NetLinx Module Interface Specification

for a

Samsung
SIR-TS360

TABLE OF CONTENTS

Revision History

Date	Initials	Comments
10-21-04	DC	Initial release

Introduction

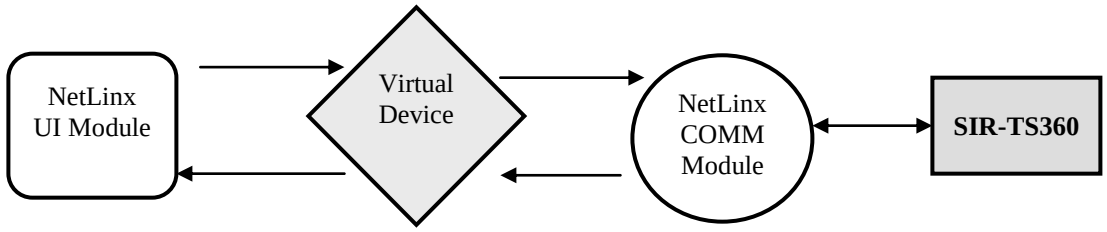
This is a reference manual to describe the interface provided between an AMX NetLinx system and a Samsung SIR-TS360 dss receiver. The required communication settings are a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and handshaking off.

Overview

The COMM module translates between the standard interface described below and the **SIR-TS360** serial protocol. It parses the buffer for responses from the dss, sends strings to control the dss, and receives commands from the UI module or telnet sessions.

A User Interface (UI) module is also provided. This module uses the standard interface described below and parses the string responses for feedback.

The following diagram gives a graphical view of the interface between the interface code and the NetLinx module.



Implementation

To interface to the SIR-TS360 module, the programmer must perform the following steps:

1. Define the device ID for the SIR-TS360 that will be controlled.
2. Define the virtual device ID that the SIR-TS360 COMM module will use to communicate with the main program and User Interface. NetLinx virtual devices start with device number 33001.
3. If a touch panel interface is desired, a touch panel file SIR-TS360,REV0.tpd and module (Samsung_SIR-TS360_UI.axs) have been created for testing.
4. The NetLinx SIR-TS360 module must be included in the program with a DEFINE_MODULE command. This command starts execution of the module and passes in the following key information: the device ID of the dss to be controlled, and the virtual device ID for communicating to the main program.

An example of how to do this is shown below.

```
DEFINE_DEVICE
   dvSIR-TS360        = 5001:1:0      // The SIR-TS360 connected to the NetLinx on 1st RS-232 port
   dvSIR-TS360 TPl    = 10001:1:0     // The touch panel used for output

   vdvSIR-TS360       = 33001:1:0     // The virtual device use for communication between the
                                      // Comm module interface and User_Interface (UI) module interface

DEFINE_VARIABLE
//Define arrays of button channels used on your own touch panel
integer nTP_BUTTONS[]={1,2,3,4,5,6,7}

DEFINE_START       // Place define_module calls to the very end of the define_start section.
// Comm module
DEFINE_MODULE 'Samsung_SIRTS360_Comm' mdlTS360_APP(dvTS360, vdvTS360)

// Touch panel module
DEFINE_MODULE 'Samsung_SIRTS360_UI' mdlTS360_APP(vdvTS360, dvTPArray, nTPButtons,
nTuneButtons)
```

Upon initialization the AMX Comm module will communicate with the Samsung SIR-TS360 and information will be exchanged.

Command Interface

The UI module controls the dss via command events (NetLinx command *send_command*) sent to the COMM module. The commands supported by the COMM module are listed below.

NOTE: AMX has not defined a standard API for DSS devices.

Command	Description
AUDIO	Selects the audio adjustment screen. AUDIO
BACK	Selects the previous channel. BACK
CCAPTION	Indicates to the unit that the closed caption button has been pressed. CCAPTION
CHANNEL=<value>	Selects a channel. <value> : + the next channel number - the previous channel number n go to channel number n CHANNEL=12
CHANNEL?	Request the current channel setting. See device notes relating to tuner queries. CHANNEL?
DEBUG=<state>	Set the debug state <state> : 0 OFF 1 ON DEBUG=1
DEBUG?	Request the debug state. DEBUG?
DOWN	Indicates to the unit that the down button has been pressed. DOWN
EXIT	Indicates to the unit that the exit button has been pressed. EXIT
FORMAT	Indicates to the unit that the format button has been pressed. FORMAT

GUIDE	Indicates to the unit that the guide button has been pressed. GUIDE
INFO	Indicates to the unit that the info button has been pressed. INFO
INPUT	Toggles the input sources. INPUT
INPUT?	Request the current input setting. INPUT?
LEFT	Indicates to the unit that the left button has been pressed. LEFT
PASSTHRU=<string>	Allows user the capability of sending commands directly to the SIR-TS360 without processing by the NetLinx module. User must be aware of the protocol implemented by the unit to use this command. This gives the user access to features which may not be directly supported by the module. See "Adding Functions to Modules" section at the end of this document for more information. The COMM module **automatically adds the required checksum to the command string.** <string> : string to send to unit PASSTHRU=RVER
POWER=<state>	Sets the power state. <state> : 1 = on this is the notation used 0 = off by the Samsung POWER=0
POWER?	Request the current power setting. POWER?
RESET	Used to flush the RS232 buffer. See device notes. A single RESET command will result in the 'No Operation' command being sent 3 times to the device.
RIGHT	Indicates to the unit that the right button has been pressed. RIGHT
SELECT	Indicates to the unit that the select button has been pressed. SELECT

UP	Indicates to the unit that the up button has been pressed. UP
VERSION?	Request the current version number of the NetLinx module. VERSION?

Table 1 – Send Command Definitions

String Feedback

The NetLinx COMM module provides feedback to the User Interface module for **SIR-TS360** changes via string events. The strings supported are listed below.

String	Description
CHANNEL=<value>	Reports the tuner channel. <value> : CHANNEL=4
INPUT=<value>	Reports the input selection. <value> : 1 Ant/Sat 2 Component 3 Svideo 4 Video INPUT=1
POWER=<value>	Reports the power status. <value> : 0 power is off 1 power is on
VERSION=<version>	Reports the current version number of the NetLinx module. <value> : current version number in xx.yy format VERSION=1.06

Table 2 - String Feedback Definitions

Device Notes

Samsung has not published the RS232 command protocol for the SIR-TS360. According to Samsung Tech Support, they may never publish the codes. The commands used in this module were derived with the help of friends, and trial and error. There is very little information returned by the device. A command is acknowledged by repeating the command. Any command that causes a channel change will generate a channel notification in the form *Chan=###-##. The three digit number is the channel code. For the Satellite/Antenna input mode, the two-digit number will be 00. For video input, the channel is 000-01. For component input 000-02 is returned. For svideo input, 000-03 is returned.

All commands to the SIR-TS360 appear to be in the form *dtv (all lowercase) followed by a byte that indicates a particular command. For example, *dtvJ places the unit in standby mode.

There is one poll command (*dtv~) that returns status information. The response from this command returns *dtv~ followed by 14 bytes of data. This module uses the first byte to decode the power status (0=on, 1=off). Byte 3 contains a number that indicates the input mode: 0=Ant/Sat, 1=component, 2=svideo, 3=video. Note that these do not correspond to the data returned in the "*Chan=" response.

There are two .tpd and one .tp4 sample screens. The .tp4 screen does not have the "INPUT" button.

There are three commands that are not implemented in this version of the module. These are the FREEZE (*dtvA), HELP (*dtvD), and GAME (*dtvB) commands. If required, these can be implemented via the PASSTHRU command.

Programming Notes

Because the SIR-TS360 responds very slowly to the serial commands, the COMM module imposes synchronous operation i.e. the COMM will not send a command to the SIR-TS360 until it has gotten a response or a timeout from the previous command.

A queue is implemented so that the commands from can be stacked (such as channel changing commands that require one command for each digit). Commands are pulled off the queue (1) when a reply is received or, (2) after the reply timer times out (200 milliseconds). The device is polled every 10 seconds to return the power state and the input selected.

The query commands return the state that is currently stored in the COMM module. They do NOT send a new query to the device. The COMM module states are updated either by the periodic poll or by the enforced feedback. When a state changes, the UI is automatically informed.

Adding Functions to Modules

Commands to the device

This module supplies a mechanism to allow additional device features to be added to software using the module. This is the PASSTHRU command, which allows protocol strings to be passed through the module. The device-specific protocol must be known in order to use this feature.

As an example, suppose that a module for a projector has not implemented the 'white balance adjustment' feature. The command that the projector protocol requires is 03H, 10H, 05H, 14H, followed by a checksum. The documentation for the PASSTHRU command specifies that the module will automatically generate the checksum. In this case, the following string should be sent from the UI code to implement 'white balance adjustment'.

```
send_command vdvDevice,"'PASSTHRU=',$03,$10,$05,$14"
```

The reason to use PASSTHRU instead of sending a protocol string directly to the device port is that the device may require command queuing, calculation of checksums, or other internal processing, which would not be done if the string was sent directly. Because of this, it is best to filter all communication TO the device through the module. (The module documentation will indicate any processing that will be automatically done to the PASSTHRU string like checksum calculation.)

Additional Feedback from the device

The module documentation indicates what feedback is provided. If additional feedback is required, a CREATE_BUFFER for the device must be implemented in the user code to process the strings from the device manually. Note that the module will still be processing the response strings independently and sending the interpreted feedback up to the user code.