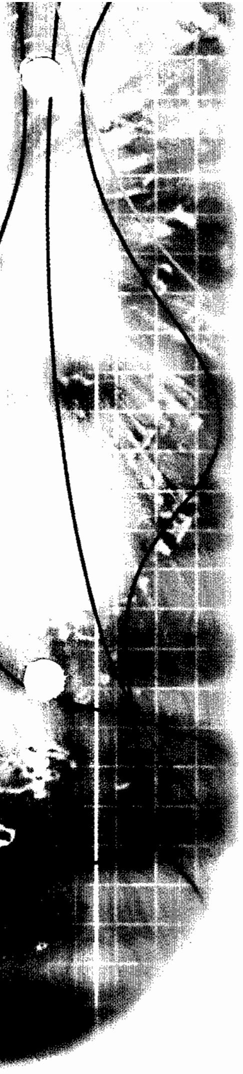


TUNICS Labview Driver

USER MANUAL

3642-IU-01-A

photonetics

CONTENTS

Chapter I - Instrument Driver Overview.

This document assumes that the user already owns the Labview software, and that he is familiar with its use. Refer to the Labview documentation for basic information.

The TUNICS Labview Driver V. 1.00 consists of two 1.44M floppy disks, and this manual. It is intended for use with **Labview V. 3.1.1** or later. It consists of **two libraries** containing VI's in their source code form. You are free to modify those VI's at your own risks. This chapter describes the installation process, what type of VI's are provided, and gives important information about their use.

I - INSTALLATION.

On the first distribution disk, you will find the installation program : **INSTALL.BAT**. To install the package on your computer, first change current path to the letter of your floppy disk drive. Then you can invoke the installation program with the following command line :

INSTALL disk_drive_letter: destination_path
example : install a: c:\vi_dir

The program will copy the following files to your hard disk :
TUNICS_1.LLB
TUNICS_2.LLB

II - VI LIBRARIES CONTENT.

The TUNICS VI's are in two libraries. The two libraries must remain in the same directory. The first library : **TUNICS_1.LLB**, contains VI's to include in your own block diagrams. Those VI's allow you to set the Power, the Current, and the Wavelength (or Frequency.) They also allow you to enable or disable the optical output, and control the Constant Power Mode (ON or OFF.) There is one VI for each parameter.

The second library : **TUNICS_2.LLB**, contains the examples and demo VI's, which are stored as top-level VI's. The examples demonstrate the use of the VI's located in the first library. They are described in the following chapters. The demo VI's demonstrate the capabilities of the driver. All other VI's in the library are sub-VI's, and you don't normally need to use them.

III - Important informations.

This driver is made to control only one Tunics at a time. You can't use it to control two Tunics, even if you use the Initialize VI to produce two different instrument handles. This is due to the use of a unique set of global variables inside the VI's, that are used by all instances of the VI's.

Don't hesitate to modify the examples provided, to build your own applications. **Basic VI's** are given, they can modify one parameter at a time. Some **« Panel » VI's** are also provided : they are made to help you managing controls on your front panel. Those panel VI's see if the values of your controls have changed between two calls, and they send the corresponding orders to Tunics. The examples illustrate the use of these two types of VI's.

Online help information is supported. You can read each VI's description by using the File / Get Info... option on each VI's front panel. You can also view it while designing your block diagrams : just open the Help / Show Help window, and maintain the mouse on the VI's icon. It also helps you wiring easily your VI's. A description of each control and indicator is provided. It can be read by popping-up on the control or indicator, and selecting the Operations / Description... sub-menu item.

Chapter II - How to build your block diagrams.

You will find here which VI's you must use in each of your block diagrams, in order to build your VI's chain. Then an example will show you the use of some basic VI's designed to set the main parameters of the Tunics. Error reporting is also discussed.

I - THE VI'S CHAIN.

Each VI has the following terminals : **Instr handle in**, **Instr handle out**, **Error in**, **Error out**. Those terminals help you build a chain to control the order in which the VI's execute. At the beginning of the chain, you put the *TUNICS Initialize* VI, which produces the Instr handle, and at the end of the chain, you put the *TUNICS Close* VI. This establishes and closes communication with your instrument.

Figure 5 shows the front panel of the *TUNICS Initialize* VI. This VI produces the Instr handle for your instrument.

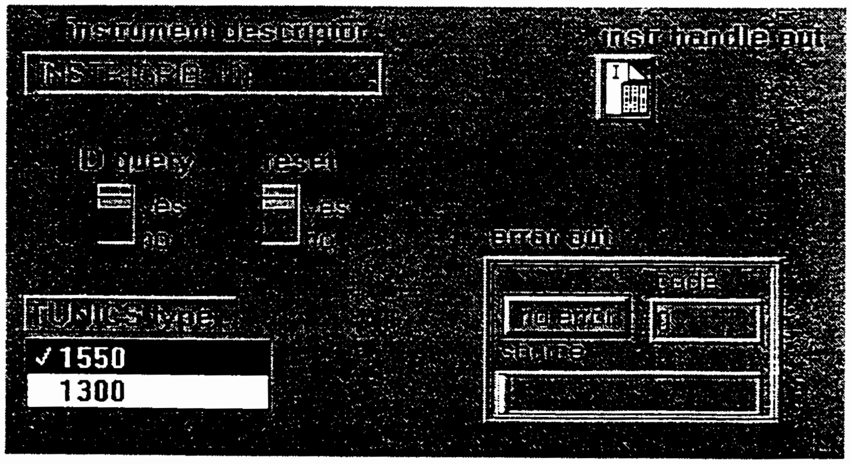


Figure 5 : The TUNICS Initialize VI front panel.

The instrument descriptor indicates the GPIB interface you use, and the TUNICS **GPIB address**, with the following syntax :

INSTR{GPIB[board]::primary address[::secondary address]}

Example, for a Tunics at GPIB address 10, and only one GPIB interface board in your system :
INSTR{GPIB::10}

The **ID Query** is performed to verify that a Tunics is really connected at the address you specified. The *TUNICS Initialize* VI performs a **Reset** that places the device in it's default state. If you disable the Reset, the other TUNICS VI's won't run properly, so it's strongly recommended not to disable it.

The **TUNICS Type** input is not linked to a terminal on the VI, because you have to choose once which type of TUNICS you use (**1550** or **1300**), then make that value default and save your VI. The default value is currently 1550. If you use TUNICS 1300, choose this value in the Menu Ring, then pop-up on the control and select : Data Operations / Make Current Value Default. You can make this change and save the VI, so that the change take effect in all your applications. If you want to do this change for only one application, do the change after having placed a copy of the VI in your block diagram.

II - How to wire your VI's.

On the block diagram of your application, the **Instr handle** propagates from one TUNICS VI to another. The **Error cluster** goes from one VI to another (not necessarily a TUNICS VI), in the order of execution needed by your application. This is illustrated in Figure 1.

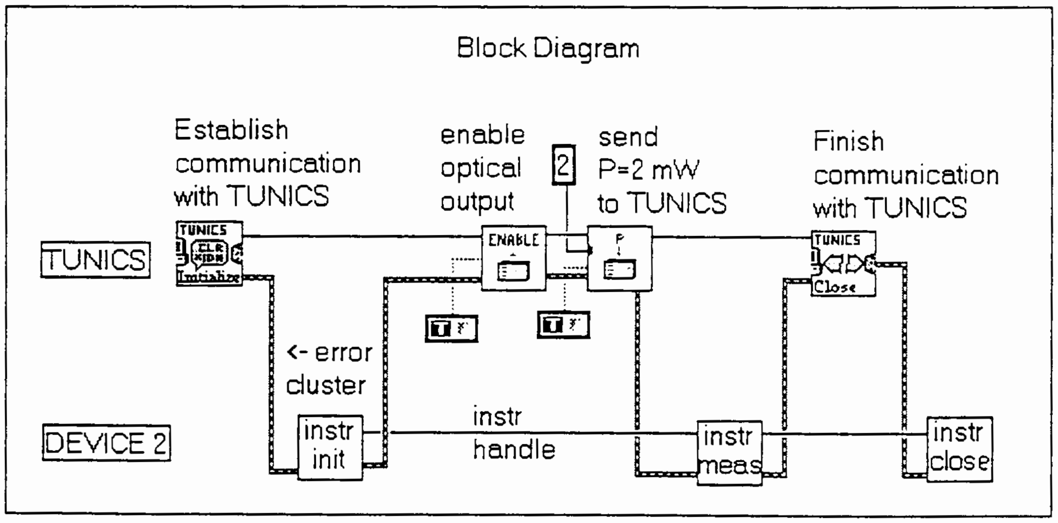
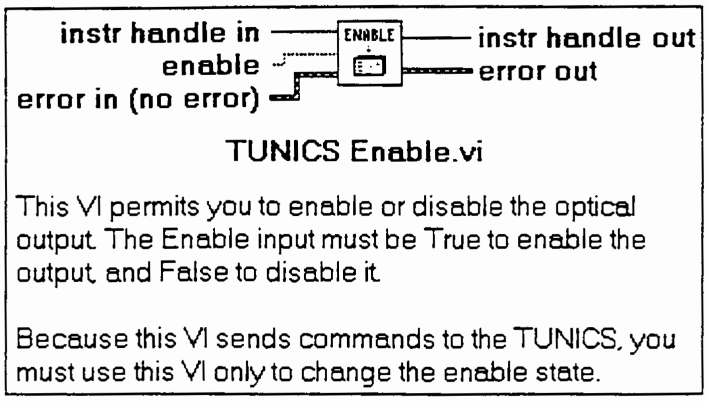


Figure 1 : Propagation of the Instr handle and the Error cluster.

Figure 1 shows the block diagram of an application where TUNICS emits power, and DEVICE 2 takes a measurement. There's one instr handle by instrument, but the error cluster propagates from one VI to another in the order of execution of the application. If an error arises in one VI, the following VI's will test the error cluster, detect the error and do nothing.

Figure 1 also demonstrates the use of two TUNICS VI's : *TUNICS Enable*.vi, and *TUNICS Set P*.vi. This diagram has been wired using the Help Window option in the Help menu. When the mouse is on the *TUNICS Enable*.vi, you can see the Help in Figure 2.


instr handle in
enable
error in (no error)
instr handle out
error out

TUNICS Enable.vi

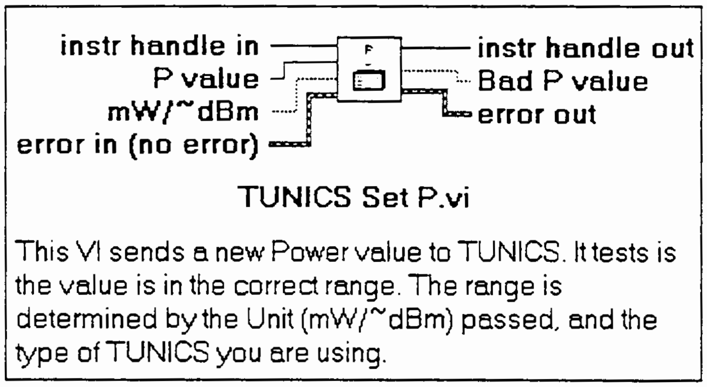
This VI permits you to enable or disable the optical output. The Enable input must be True to enable the output, and False to disable it.

Because this VI sends commands to the TUNICS, you must use this VI only to change the enable state.

Figure 2 : Help for the TUNICS Enable.vi.

The Help window displays the inputs and the outputs of the VI. Default values are indicated in parenthesis. The Enable input is gray, it's a boolean. If you wire it a True value, Tunics' optical output will be enabled; otherwise it will be disabled. In Figure 1, a constant that evaluates to True is wired to this input, in order to enable the optical output.

When the mouse is on the *TUNICS Set P*.vi, the Help window displays the content of Figure 3.


instr handle in
P value
mW/~dBm
error in (no error)
instr handle out
Bad P value
error out

TUNICS Set P.vi

This VI sends a new Power value to TUNICS. It tests is the value is in the correct range. The range is determined by the Unit (mW/~dBm) passed, and the type of TUNICS you are using.

Figure 3 : Help window for the TUNICS Set P.vi.

In Figure 3, the main inputs of the *TUNICS Set P*.vi are : **P value** (in red, a floating point value), and **mW/~dBm** (in gray, a boolean.) If you want to send a value in mW, you wire a True boolean to mW/~dBm; if you want to send a value in dBm, you wire a False boolean. The output : Bad P value, if True, signals that the value you passed was out of range, and the order wasn't sent to Tunics.

Some other basic VI's are provided to set the **Wavelength** (or Frequency) : *TUNICS Set Lambda*.vi, and set the **Current** of the source : *TUNICS Set I*.vi. These VI's behave the same way as the *TUNICS Set P*.vi. The online help information for these two VI's is shown in Figure 4.

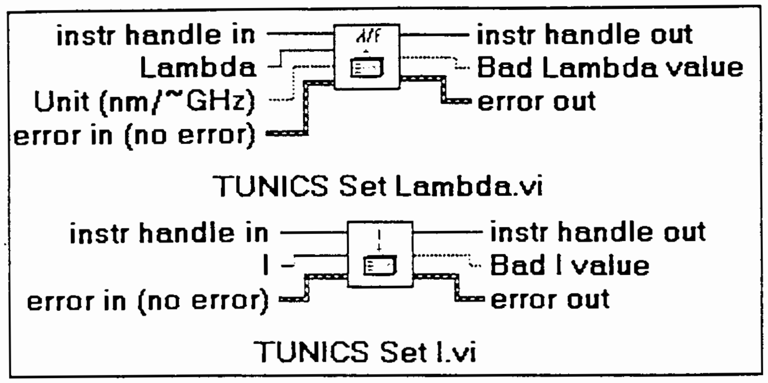


Figure 4 : Online Help for the TUNICS Set Lambda, and Set I VI's.

III - ERROR REPORTING.

Two utilities VI's are provided for **error detection** and **error display**. These are the *TUNICS Error Query* and the *TUNICS Error Message* VI's, illustrated in Figure 6.

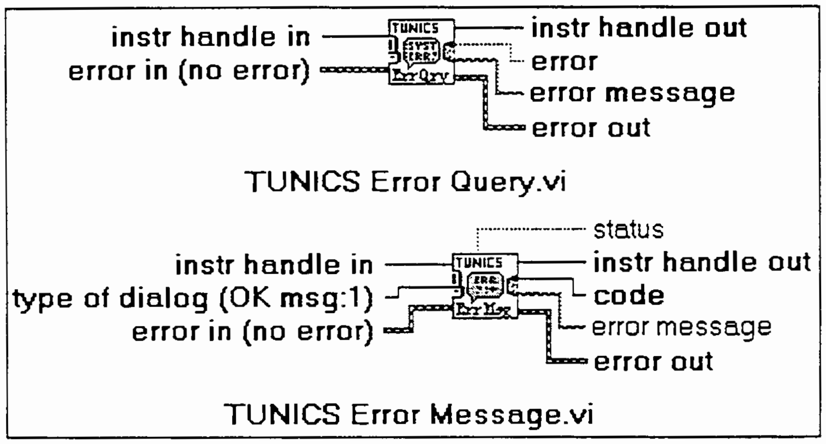


Figure 6 : The TUNICS Error Query and Error Message VI's.

The *TUNICS Error Query* detects only one error : the **LIM** (Current Limitation) error. If you consider that your application musn't stop when it occures, don't use that VI. You will prefer to use the *TUNICS Read Status* VI where the LIM output notifies that event.

If you want to be notified when an error occures in your application, use the *TUNICS Error Message* VI. This VI can display error messages which are specific to Tunics. Its default behavior is to display an OK message indicating the origin and the cause of the error. It is better to put this VI just before the Close VI in the Tunics VI's chain.

Many other VI's of different categories are provided to build your own applications. They are described in the following chapter.

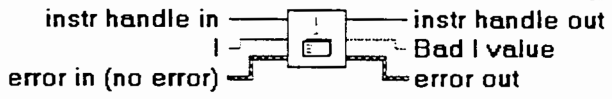
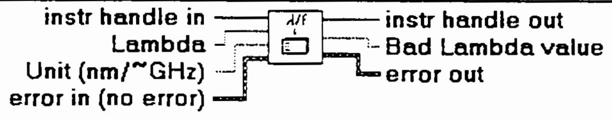
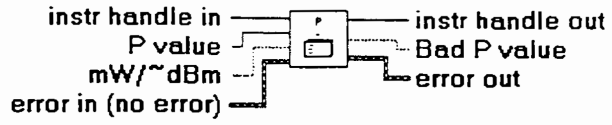
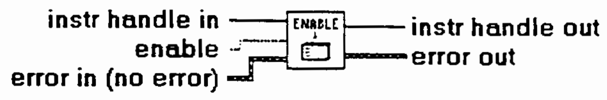
Chapter III - Description of the VI's.

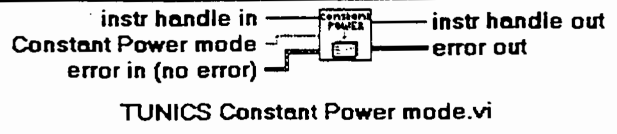
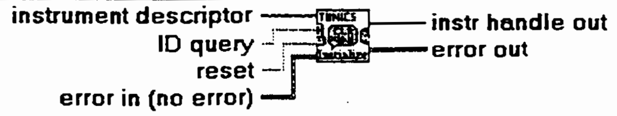
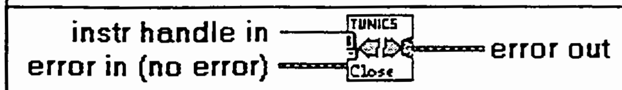
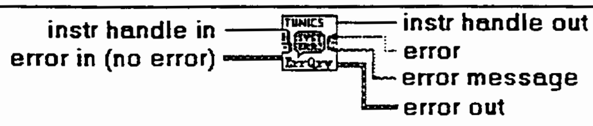
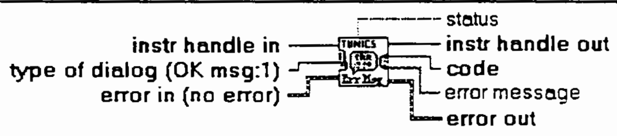
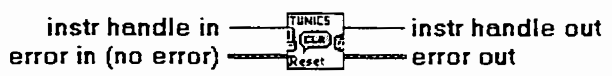
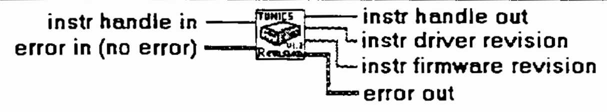
Three classes of VI's are provided. VI's for basic operations let you control the operation of Tunics, and set the parameters' values. Panel VI's help you managing your front panel controls. The other VI's are made for advanced programmatic use. Some examples, illustrating the use of those VI's, are described.

I - VI'S FOR BASIC OPERATIONS.

I - 1 - VI's Description.

Basic VI's mainly are utility and configuration VI's. Here is a list of the VI's, with their description.

VI and Terminals	Category	Description
 instr handle in I error in (no error) instr handle out Bad I value error out TUNICS Set I.vi	Configuration	**Set I :** Sends a new Current value to Tunics. The value must be in the correct range.
 instr handle in Lambda Unit (nm/~GHz) error in (no error) instr handle out Bad Lambda value error out TUNICS Set Lambda.vi	Configuration	**Set Lambda :** Sends a new Wavelength (or Frequency) value to Tunics. The value must be in the correct range, determined by the unit (True : nm=Wavelength, False : GHz=Frequency) selected. See Example 1.
 instr handle in P value mW/~dBm error in (no error) instr handle out Bad P value error out TUNICS Set P.vi	Configuration	**Set P :** Sends a new Power value to Tunics. The possible units are True : mW, and False : dBm. The value must be in the correct range, determined by the unit.
 instr handle in enable error in (no error) instr handle out error out TUNICS Enable.vi	Action	**Enable :** Enables or disables the optical output.

 instr handle in Constant Power mode error in (no error) instr handle out error out TUNICS Constant Power mode.vi	Configuration	**<u>Constant Power Mode :</u>** Sets the Constant Power Mode (apc) of Tunics. If apc is ON, Tunics follows the Power value set; if apc is OFF, Tunics follows the Current value set.
 instrument descriptor ID query reset error in (no error) instr handle out error out TUNICS Initialize.vi	Initialize	**<u>Initialize :</u>** Establishes communication with the TUNICS which GPIB address is given by the Instrument Descriptor input. The VI performs a Reset, and reads the identification message at given address to verify that a Tunics is present. Open the front panel to verify and modify the default model of Tunics selected (1550 or 1300.) See Chapter II § I for further information.
 TUNICS Close.vi	Close	**<u>Close :</u>** Closes the I/O interface with Tunics. If you use the Initialize VI, you MUST use the Close VI.
 instr handle in error in (no error) instr handle out error error message error out TUNICS Error Query.vi	Utility	**<u>Error Query :</u>** Asks Tunics for only one possible error : LIM (Current Limitation.) If you think this is not an error and your application musn't be stopped for that, don't use that VI. You will then prefer him the Read Status VI, which has a LIM (T/F) output.
 status instr handle in type of dialog (OK msg:1) error in (no error) instr handle out code error message error out TUNICS Error Message.vi	Utility	**<u>Error Message :</u>** This VI can display error messages which are specific to Tunics. Its default behavior is to display an OK message indicating the origin and the cause of the error.
 instr handle in error in (no error) instr handle out error out TUNICS Reset.vi	Utility	**<u>Reset :</u>** Sends a Reset interruption to Tunics, and waits 7 seconds for its completion. It causes Tunics to return in its default state. Reset is also performed by the Initialize VI.
 instr handle in error in (no error) instr handle out instr driver revision instr firmware revision error out TUNICS Revision Query.vi	Utility	**<u>Revision Query :</u>** Returns Tunics' firmware revision, and the TUNICS Labview Driver revision.

<u>I - 2 - Example of use : TUNICS Set Lambda.vi (Example 1.)</u>

The *TUNICS Example 1* VI performs a **Wavelength Scan** : you set the starting and ending values for the Wavelength, and the step in nanometers. The delay you wait between two values, and the Power (mW) at which you want to work, have default values that you can also change. When you click on the START button, the process begins. You can stop it before the end with the STOP button. You click on the ON/OFF button to stop the example. See Figure 7.

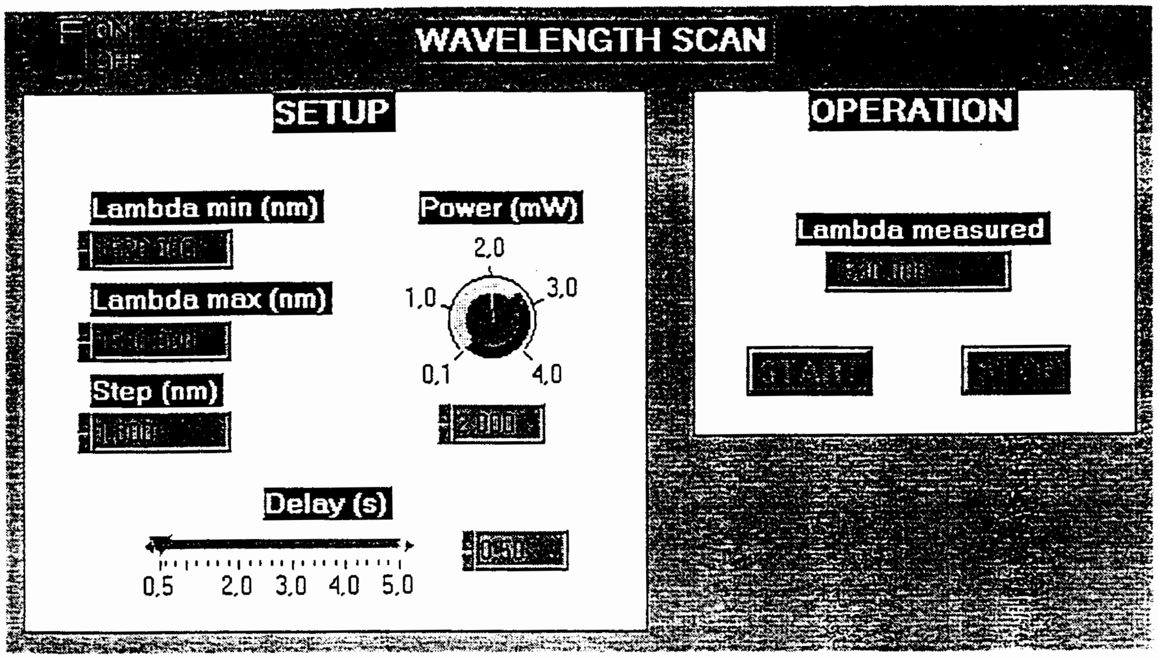


Figure 7 : Front panel of the TUNICS Example 1.vi (Wavelength Scan.)

The block diagram of this example contains a while loop that runs until the end value is reached. A simplified version of this diagram is available in Figure 8 (without error checking.)

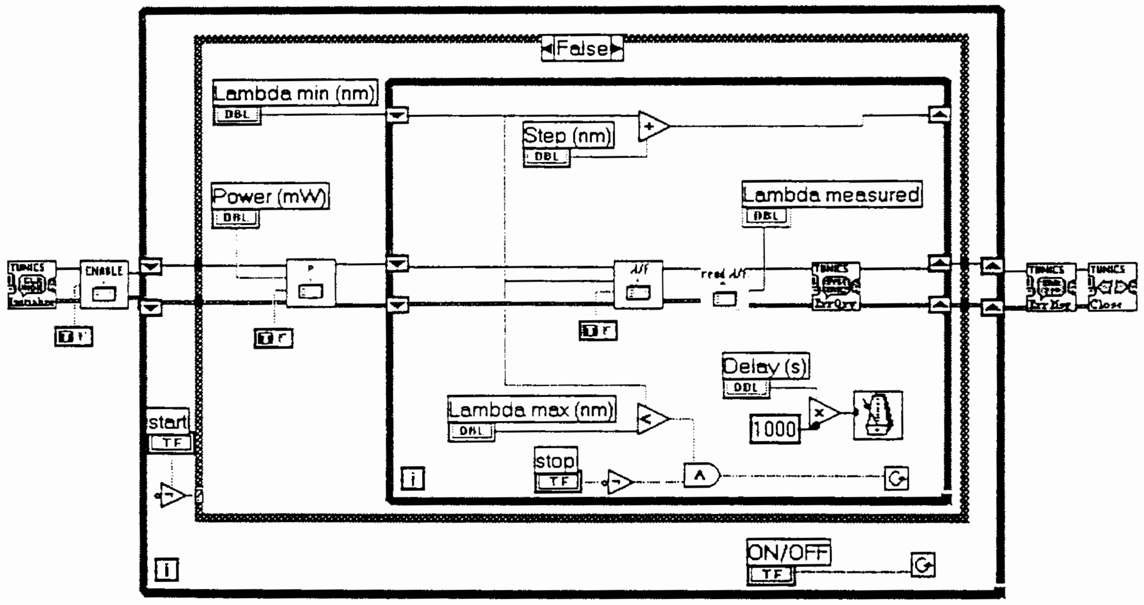


Figure 8 : Simplified Block Diagram for the TUNICS Example 1 VI.

The while loop has three shift registers : one for the wavelength, one for the instrument handle, and one for the error cluster. This permits to use the value from the preceeding execution of the loop. In this example, you can replace the *TUNICS Read Lambda* VI with your own processing.

II - The « Panel » VI's.

These VI's are designed to manage your front panel controls. Try examples two and three to understand what these VI's can do. Then try to understand how they work by reading the example description below.

II - 1 - Example of use for the TUNICS P Panel VI.

For instance, if you want to place two controls : « **Power** » and « **Unit** (mW/~dBm) », on the front panel of your own application to adjust permanently the optical Power delivered by Tunics, you just have to place the *TUNICS P panel* VI in a while loop of your own, and wire your controls to the VI. This VI will adjust Power automatically for you. The block diagram will look as the one in the *TUNICS Example 2* VI, shown in Figure 9.

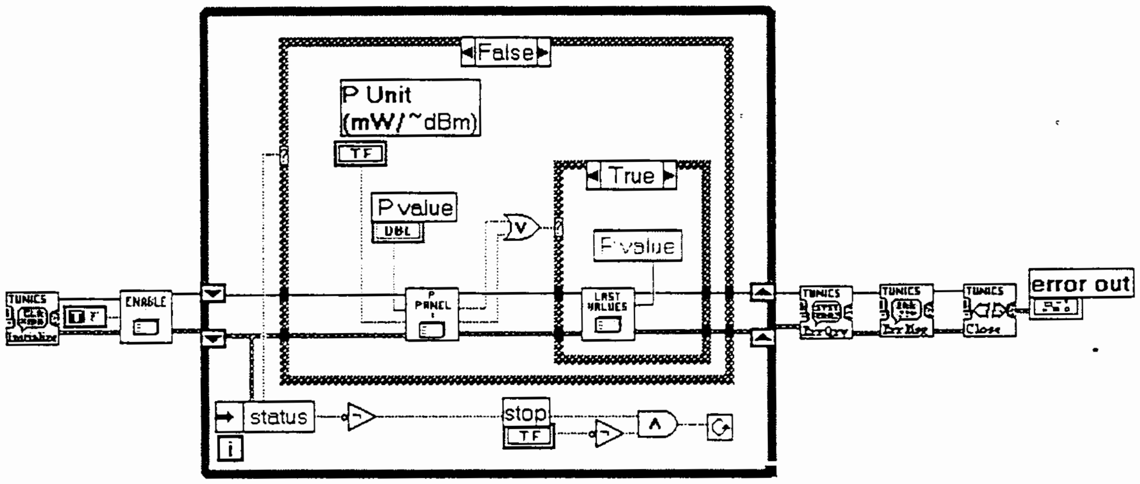


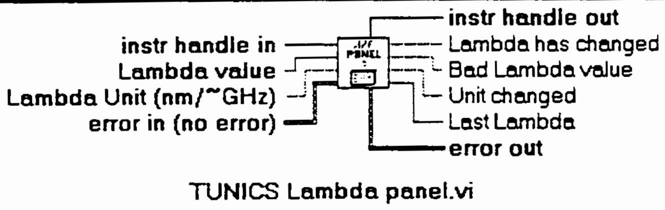
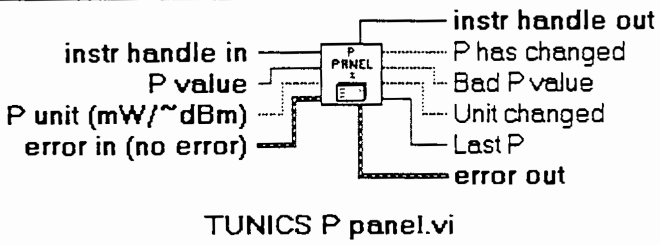
Figure 9 : Block Diagram for the TUNICS Example 2 VI.

In this figure, the «**Bad P value** » and « **Unit changed** » outputs of the *TUNICS P Panel* VI are wired to an OR gate, in order to copy the last valid Power value to the « P value » control (using the « P value » local variable). This action erases a bad value entered, and displays a converted value if you change the unit. The last valid P value is obtained using the *TUNICS Last valid values sent* VI. If you want to include this example in your own application, you just have to replace the outer while loop by the one of your application.

TUNICS Example 3 VI is the same, but it illustrates the use of the *TUNICS Lambda panel* VI.

II - 2 - « Panel » VI's reference.

Here is a list of the VI's, with their description.

VI and Terminals	Description
 instr handle in Lambda value Lambda Unit (nm/~GHz) error in (no error) instr handle out Lambda has changed Bad Lambda value Unit changed Last Lambda error out TUNICS Lambda panel.vi	**Lambda Panel :** Designed to manage two controls : « Wavelength (or Frequency) value », and « Unit (nm/GHz) ». The VI sends new valid values, and new unit to TUNICS if necessary. See Example 3 for use.
 instr handle in P value P unit (mW/~dBm) error in (no error) instr handle out P has changed Bad P value Unit changed Last P error out TUNICS P panel.vi	**P Panel :** Designed to manage two controls : « Power value », and « Unit (mW/dBm) ». The VI sends new valid values, and new unit to TUNICS if necessary. See Example 2 for use.

III - ADVANCED PROGRAMMATION.

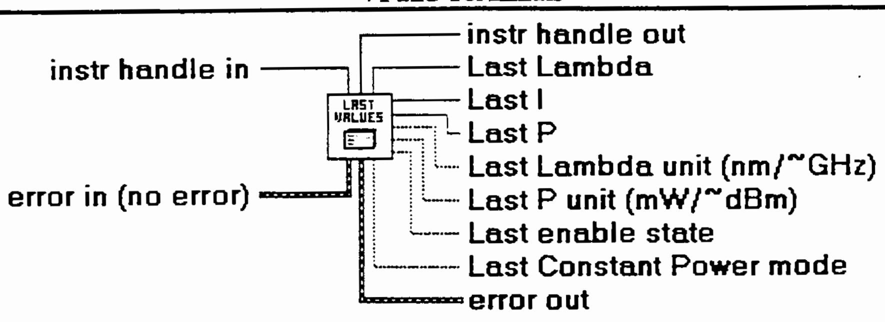
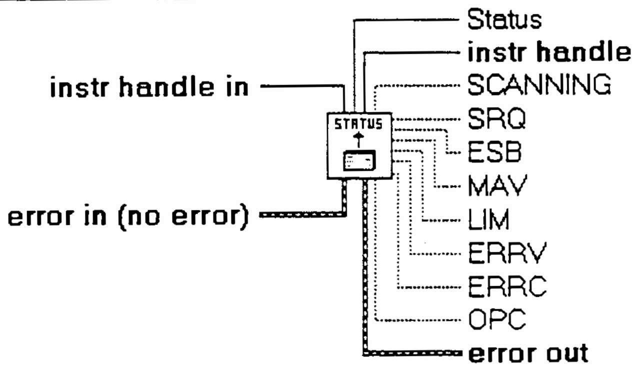
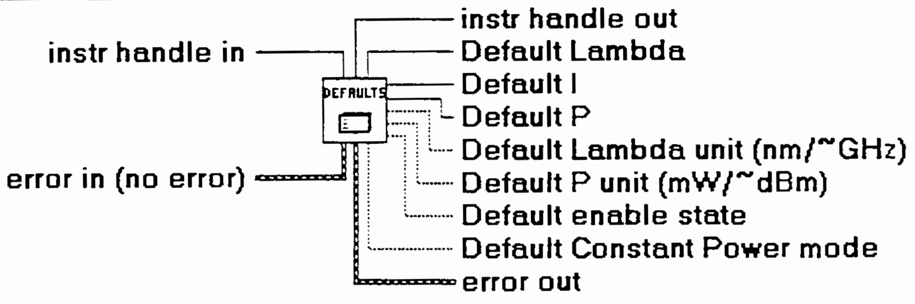
III - 1 - Controlling the status of Tunics.

Three VI's are provided to know the exact state of Tunics. The *TUNICS Last valid values sent* VI gives you the state the device is supposed to be in. Values returned are not measured : they have been memorized when the driver sent them to Tunics. It is useful to test if the value of a control has changed from last value sent.

The *TUNICS Read status* VI reads the status byte from Tunics, and separates it in eight booleans. See the **TUNICS USER'S GUIDE** for the description of the **status byte** of Tunics, or see the online help for each boolean output.

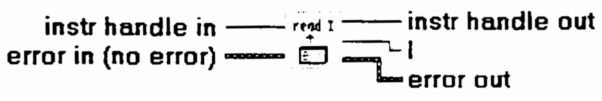
The *TUNICS Defaults* VI returns the default state of the device, after the **Reset** operation. These values are useful to get a value for initializing shift registers or controls, in using local variables. The **default state** returned depends on the type of Tunics you have choosen on the front panel of the *TUNICS Initialize* VI.

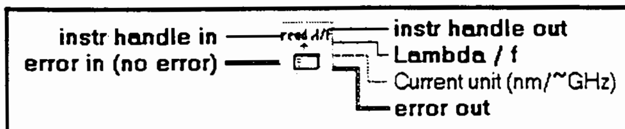
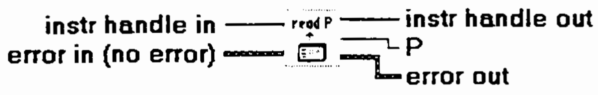
Here is a representation of the terminals of those VI's.

VI and Terminals
 instr handle in error in (no error) LAST VALUES instr handle out Last Lambda Last I Last P Last Lambda unit (nm/~GHz) Last P unit (mW/~dBm) Last enable state Last Constant Power mode error out TUNICS Last valid values sent.vi
 instr handle in error in (no error) STATUS Status instr handle out SCANNING SRQ ESB MAV LIM ERRV ERRC OPC error out TUNICS Read Status.vi
 instr handle in error in (no error) DEFAULTS instr handle out Default Lambda Default I Default P Default Lambda unit (nm/~GHz) Default P unit (mW/~dBm) Default enable state Default Constant Power mode error out TUNICS Defaults.vi

<u>III - 2 - VI's for measuring current values.</u>

Three VI's exist to measure : the **Current**, the **Power** and the **Wavelength**. The measurements taken are different from an instant to another, because they aren't absolutely precise. So we strongly recommend to use the values returned for displaying only. Here is a description of these VI's.

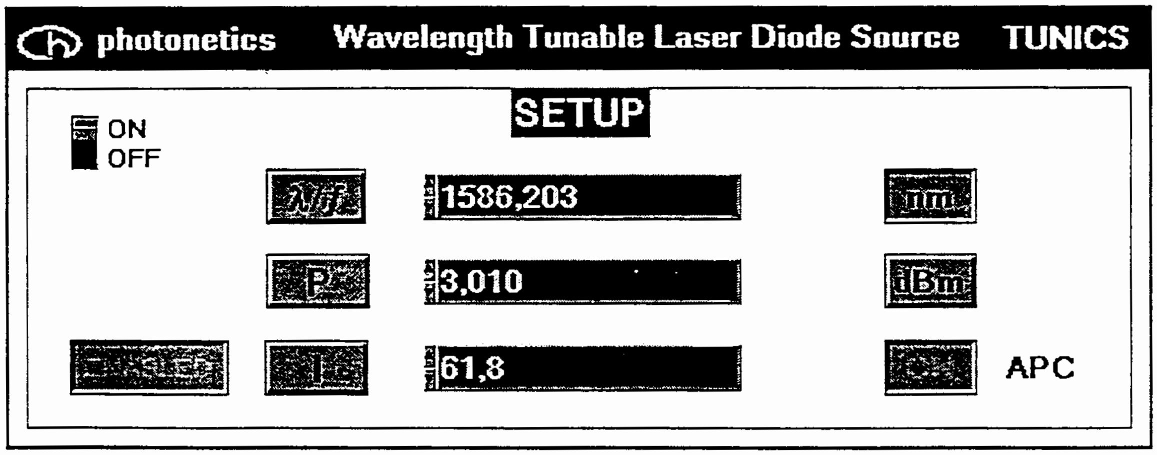
VI and Terminals	Description
 instr handle in error in (no error) read I instr handle out I error out TUNICS Read I.vi	<u>**Read I :**</u> Returns the Current measured. If the optical output is disabled, the result is 0.

 instr handle in error in (no error) read λ/f instr handle out Lambda / f Current unit (nm/~GHz) error out TUNICS Read Lambda.vi	**<u>Read Lambda :</u>** Checks the current unit (nm, or Ghz), and returns the Wavelength or Frequency value.
 instr handle in error in (no error) read P instr handle out P error out TUNICS Read P.vi	**<u>Read P :</u>** Checks the current Power unit (mW or dBm), and returns the Power measured in that unit.

<u>IV - APPLICATION EXAMPLES.</u>

<u>IV - 1 - The TUNICS Front panel demo VI.</u>

As you can see in Figure 10, this **demonstration** VI reproduces the main controls of the Tunics' front panel. This is an application that **runs continuously** until you click on the ON/OFF button.



<u>Figure 10 : Front panel of the VI.</u>

The VI runs automatically when loaded. The initialization takes 7 **seconds**. After that period, you have to enable the device by clicking on the DISABLED button. Then the VI displays the Current and the Power measured in the corresponding controls.

To change a value, first click on the button at the left of the control. It stops the displaying of measurements, and displays the last valid value sent. If you modify this value, or its corresponding unit, the new value is tested for correct range (that depends on the unit). If it is correct, the new value is sent to Tunics.

This demonstration VI uses nearly all « **basic** » and « **panel** » VI's provided, so if you search for the example of use of a particular VI, just have a look at this block diagram.

IV - 2 - The TUNICS Graph with Rifocs VI.

This VI performs a **wavelength scan**, it is made to draw a **Power versus Wavelength** graph. It makes Tunics emit Power at different wavelengths, and makes a measurement for each value. You can connect an optical device between Tunics and your powermeter.

This demonstration VI is given in two versions :

- *TUNICS Graph with Rifocs*.vi : you must own a *Rifocs 575L* Powermeter, and a RS232 interface link to run this demonstration;
- *TUNICS Random Graph*.vi : measurements displayed are random numbers.

First you must enter the min value, the max value and the step in nanometers. Then you have to adjust the Power. Now you can run the VI, and click on the START button. You can stop measurements by clicking on the STOP button. You stop the VI by clicking on the ON/OFF button when no measurement is in progress.

This is an application that **runs continuously** until you click on the ON/OFF button. VI's used to control the *Rifocs* Powermeter are not provided by *Rifocs*, they are for use in this demonstration only.

INDEX