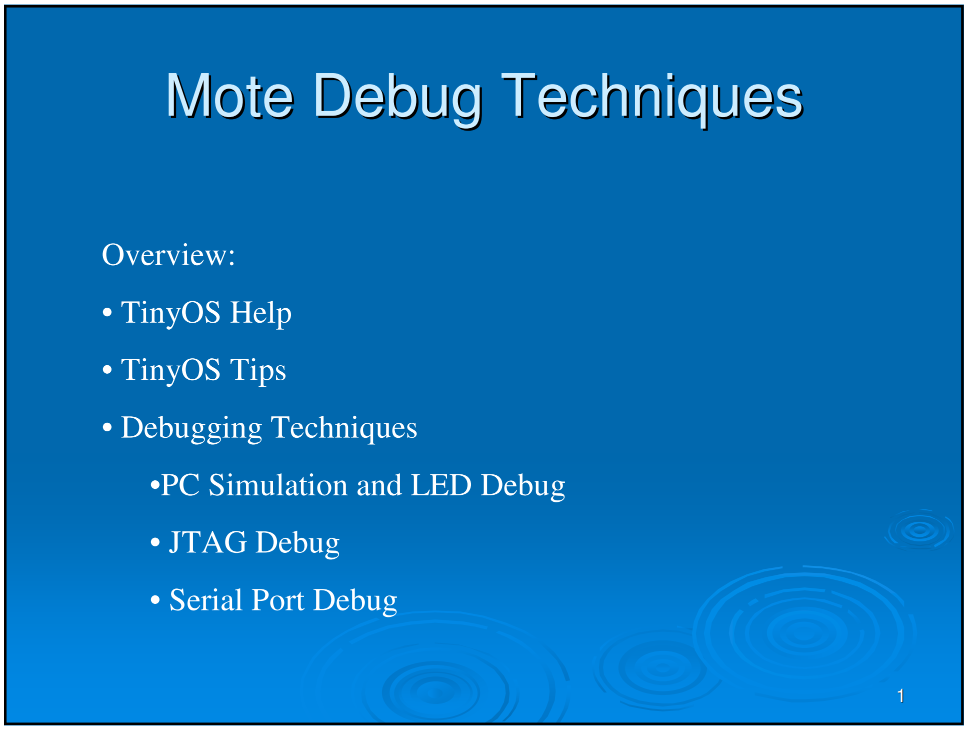
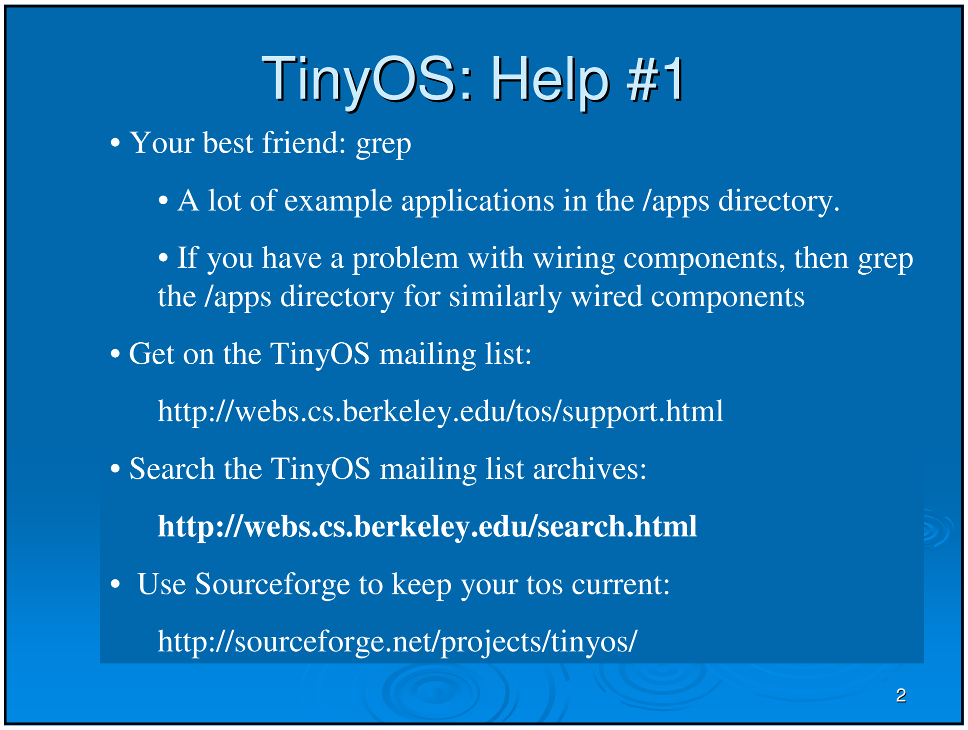


Mote Debug Techniques

Overview:

- TinyOS Help
- TinyOS Tips
- Debugging Techniques
    - PC Simulation and LED Debug
    - JTAG Debug
    - Serial Port Debug

1



TinyOS: Help #1

- Your best friend: grep
    - A lot of example applications in the /apps directory.
    - If you have a problem with wiring components, then grep the /apps directory for similarly wired components
- Get on the TinyOS mailing list:

  http://webs.cs.berkeley.edu/tos/support.html
- Search the TinyOS mailing list archives:

  http://webs.cs.berkeley.edu/search.html
- Use Sourceforge to keep your tos current:

  http://sourceforge.net/projects/tinyos/

2

TinyOS: Help #2

Use Sourceforge to update your TinyOS code and keep current:

• Before downloading; rename your present tinyos dir to keep a backup

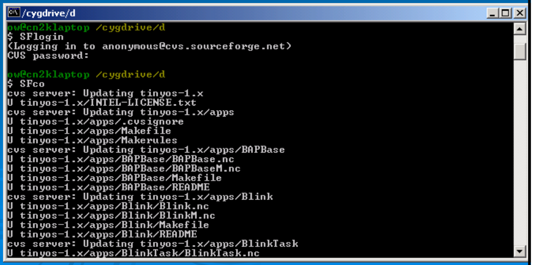
Ex (tinyos-1.x -> tinyos-1.x-rev1

• Add the following scripts to /cygwin/etc/profile:

alias SFlogin="cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/tinyos login"

alias SFco="cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/tinyos co tinyos-1.x"

• When Sourceforge asks for a password just hit return.



```
ow@cn2klaptop /cygdrive/d
$ SFlogin
(Logging in to anonymous@cvs.sourceforge.net)
CVS password:

ow@cn2klaptop /cygdrive/d
$ SFco
cvs server: Updating tinyos-1.x
U tinyos-1.x/INTEL-LICENSE.txt
cvs server: Updating tinyos-1.x/apps
U tinyos-1.x/apps/.cvsignore
U tinyos-1.x/apps/Makefile
U tinyos-1.x/apps/Makerules
cvs server: Updating tinyos-1.x/apps/BAPBase
U tinyos-1.x/apps/BAPBase/BAPBase.nc
U tinyos-1.x/apps/BAPBase/BAPBaseM.nc
U tinyos-1.x/apps/BAPBase/Makefile
U tinyos-1.x/apps/BAPBase/README
cvs server: Updating tinyos-1.x/apps/Blink
U tinyos-1.x/apps/Blink/Blink.nc
U tinyos-1.x/apps/Blink/BlinkM.nc
U tinyos-1.x/apps/Blink/Makefile
U tinyos-1.x/apps/Blink/README
cvs server: Updating tinyos-1.x/apps/BlinkTask
U tinyos-1.x/apps/BlinkTask/BlinkTask.nc
```

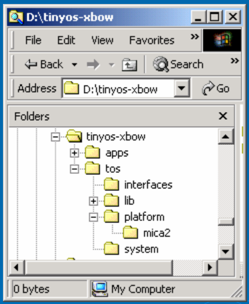
TinyOS: Help #3

• You can also do updates from Sourceforge without downloading an entire new copy of tinyos (**update –dP**). See the Sourceforge website.

•Be patient with Sourceforge. Sometimes site is too busy or upgrading. You may have to try a few times.

•Don't let Sourceforge updates or accidental deletions overwrite your application development. Develop your code in a separate directory outside of TinyOS. Ex: tinyos-xbow

- • Structure your directory like tinyos (i.e /apps, /tos,..)
- • Put modified or new tos modules in these directories, they will override the TOS modules
- • Create a Makelocal file in your /apps directory (same level as Makerules)



4

TinyOS: Help #4

- Makelocal file
- Sets the path to your code modules
- Good place to set the radio frequency.
- Makelocal example:

```
BASEDIR = ../../../tinyos-1.x/tos

LOCAL_PATH += -I../../tos/platform/mica2 -I../../tos/interfaces
PFLAGS := -tosdir=$(BASEDIR) $(LOCAL_PATH) $(LOCAL_DEFINES)
$(PFLAGS)
#CFLAGS = -DCC1K_DEFAULT_FREQ=CC1K_433_002_MHZ
CFLAGS = -DCC1K_DEFAULT_FREQ=CC1K_916MHZ
```

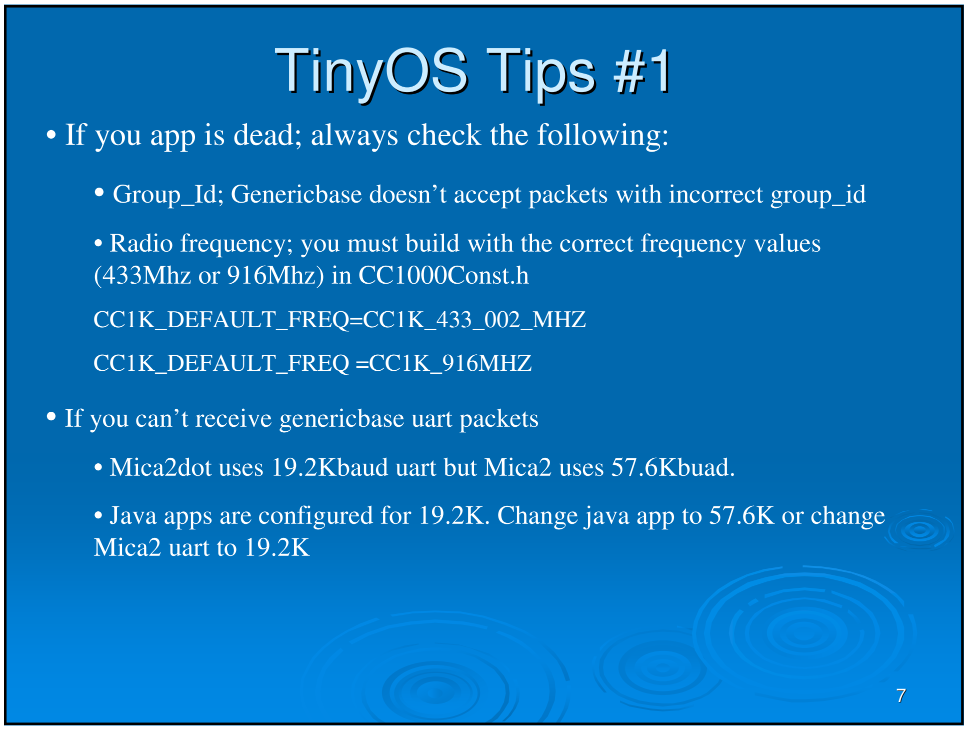
TinyOS: Help #5

- You also need to keep your Cygwin updated:

  http://www.cygwin.com/

- The Cygwin site will update your cgywin dir over the web. (I.e. use the 'update' feature.

•You do this in the lab, BAAD!!!



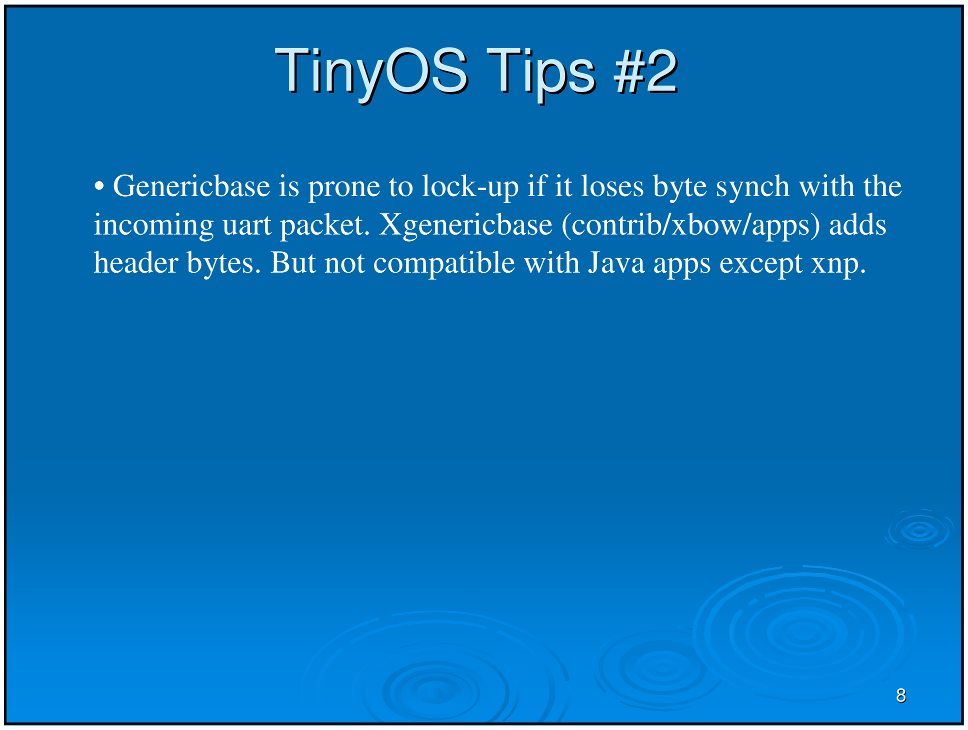
TinyOS Tips #1

- If you app is dead; always check the following:
  - Group_Id; Genericbase doesn’t accept packets with incorrect group_id
  - Radio frequency; you must build with the correct frequency values (433Mhz or 916Mhz) in CC1000Const.h

    CC1K_DEFAULT_FREQ=CC1K_433_002_MHZ

    CC1K_DEFAULT_FREQ =CC1K_916MHZ
- If you can’t receive genericbase uart packets
  - Mica2dot uses 19.2Kbaud uart but Mica2 uses 57.6Kbuad.
  - Java apps are configured for 19.2K. Change java app to 57.6K or change Mica2 uart to 19.2K

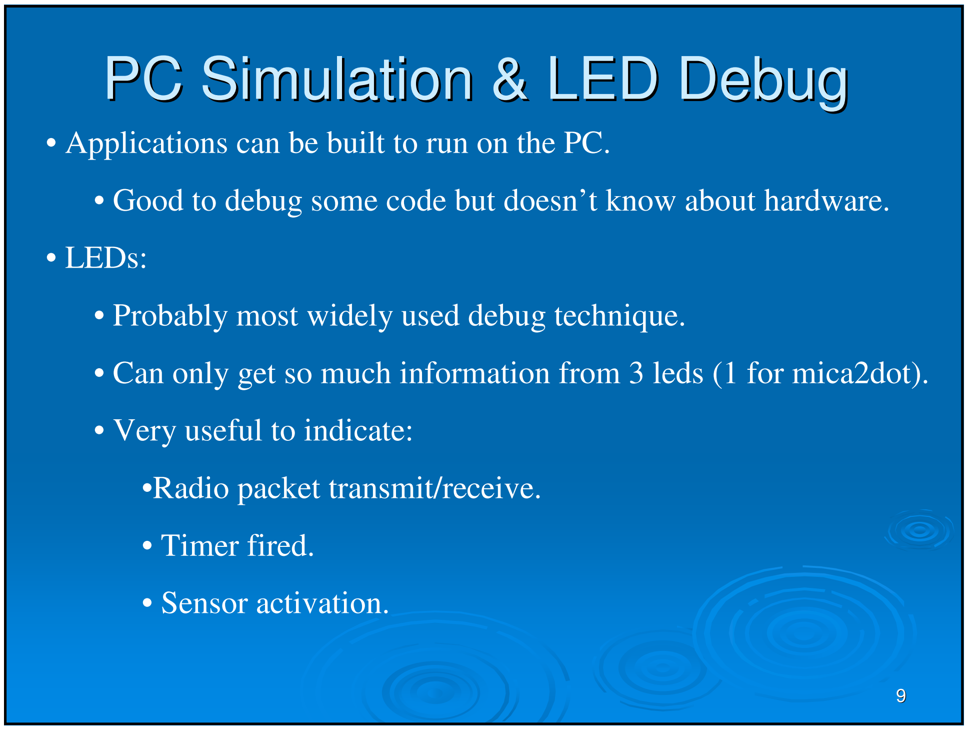
7



TinyOS Tips #2

- Genericbase is prone to lock-up if it loses byte synch with the incoming uart packet. Xgenericbase (contrib/xbow/apps) adds header bytes. But not compatible with Java apps except xnp.

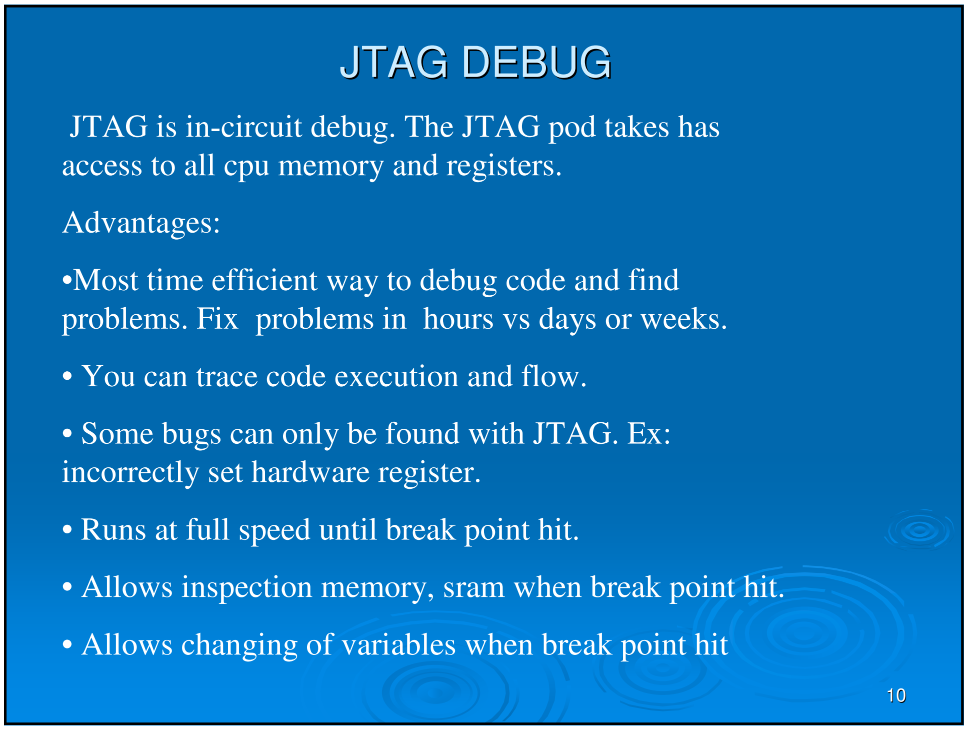
8



PC Simulation & LED Debug

- Applications can be built to run on the PC.
  - Good to debug some code but doesn’t know about hardware.
- LEDs:
  - Probably most widely used debug technique.
  - Can only get so much information from 3 leds (1 for mica2dot).
  - Very useful to indicate:
    - Radio packet transmit/receive.
    - Timer fired.
    - Sensor activation.

9



JTAG DEBUG

JTAG is in-circuit debug. The JTAG pod takes has access to all cpu memory and registers.

Advantages:

•Most time efficient way to debug code and find problems. Fix problems in hours vs days or weeks.

• You can trace code execution and flow.

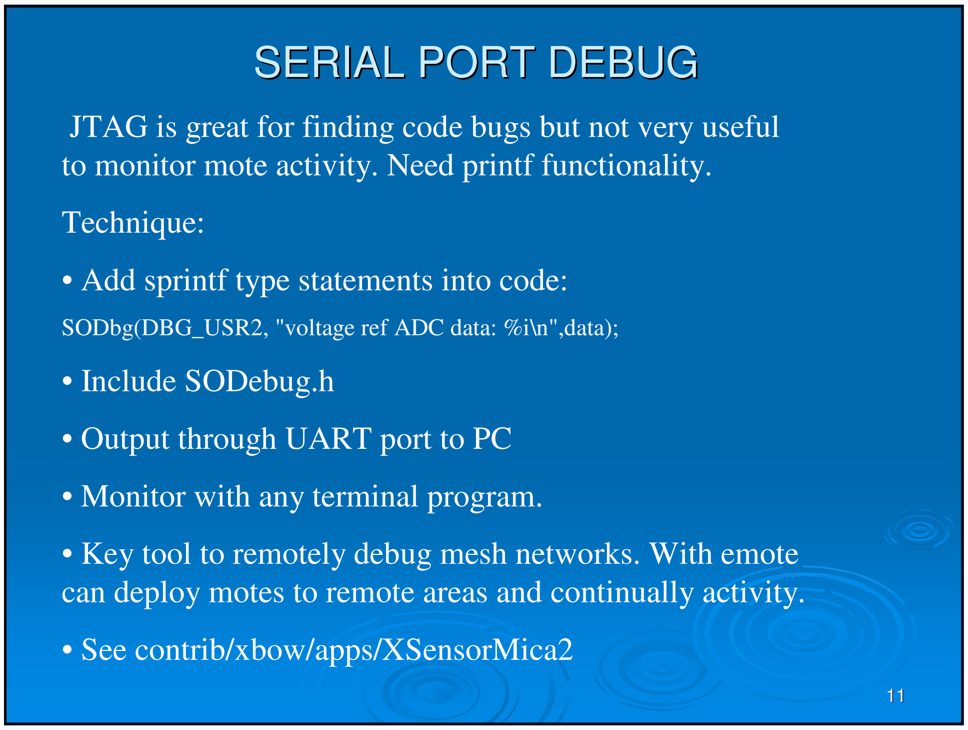
• Some bugs can only be found with JTAG. Ex: incorrectly set hardware register.

• Runs at full speed until break point hit.

• Allows inspection memory, sram when break point hit.

• Allows changing of variables when break point hit

10



SERIAL PORT DEBUG

JTAG is great for finding code bugs but not very useful to monitor mote activity. Need printf functionality.

Technique:

• Add sprintf type statements into code:

```
SODbg(DBG_USR2, "voltage ref ADC data: %i\n",data);
```

• Include SODebug.h

• Output through UART port to PC

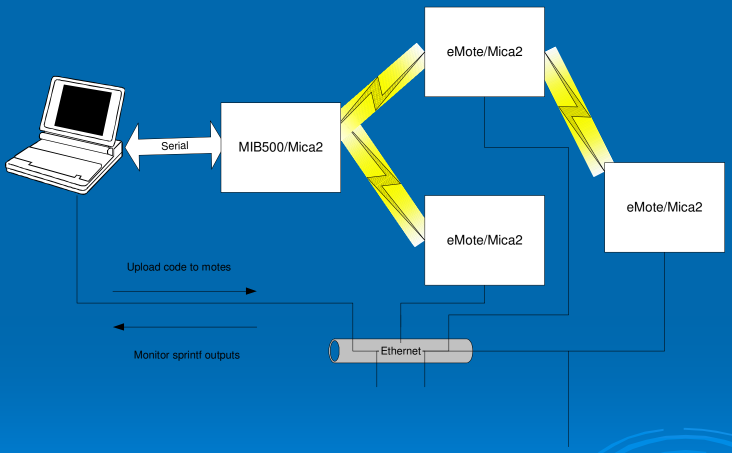
• Monitor with any terminal program.

• Key tool to remotely debug mesh networks. With emote can deploy motes to remote areas and continually activity.

• See contrib/xbow/apps/XSensorMica2

11

eMote DEBUG



12