANSI standard formally defined
- To improve efficiency, ANSI extensions added
- C has Features of High and Low Level Languages

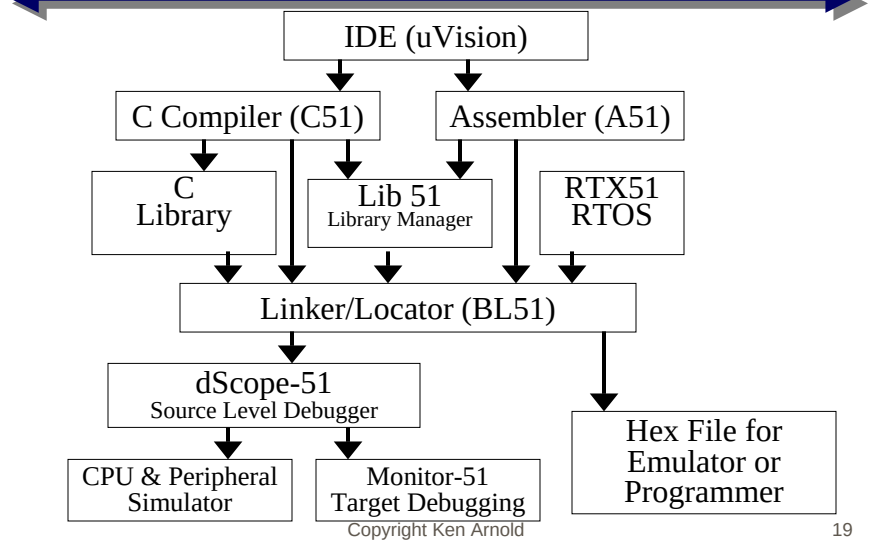
 17

C (and 8051) References

- Text: "Embedded C" by Michael Pont
  - ISBN 0 201 79523 X
- Also:
  - C and the 8051, volume I– Thomas Schultz
  - The C Programming Language – Kernighan and Richie (a.k.a. K&R C, the first book on C)
  - The Standard C Library – P.J. Plauger
  - Programming in ANSI C – Stephen G. Kochan
  - Keil & SDCC Documentation

 18

The Development Process



 19

Commenting C Programs

- Use comments to enhance your programs
  - Don't put obvious comments in just to have them there.
  - Do clarify what your doing with comments.
- The C standard for comments is:
  - /* This is a C language comment */
  - /* Your comments can extend over many
    lines in your program. It ends with */

 20

Commenting C Programs

- Recent C Compilers use the // comment:
  - // This is a valid C++ comment.
  - // It ends with the end of the line, so no */ is required
  - Many IDEs (Integrated Development Environments) Also Color-code the Comments, Reserved Words, etc.
- Common errors are:
  - Not having * and / together:
    /* This doesn't end the comment * /
  - Switching the * and / as in:
    /* This doesn't end the comment /*

 21

Common C Data Types

char*	8 bits
enum	16 bits
short*	16 bits
int*	16 bits
long*	32 bits
float	32 bits
double	64 bits

*May be signed or unsigned

NOTE:
Using signed and unsigned may pull in two versions of the library functions! If no negative numbers are involved, go with unsigned numbers.

 22

Data Types (Numeric)

- Range, Resolution, Accuracy
  - Many operations in embedded applications have very specific limits to the range and resolution.
  - Use only the range and precision you really need !
- Speed
  - The 8051 is an 8 bit processor
  - Use 8 bit variables when possible

 23

Data Types (Numeric)

- Accuracy
  - Many operations in embedded applications have very specific limits to the range and precision of number.
  - Use only the range and precision you really need !
  - 8, 16, 32 bit Integers also for Fixed Point
  - Floating Point

 24

Data Types - Speed

- The 8051 is an 8 bit processor
  - To maximize speed, use 8 bit variables
  - Integers Are Fastest
  - Floating Point is Slow, has a Huge Library
    - SDCC only -Keil eval doesn't support floating point
  - Fixed Point Arithmetic is Much Faster than Floating Point, Uses Integers. Requires some effort on the part of the programmer.

 25

Data Type Usage

- Truncation
  - Be careful with variable truncation
  - Example: Adding 2 8 bit number and assigning to a 16 bit value
  - This may result is truncation after addition then typecasting
  - Cast variables before addition
- Casting
  - Cast variables if there is any question about type of result
  - Example: `z = (unsigned int)a*b; // a,b are chars`
- Avoid using floats and doubles if possible – They work slowly and take up a lot of space.

 26

C Data Types – Arrays

- An array is a group of elements sharing a common name:

```
char  string1[20];
char  string2[] = “Enter a key when ready:”;
int   result[10][10];
char  lookup_Table[] = {31, 35, 38, 0x20};
unsigned char day[][]= {“Mon”, “Tue”, “Wed”};
```

 27

C Data Types – Arrays

- An array of chars is by nature a string, so C has no explicit string type.
- Be careful when indexing array elements as C provides no way to check for invalid indices.
- Avoid using large dimensional arrays if possible.
- The i^{th} element of an array is array[i-1].
  - Starts with element ZERO
- Arrays are used to implement lookup tables.

 28

C – Functions

- Functions provide the building blocks of a C program.
- Standard function declaration:
  `return_type func_name( parameters )`
- Functions without arguments:
  `return_type func_name( void )`
- Functions without return value:
  `void func_name( parameters )`
- Embedded: requires extensions - compiler specific

 29

C – Functions

- Compiler specific extensions:
  `return_type func_name( parameters ) [{mem_model}]`
  `reentrant interrupt-n using`
- Where:
  - return_type is the SINGLE value returned from the function
  - func_name is the name of the function
  - parameters are the arguments passed to the function
  - mem_model is small, compact, or large
  - reentrant indicates that function is recursive and reentrant
  - interrupt-n indicates the function is an ISR
  - using specifies the register bank used by the function arguments

 30

Demonstration of Compiler

- Using the C Development Environment
- Editing “Hello World” program
- Compiling program
- Downloading Hex file to SDK
- Running the program on the SDK

 31

Homework #1

- “Hello” Program in C
- Compile with Keil Tools
  - Download and Test on SDK
- Compile with SDCC Tools
  - Download and Test on SDK
- Play!
- Installation & Operation of Both Tools...

 32

Summary

- C Review
- Simple Data Types
- Defining Functions
- Revise “Hello World” to “Hello <name>” and run on the SDK
- REMEMBER: send e-mail to ecp2@hte.com

 33