TinyOS Programming II

Tips in TinyOS Programming

- Always use interfaces in wiring explicitly.
  - Optional, but make the code more clear.
- Always declare all local variables at the beginning of each command, event, or function.
  - Required by old C standard.
- Always avoid using int, /, %
  - Use uint8_t or uint16_t, signed vs. unsigned.
- Always avoid using large arrays.
  - Data memory is tiny.

Compilation Procedure

- Cross-compilation
- Source code : .nc
- After “make micaz”
- C code : app.c
- Executable on host : main.exe
- Executable on target : main.ihex (or main.srec)
- Example : Blink (lab1)

Compilation Procedure

- What if errors occur?
- Compilation error
  - error message?
- Loading error
  - error message?
- Execution error
  - error message?

API and Documentation

- "An **application programming interface** (API) is a set of functions, procedures or classes that an operating system, library or service provides to support requests made by computer programs."
- Well known APIs
  - Windows API (MSDN)
  - Java API
  - IEEE POSIX
  - Iphone API
  - OpenGL

Good APIs

- Easy to learn
  - With sufficient descriptive information
- Easy to use
  - Good user interface, organization, and navigation
- Hard to misuse
  - Clear descriptions on all use scenarios
- Easy to extend
  - Easy to add more for new and revised code

Bad APIs

- You may most likely find them in non-popular embedded systems
  - Systems used by small communities
- Open APIs may be bad APIs
  - APIs of commercial products could be bad too.
- You may have to read source code
- You may have to stop working
- Good APIs are indispensable to promote your products in the market

API Quality

- Description
  - Function: arguments, return, errors, exceptions, ...
  - Class: members, heritages, ...
- Categorization
- Code examples
- Document layout
  - JDK
  - MSDN
  - POSIX

API Examples

- Good
  - Java SE API : http://java.sun.com/javase/6/docs/
  - MSDN : http://msdn.microsoft.com/en-us/default.aspx
- Bad vs. Good
  - Linux kernel API
  - The best I could find http://www.gnugeneration.com/

API Generation

- API standardization
  - Output format : file and style
  - Content
  - Language specific : Java
- API tools
  - Public and open tools : http://en.wikipedia.org/wiki/Comparison_of_documentation_generators
  - Many are integrated in development toolkits.

Doxygen

```
/**
  * The time class represents a moment of time.
  * @author John Doe
  */
  class Time {
    public:
       /**
         * Constructor that sets the time to a
         * given value.
         * @param timemillis is a number of
         * milliseconds passed since Jan 1. 1970
         */
       Time(int timemillis) {}
       /**
         * Get the current time.
         * @return A time object set to the
         * currenttime.
         */
       static Time now() {}
     };
```

Main Page | Class List | Class Members

Time Class Reference

List of all members.

Public Member Functions

Time (int timemillis)

Static Public Member Functions

Time now ()

Detailed Description

The time class represents a moment of time.

Author:
John Doe

Constructor & Destructor Documentation

Time::Time(int *timemillis*) [inline]

Constructor that sets the time to a given value.

Parameters:
timemillis Is a number of milliseconds passed since Jan 1. 1970

Member Function Documentation

Time Time::now( ) [inline, static]

Get the current time.

Returns:
A time object set to the current time.

The documentation for this class was generated from the following file:

- test.cpp

Generated on Thu May 19 14:46:14 2005 by doxygen *1.3.8*

TinyOS API

- http://www.tinyos.net/tinyos-2.x/doc/nesdoc/micaz/
- Java style
- Description of components
- Description of interfaces
- Wiring of components
- Categories
- A few example codes

Examples

- Build APIs and documentation
  - Make micaz docs
- Structure
  - Package
  - Interface
  - Component
- Description
  - LedsC
  - MainC

TinyOS Kernel API

- TEP (TinyOS Enhancement Proposals)
  - 1. TEP Structure and Keywords
  - 2. Hardware Abstraction Architecture
  - 3. Coding Conventions (not there yet)
  - 102. Timers
    - Three timer precisions
  - 117. Low-Level I/O
    - General Purpose I/O (GPIO) pins
  - ...

TinyOS Kernel API

- Is it good?
  - It is just OK to help read the source code of TinyOS and understand how the kernel is designed.
  - It is not good to help design and add a new component into TinyOS kernel.
  - Overall, it is not as good to kernel developers as the APIs to application developers.
  - But, no truly good documentation for any kernel. (Really personal opinions.)