

ELEC3730 Embedded Systems Lecture 1: Introduction and C Essentials

- Overview of Embedded Systems
- Essentials of C programming

Embedded System Definition and Examples

Embedded System Definition:

A device incorporating a microprocessor.

Aerospace

Navigation systems, landing control;

Automotive

Cruise Control, anti-lock braking systems;

Communications

Satellites, network routers, mobile basestation;

Industrial

Elevators, security/environment control;

Personal

Mobile handset, MP3 player, GPS, PDA;

Home

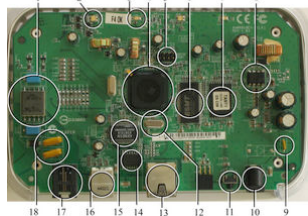
Dishwasher, home theatre components, air conditioning control;

Instrumentation

Oscilloscope, signal generators, spectrum analyser, data loggers.

DLINK Router: 16 bit ARM Processor

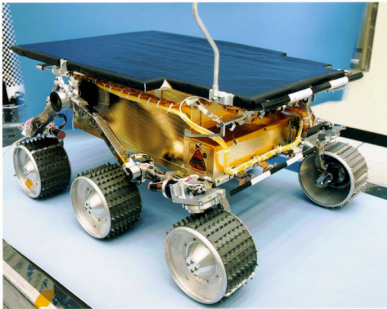
- 4. Microprocessor
- 6. RAM
- 7. Flash RAM
- 15. Ethernet Bridge



Miele Dishwasher - 8bit Motorola 68HC05



NASA Mars Sojourner: 8 bit Intel 80C85



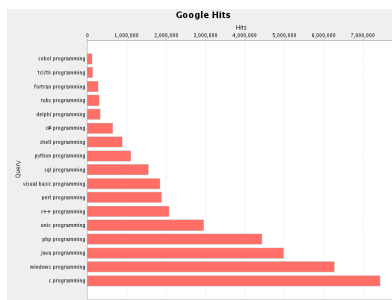
Sony Aibo Dog: 64 bit MIPS Reduced Instruction Set uP



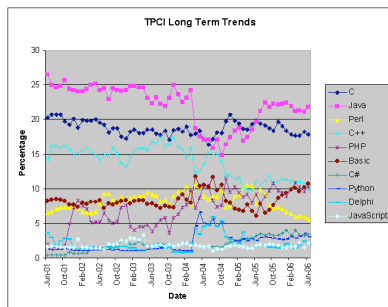
Embedded Systems - Key Aspects, Goals and Tools

- Goals
 - Reliability
 - Failure may be Life-Threatening;
 - Required service times: 24/7/365;
 - Can't reboot.
 - Low Cost
 - High Volume → Small unit savings=large total savings.
 - Low Development Time
 - Portability
 - Minimize RAM, clock speed, processor bus width → longer battery life
- Tools
 - Multitasking and Scheduling → Multiple simultaneous operations;
 - Assembly Language → Optimized I/O, efficient implementations;
 - High Level Language → Fast development cycle.
 - C is language of choice for embedded designs.

Widespread Adoption of C



Demand for programming skills (July 2006)



Syntactically - Java and C very similar

Operators same as Java:

-Arithmetic

```
• i = i+1; i++; i--; i *= 2;
```

```
• +, -, *, /, %;
```

-Relational and Logical

```
• <, >, <=, >=, ==, !=
```

```
• &&, ||, &, |, !
```

Syntax same as in Java:

```
-if ( ) { } else { }  
-while ( ) { }  
-do { } while ( );  
-for (i=1; i <= 100; i++) { }  
-switch ( ) {case 1: ... }  
-continue; break;
```

C language - Essential points

• Very simple structure - only 32 keywords

- High Acceptance;
- Compilers can be developed quickly for new platforms;
- Low-level nature → programmer has close control.
- Aggressive optimisation possible.
- Often, only hand-tuned assembly language code runs faster,
- Low-level access to computer memory via pointers possible
- Negatives
 - Need libraries to implement interesting functions
 - No error checking inherent to language → bugs!
 - Eg - arrays may go beyond bounds (buffer overflow)

Essentials of C for Embedded Programming

• Topics Covered:

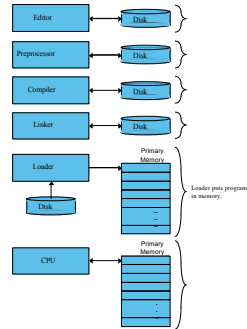
- Variables and statements
- Control and loops
- Functions
- Bitwise Operations
- Arrays and strings
- Structures
- Pointers
- The standard library
- Memory Management

• Extensions

- Mixing C and Assembly
- Accommodating Multiple Processes (Threads)
- Employing Real Time Operating System

C Program Development

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



C, the language, is just 32 keywords.

auto	do	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

A first C program

```
#include <stdio.h>

void main(void)
{
    printf("Hello World.\n "); /* print out a message */
    return;
}
```

C:\elec3730>Hello World.

Essential Points:

- #include <stdio.h> = include header file stdio.h
 - No semicolon at end
 - Small letters only - C is case-sensitive
- void main(void) { ... } is the only code executed
- /* = start of comment block, */ = end of comment block
- \n = newline, \t = tab, \a = alarm (beep), \p=backspace
- \f = formfeed, \r = carriage return, \\ = backslash, \v=vertical tab

Contents of stdio.h - A first look at header files

```
void perror(const char *);
int printf(const char * __restrict, ...);
int putc(int, FILE *);
int putchar(int);
int puts(const char *);
int remove(const char *);
int rename(const char *, const char *);
void rewind(FILE *);
int scanf(const char * __restrict, ...);
void setbuf(FILE * __restrict, char * __restrict);
int setvbuf(FILE * __restrict, char * __restrict, int, size_t);
int sprintf(char * __restrict, const char * __restrict, ...);
int sscanf(const char * __restrict, const char * __restrict, ...);
```

4 Basic Data Types

Type	Definition	Byte Size
char	Character	1
int	Integer	2 or 4
float	Real-single precision	4
double	Real-double precision	8

Data Ranges and Qualifiers

char	ASCII Characters
unsigned char	0 to 255
signed char	-128 to 127
int	-32,768 to 32,767
unsigned int	0 to 65535
signed int	As int
short int	As int
unsigned short int	As unsigned int
signed short int	As int
long int	-2,147,483,648 to 2,147,483,647
unsigned long int	0 to 4294967295
signed long int	Same as long int
float	Six digits of precision approximately
double	Twelve digits of precision approximately
long double	Twenty four digits of precision approximately

Declarations and Statements

- Example of data declarations (Memory is reserved):

```
float x;  
double d = 5;  
int *p, i, a[100];  
char s[21];
```
- Syntax:

```
type variable_name, ... [= value];
```
- Rules:
 - declarations must precede executable statements
 - int type may be modified by: long, short, unsigned
 - char and int type may be modified by unsigned
- C program is execution of one statement after another
- An expression becomes a statement when terminated with a semicolon ;
- Braces { } around a group of statements form a compound statement.
 - Syntactically equivalent to a single statement

Changing Variable Values

- Example:

```
int x, y, z;  
x = 2;  
x = x + 1;
```
- Getting Fancy

```
y = z = 4 + 5;  
x += 1;  
++x;  
x++;  
y = x--;
```
- Note:
 - assignment statements return value, which may be ignored;
 - same goes for increment statements

Pre- and Post Increment/Decrement Operators

- Operate on the value assigned to a variable either before or after all other processing.
- The difference is the location

```
ivar = 3 and inum = 5 prior to each of the following:  
  
ivalue = ivar++ + inum++;  
ivalue = 8, ivar = 4, and inum = 6 at the end;  
  
ivalue = ++ivar + inum++;  
ivalue = 9, ivar = 4, and inum = 6 at the end;  
  
ivalue = ++ivar + ++inum;  
ivalue = 10, ivar = 4, and inum = 6 at the end.
```
