The TinyOS printf Library

From TinyOS Wiki

This lesson demonstrates how to use the newest version of the `printf` library located in `tos/lib/printf` to debug TinyOS applications by printing messages over the serial port.

This tutorial replaces older versions of the tutorial written for previous versions of TinyOS.

Legacy versions are listed below:

- The TinyOS printf Library (2.0.2 Release).

Contents

Overview

Anyone familiar with TinyOS knows that debugging applications has traditionally been a very arduous, if not stressful process. While simulators like TOSSIM can be used to help verify the logical correctness of a program, unforseen problems inevitably arise once that program is deployed on real hardware. Debugging such a program typically involves flashing the three available LEDs in some intricate sequence or resorting to line by line analysis of a running program through the use of a JTAG.

It is common practice when developing desktop applications to print output to the terminal screen for debugging purposes. While tools such as `gdb` provide means of stepping though a program line by line, often times developers simply want to quickly print something to the screen to verify that the value of a variable has been set correctly, or determine that some sequence of events is being run in the proper order. It would be absurd to suggest that they only be allowed three bits of information in order to do so.

The TinyOS `printf` library provides this terminal printing functionality to TinyOS applications through motes connected to a pc via their serial interface. Messages are printed by calling `printf` commands using a familiar syntax borrowed from the C programming language. In order to use this functionality, developers simply need to include a single component in their top level configuration file (`PrintfC`), and include a `"printf.h"` header file in any components that actually call `printf()`.

Currently, the `printf` library is only supported on msp430 and atmega128x based platforms (e.g. mica2, micaZ, telos, eyesIFX). In the future we hope to add support for other platforms as well.

The TinyOS `printf` Library

This section provides a basic overview of the TinyOS `printf` library, including the components that make it up and the interfaces they provide. In the following section we walk you through the process of actually using these components to print messages from a mote to your pc. If you dont care how `printf` works and only want to know how to use it, feel free to skip ahead to the next section.

The entire `printf` library consists of only 4 files located in the `tos/lib/printf/2_0_2` directory: one module, one configuration, one interface file, and one header file.

- **MainC.nc** -- A shadowed version of the MainC system component that wires in and automatically powers up the printf subsystem
- **PrintfC.nc** -- Configuration file providing printf functionality to TinyOS applications
- **PrintfP.nc** -- Module implementing the printf functionality
- **printf.h** -- Header file specifying the printf message format and size of the flush buffer

Printf functionality can be added to any application by simply #including the 'printf.h' header file. After this header file has been included, `printf` commands can be invoked by calling `printf()`, and flush commands can be invoked by calling `printfflush`.

Below is a graph of the `MainC` and `PrintfC` configurations that make this all possible. The `MainC` component redirects the `Boot.booted()` event through the `PrintfC` component so that it can power up the serial port and get the printf buffer initialized for printing:

Error creating thumbnail: convert: no decode delegate for this image format `/tmp/magick-H1jqKwh1' @ error/constitute.c/ReadImage/532.
convert: missing an image filename `/tmp/transform_2b5ce11239b8-1.png' @ error/convert.c/ConvertImageCommand/3011.

Error creating thumbnail: convert: no decode delegate for this image format `/tmp/magick-oJBUJf11' @ error/constitute.c/ReadImage/532.
convert: missing an image filename `/tmp/transform_1bca658b355c-1.png' @ error/convert.c/ConvertImageCommand/3011.

Figure 1: The component graph of the MaincC and PrintfC configurations.

Conceptually, the operation of the TinyOS `printf` library is very simple. Developers supply strings to `printf()` commands in a distributed fashion throughout any of the components that make up a complete TinyOS application. These strings are buffered in a central location inside the `PrintfP` component and flushed out to a PC in the form of TinyOS SerialMessages upon calling a `printfflush()` command. No nesC interfaces are required, and both the `printf()` and `printfflush()` commands can be called in c-fashion by simply including the 'printf.h' header file.

By encapsulating the strings produced by calls to `printf()` inside standard TinyOS SerialMessages, applications that use the serial stack for other purposes can share the use of the serial port. Alternate implementations were considered in which `printf` would have had exclusive access to the serial port. In the end, we felt it was better to give developers the freedom to decide exactly when messages should be printed, as well as allow them to send multiple types of SerialMessages in a single application.

A circular buffer is used to store the strings supplied to `printf()` before flushing them. This means that while the buffer is being flushed, calls to `printf` will still continue to succeed. The default buffer size is 250 bytes, and flushing is automatically performed whenever this buffer becomes more than half full. Explicit flushing is also possible by making calls to `printfflush()`. Most applications can get away with the automatic flushing capabilities, but explicit flushing is still recommended in applications where the sections of code under examinatation are very timing sensitive (e.g. inside the CC2420 radio stack).

Using the TinyOS `printf` Library

To help guide the process of using the `printf` library, a tutorial application has been created. Navigate to the `tinyos-2.x/apps/tutorials/Printf` directory to follow along.

The `Printf` application demonstrates everything necessary to use the `printf` library. Go ahead and open the `TestPrintfAppC` configuration to see how simple the wiring is. Notice that unlike previous versions of `printf`, no explicit wiring to printf specific components are necessary.

```
configuration TestPrintfAppC{
}
implementation {
  components MainC, TestPrintfC;
  TestPrintfC.Boot -> MainC;
}
```

All that is needed is to wire the application's `Boot` interface into `MainC`. `MainC` takes care of all the wiring necessary to bring up the Printf service and allow you to make calls to `printf()` and `printfflush`. As mentioned before, `printf()` and `printfflush()` commands can be called from any component as long as they have included the `"printf.h"` header file.

Before examining the `TestPrintfC` component, first install the application on a mote and see what kind of output it produces. Note that the instructions here are only valid for installation on a telosb mote on a linux based TinyOS distribution. For installation on other systems or for other mote platforms, please refer to lesson 1 for detailed instructions.

To install the application on the mote, run the following set of commands.

```
cd $TOSROOT\tutorials\Printf
make telosb install bsl,/dev/ttyUSBXXX
```

To see the output generated by the `Printf` tutorial application you need to start the `PrintfClient` by running the following command:

```
java net.tinyos.tools.PrintfClient -comm serial@/dev/ttyUSBXXX:telosb
```

After resetting the mote, the following output should be printed to your screen:

```
Hi I am writing to you from my TinyOS application!!
Here is a uint8: 123
Here is a uint16: 12345
Here is a uint32: 1234567890
```

Note that the 'tty' device (i.e. COM port) specified when starting the PrintfClient MUST be the one used for communicating with a mote over the serial line. On telos and mica motes this is the same port that the mote is programmed from. Other motes, such as eyesIFX, have one port dedicated to programming and another for communication. Just make sure you use the correct one.

If for some reason you do not receive the output shown above, please refer to lesson 4 to verify you have done everything necessary to allow serial communication between your pc and the mote. Remember that when using the MIB510 programming board that the switch on the very front of the board must be set to the **OFF** position in order to send messages from the mote to the pc.

Go ahead and open up `TestPrintfC` to see how this output is being generated.

Upon receiving the booted event, the `Printf` service has already been started up and initialized and we are ready to start printing messages. The following code prints the messages shown above and flushes them out over the serial line.

```
event void Boot.booted() {
      printf("Hi I am writing to you from my TinyOS application!!\n");
      printf("Here is a uint8: %u\n", dummyVar1);
      printf("Here is a uint16: %u\n", dummyVar2);
      printf("Here is a uint32: %lu\n", dummyVar3);
      printfflush();
}
```

Remember that the default printf buffer size is only 250 bytes, so don't try to print more than this at one time or you may end up cutting off the end of your string. This buffer size is configurable, however, by specifying the proper CFLAGS option in your Makefile.

```
CFLAGS += -DPRINTF_BUFFER_SIZE=XXX
```

Conclusion

A few points are worthy of note before jumping in and writing your own applications that use the functionality provided by the `printf` library.

1. In order to use the `printf` library, the `tos/lib/printf` directory must be in your include path. The easiest way to include it is by adding the following line directly within the Makefile of your top level application:

   ```
   CFLAGS += -I$(TOSDIR)/lib/printf
   ```

2. Remember that changing the `printf` buffer size is done similarly:

   ```
   CFLAGS += -DPRINTF_BUFFER_SIZE=XXX
   ```

3. You MUST be sure to #include `"printf.h"` header file in every component in which you would like to call the `printf()` command. Failure to do so will result in obscure error messages making it difficult to identify the problem.

Hopefully you now have everything you need to get going with the TinyOS `printf` library. All questions (or comments) about the use of this library should be directed to tinyos-help (mailto:tinyos-help@millennium.berkeley.edu) mailing list.

Enjoy!!

< **Previous Lesson** | **Top** | **Next Lesson** >

- This page was last modified on 30 May 2010, at 08:45.
- This page has been accessed 61,260 times.