Mote In-Network Programming User Reference

Overview

This document discusses remote, in-network programming (INP) of MICA2 and MICA2DOTs using radio communication.

The tinyos and java applications were derived from Rob Szewczyk's mica, remote programming work.

In Network Programming (INP) consists of 2 main phases

1. Program download and storage (serial flash) over the network to target mote(s).
2. In-System Programming with-in each mote with the new program.
3. Rebooting the user's application and restoring the Group_Id and Mote_Id

Program Download Phase

At the Mote end, reprogramming support is provided by the xnp module. This module performs the following:

1. Handles INP specific network commands (messages) using a reserved Active Message Handler (#47).
2. Signals the Mote Application program about INP status and request of Mote resources used in downloading.
3. Stores the downloaded Program ID in external Flash.
4. Processes incoming TOS messages containing srec-type records representing the new program.
5. Stores the s-records in the Mote's external Flash (i.e. it does not alter the Mote's current program).
6. Stores the mote_id,group_id, and prog_id in Atmega's internal eeprom at the following addresses:
   a. Mote_Id(word) = address 0xff0
   b. Group_Id(byte) = address 0xff2
   c. Prog_Id(word) = address0xff4

Led Operation

The red led is used as a visual indicator when xnp is active.

- Flashes when a code capsule is received during download.
- On after all code capsules have been received.

In-System Programming Phase

A separate "Boot Loader" that resides in a reserved section of the Mote processor's program memory provides In-System Programming. When called by either the Application or the INP service module, the Boot Loader performs the following:

1. Validates the function call by checking the passed parameters for consistency

2. Accesses External FLASH memory at location specified by caller
3. Checks that the location in FLASH contains the requested Program ID
4. Sequentially reads s-record images from the FLASH and programs (loads) the Mote's microprocessor main memory.
5. Re-boots the processor, there-by causing the new program to execute.

Reboot of User Application Phase

The downloaded srec file does not require the correct mote_id and group_id. This allows the same srec file to be loaded to multiple motes. (Typically use the "main.srec" binary file that is created in ../mica2 or ../mica2dot as the download file)

The mote's group_id, mote_id and prog_id are saved by xnp in the Atmega's eeprom before calling the bootloader. During initialization of the user's application the command NPX_SET_IDS will load the correct group_id and mote_id.

The xnp module must be "wired" into the application for INP to function. This also applies to the "downloaded" program – i.e. for INP to function *after* a successful re-programming of the Mote, the new program must also include the INP services.

TOS Requirements

In-Network Programming requires the following in addition to a standard TOS development environment:

ATMega128 Native Mode

INP resources and capabilities of the ATMega128 processor operating in native (NOT ATMega103 Compatibility mode). Refer to appropriate documents for building and programming Motes under ATMega128 native mode. Note that this requirement applies to both the Application and downloaded programs. *A TOS program compiled for a ATMega103 target will not execute on the ATMega128 running in native mode.*

Mote Resource Issues

From the Mote Application program view use of INP will impact use of the following resources

1. Radio Link
2. External FLASH memory

Radio Link

INP reserves an Active Message Handler (#47) for network downloads and INP commands. Technically, these messages are processed independently of the application. However, once download sequence has started, it is best for the application to remain idle if possible. This reduces TOS message packet collisions and/or dropped packets in the receiver.

External FLASH Memory

INP uses the external serial FLASH memory for download program storage. This resource must be released by the application before INP downloading can begin. After download the External FLASH resource is released back to, and can be used (with care) by, the application. Specifically, the application must not write or erase those sections of the External FLASH that contain the downloaded program.

The Application+INP Build

The standard TOS build operation is used. Target must be for ATMega128 native platform. Build is invoked by

“Make mica2” or “Make mica2dot”

Bootloader Srec Files

There is a separate bootloader file for the MICA2 (inpispm2.srec) and for the MICA2DOT (inpispm2d.srec). This is required because the serial flash memory is mapped to different Atmega128 pins on the MICA2 vs MICA2DOT. A bootloader file must be initially loaded into the mote, after loading the application code. Once the bootloader file is loaded it will remain in memory during all network reprograms.

Anytime the mote is reprogrammed via UISP (eg during a REINSTALL or INSTALL), the bootloader must be reloaded because UISP erases all of the target’s program memory.

The Application+INP Target Load

A special install script has been added to TOS makeinclude (“make install_INP.<moteid>). This is required to load the target platform with the application plus INPISP boot-loader. Note that the INPISP bootloader gets erased whenever UISP does an “—erase” operation on the target mote.

- **make install_inp.<moteid#> mica2**

```
@echo "    installing $(PLATFORM) binary"
@#$(SET_ID) $(MAIN_SREC) $(MAIN_SREC).out `echo $@ |sed 's:reinstall.::g'`
$(SET_ID) $(MAIN_SREC) $(MAIN_SREC).out `echo $@ |perl -pe 's/^reinstall.//; $$_=hex if /^0x/i;'`
$(PROGRAMER) $(PROGRAMMER_FLAGS) --erase
sleep 1
$(PROGRAMER) $(PROGRAMMER_FLAGS) --upload if=$(MAIN_SREC).out
sleep 1
```

```
$(PROGRAMER) $(PROGRAMMER_FLAGS) --verify if=$(MAIN_SREC).out
sleep 1
$(PROGRAMER) $(PROGRAMMER_FLAGS) --upload if=inpispm2.srec
```

or for the MICA2Dot

```
$(PROGRAMER) $(PROGRAMMER_FLAGS) --upload if=inpispm2d.srec
```

It operates as follows:

1. Creates a standard srec file from the application's binary
2. Installs the MoteID in the srec file
3. Erase the target Mote via UISP
4. Uploads the application srec file to the target Mote
5. Verify the target Mote program memory matches the application
6. Uploads the INPISP Bootloader srec (INPISPM2 or INPISPM2D.srec) file to the target Mote.

Using Java Application

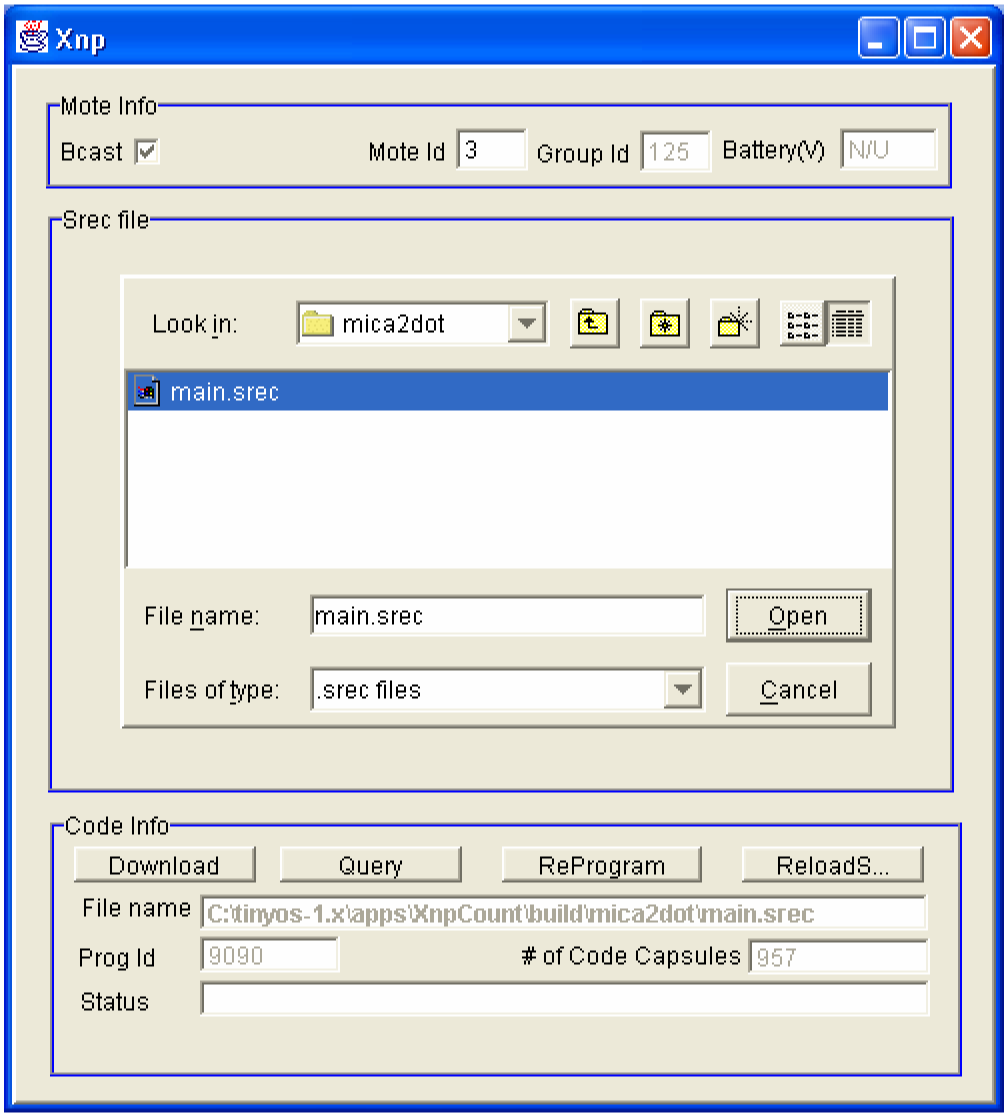
TOS Environment

- The Xnp java files should be stored in $TOSROOT/tools/java/net/tinyos/xnp.

Running the application

Xnp java application should be connected to UART when it starts. The application follows the standart UART interface introduced at TinyOS 1.1 release.

- Connect the base station mote to a serial port. TOSBase program is used for the base station mote.
- Run SerialForwarder if you want the Xnp application to connect to UART through SerialForwarder.
- Check MOTECOM environment variable.
  It MOTECOM is not defined, Xnp application is connected to sf@localhost:9001 (SerialForwarder running at hostname localhost and port 9001) by default.
- Run xnp java application.
  ```
  cd $TOSROOT/tools/java
  java net/tinyos/xnp/xnp &
  ```



Bcast ✓ Mote Id 3 Group Id 125 Battery(V) N/U

File name: main.srec

Files of type: .srec files

File name C:\tinyos-1.x\apps\XnpCount\build\mica2dot\main.srec

Prog Id 9090 # of Code Capsules 957

GUI Description

Mote Info

Bcast: If not checked then only the mote with Mote Id will be reprogrammed. If Bcast is checked then all motes with the same group id will be reprogrammed.

Mote Id: Address of a mote to reprogram if Bcast is not checked.

Group Id: The group id of the mote(s) to reprogram.

Battery: Not implemented. This is a place holder to display the battery voltage of the mote to be reprogrammed. Motes should not be reprogrammed if the battery voltage is less than 2.8 volts.

Srec file

This dialog box is used to select an srec file to be downloaded to the mote(s). The path will look for a /srec directory in the local /xnp java directory. Srec files are created in the apps/appname/build/mica2 or apps/appname/build/mica2dot directories. Warning: MICA2 and MICA2DOTs use **different** srec files. A file must be selected before downloading starts.

Code Info

Download: This button starts the download process. When code capsules are being downloaded it will display “Abort” if users wish to terminate the process.

Query: This button can be used after completion of a bcast download to check if any motes are still missing code capsules. This is automatically done during the bcast download process.

Reprogram: This button sends a command to the motes to reprogram using the downloaded code capsules.

File name: The path and file name of the srec file to download.

Prog Id: This is the CRC of the srec file being downloaded. It is used as a unique identifier.

# of Code Capsules: Program code is downloaded to the mote(s) in sections. Each section contains 16 bytes of new code. This is the number of messages (code capsules) that must be sent for the selected srec file.

Status: Display status during download.

Before downloading a file:

1) Check that the COM1 port (or the serial port you use) is the one connected to the base station.

To start downloading:

1) Navigate to the desired .srec file and open it.
2) Enter the Group Id for the target mote. The Group Id must be the same as the base station.
3) If you want to reprogram all motes with the same Group Id, check Bcast.
   When reprogramming a single mote, you can either check Bcast or uncheck Bcast and enter the Mote Id. The former is preferred because the java application queries missing packets after it sends all capsules, which is faster than querying the capsule at each time.

4) Select Download which sends code capsules to the target mote(s). The Abort button can be used to stop the downloading.
5) After the status box displays that downloading is complete then select ReProgram to command the mote to reprogram itself.

Notes:
1. In Bcast mode, Xnp will query the motes for missing code capsules after broadcasting all the capsule messages. Only motes with missing code capsules respond. If the status box displays a message that no response was received after query, then all motes should have received all code capsules.

Limitations and Know Defects