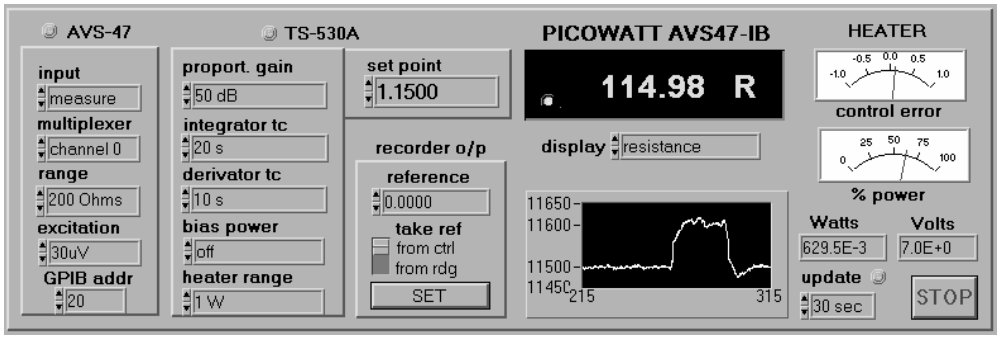
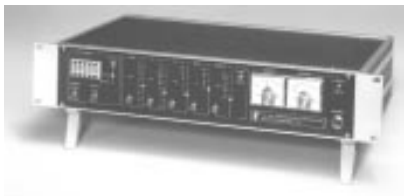
INSTRUCTION MANUAL

LABVIEW DRIVER



For the AVS47-IB & AVS-47 & TS-530A

LabView version 5.01



RV-Elektroniikka Oy PICOWATT
Veromiehentie 14
FIN-01510 VANTAA, Finland
phone +358 9 822087
fax +358 9 822184
Internet: www.picowatt.fi

CONTENTS

1. INTRODUCTION

This LabView driver for the AVS47-IB covers all the functionality of the AVS-47 Resistance Bridge, the TS-530A Temperature Controller and the intelligent AVS47-IB Secondary Interface Unit. The driver more or less conforms to National Instruments' recommendations (or "standards" as NI says), consisting of more than 40 VI's. The first version of the driver is released without any error handling. However, most of the virtual instruments have been written so that they are just bypassed if they cannot work properly, and the VI's themselves often provide error indicators which can be inspected by opening the VI.

Instead of writing drivers separately for the AVS-47 and the TS-530A, we have handled both instruments within the same package. This corresponds to the facts that these instruments are often used together, and that the TS-530A without a resistance bridge is of little value. The AVS47-IB interface box contains many powerful features like computer-independent averaging and scanning. One purpose of this driver is to enhance the use of these features.

One may ask, whether it would be more economical to avoid the AVS47-IB by using the no-cost Picobus primary interface of the resistance bridge. Unfortunately, modern operating systems, like WindowsNT, discourage the use of software that directly accesses computer hardware. However, that is exactly what Picobus needs to do when it reads and operates the hardware handshake lines of the Com: serial port. An additional new reason to use the AVS47-IB is the new optical fibre **PICOLINK**, which is available as an option to provide the best possible EMI performance and to break any ground loops between the AVS-47 and the AVS47-IB.

This driver has been written using a minimum of VISA functions, so compatibility with future platforms should be good. You need not devote your finest computer for running our driver. We have written and tested this driver on a 166MHz Windows 3.11 computer, using an old National Instruments GPIB-PCII card. Hopefully, a newer OS, and a faster computer with a modern GPIB card should work even better. This driver was developed for LabView version 5.01.

NOTE: Check the version of your driver by opening the "AVS47IB VI-Tree.vi" and looking the "Show VI Info" and "Show History".

2. ABOUT THE DESIGN PHILOSOPHY

The AVS47-IB is not just a dummy protocol converter between Picobus and the IEEE-488 interface standards. The fact that the secondary interface unit contains an embedded PC made it natural to incorporate many programmable features like recording averages or scanning several sensors at predetermined intervals. Also buffering of the results was included, as it can sometimes solve time-critical problems.

This kind of features can of course be implemented using an external computer. But if the computer is used to control and read a large number of laboratory instruments, then the workload can make the computer too slow for a timely response or for real-time data acquisition. Despite the development in computer performance and even if LabView is now much faster than years ago, you may encounter timing problems in a large system.

Some help to the performance problem can be achieved if any or some of the instruments can perform time-taking tasks without the computer's supervision. As far as the AVS47-IB is concerned, examples of such slow tasks are averaging and scanning with or without simultaneous temperature control.

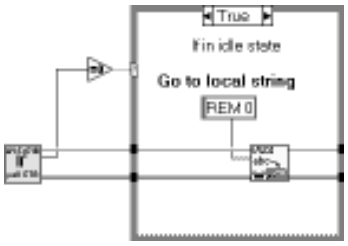
According to National Instruments' recommendations, instrument drivers should contain an **Initialize.vi** that sets the instrument into a known initial state. The drivers should also contain a **Close.vi** that leaves the instrument in a known state. We have deviated from these recommendations. Our **Initialize.vi** sets the system in remote control mode, but it *does not change the state of the instruments*. Our **Close.vi** does even less - it leaves the system in local control mode, but you can also tell it to leave it in the remote mode. Contrary to NI's recommendations, we have also included a number of VI's for *reading the current state* of the system into the LabView. The purpose of all this was to

-allow LabView to use more of its capacity for other tasks
-allow the user to close some virtual instruments in order to make more memory free.
- allow the user to exit LabView and to use the computer for something else
- allow the user to shut off and later re-start the computer,

without disturbing the system, and without disturbing the controlled temperature in the cryostat.

The AVS-47 uses AC excitation whose frequency is only 12.5Hz. Therefore the instrument is very slow, and the measurement commands need a long time to complete. This driver makes extensive use of both the *OPC (operation complete) command and serial polling of the IEEE-488.2 defined Status Byte (STB) in loops waiting either for the operation to complete of for a message to become available (MAV). These loops give their waiting time to other simultaneously running VI's.

Throughout this driver you can see one **common structure: the VISA read and write operations are enclosed in a CASE statement**. The case is TRUE if the AVS47-IB is in idle state. There are only a few commands that the interface can accept if it is not idling, and these situations are handled differently. Usually, the idle state is first verified, and the VISA operation is allowed after that. If the system was not idling, the VISA operation is not performed, and usually some error indicator (like "not in idle state") is set.



Verifying the idle state

Following National Instruments' standards, communications with the GPIB controller card have been realized using VISA. We needed five VISA functions in this driver: VISA open, VISA Read STB, VISA Write, VISA Read, and VISA Close.

VIRTUAL INSTRUMENTS

3. THE FIRST AND THE LAST

3.1 Initialize

Within LabView, **Initialize.vi** establishes communications between the software, the GPIB card and the GPIB device. For that purpose it needs to know the bus address of the AVS47-IB. The factory default address is 20 (refer also to **Configure addresses.vi**). Further, the LabView application which uses this particular AVS47-IB in this particular address is allocated a unique handle, called "VISA session". All the subsequent VI's that belong to this application are normally connected together using this VISA session handle. The VI's are like inserted in a thread, which controls the execution order of the VI's in a multitasking environment.

Within the AVS47-IB, **Initialize.vi** disables the front panel started measurements (the START/PRINT button on the AVS47 front panel). Such measurements shall not be made in the remote control mode. The command is FSM0.

The response message headers are also disabled. After power-on, the AVS47-IB includes headers in all responses. For example, response to the *STB? query is STB16. After the headers have been disabled by HDR0, the response to *STB? will be just the number 16. Responses that contain only numbers, possibly separated by semicolons, are much easier to use in LabView as they need no parsing.

The operation complete -event is enabled by the common command *ESE1. Each time the 47IB encounters the *OPC command, it sets bit 2^0 of the ESR status register. After the *ESE1 command, this bit will be OR'ed into bit 2^5 of the Master Status Byte register STB. By means of polling the STB (serial polling) one can determine whether the operation or a longer series of operations has completed.

These enabling and disabling actions are performed in the **Default setup.vi,** inside **the Initialize.vi**

The system is set into remote control mode using REM1.

The **Initialize.vi** offers two important choices:

1) **Identification query**. If enabled, the AVS47-IB will receive the query *IDN?, and it will answer with response PICOWATT,AVS47-IB,0,2.0. If the AVS47-IB is not idling, do not enable this query. As default, the query is disabled. It is safest to keep it disabled, unless you necessarily need to verify the kind of the device or the version of the firmware.
2) **Reset**. The default is not to reset neither the AVS-47 nor the TS-530A, as this can lead to unwanted changes in the system state or controlled temperature. You can reprogram all aspects of the system without first resetting it. If you want to avoid changes in the system state, find the system status by first reading it. Refer to the AVS47-IB manual for the reset states of these instruments.

Using these defaults, you can initialize the VISA session without disturbing the system.

Note that the **Default setup.vi** is not run if the AVS47-IB is not idling. The state is verified by polling the STB.

3.2 Close

The **Close.vi** contains two actions. First, it runs the **Go Local.vi** that sets the AVS-47 in the local control mode using REM0. Here again, as with most of the VI's in this driver, the status byte is first polled in order to find whether the 47IB is idling. If it is not idling, the **Go Local.vi** is bypassed without any error message.

As default, going to local is enabled. However, you may want to disable this command, so that the AVS-47 remains in remote mode. Disabling is done from the front panel of the **Close.vi** or by using the corresponding Boolean control.

As the last thing, **Close.vi** closes the VISA session.

If you want to stop computer control only momentarily, consider disabling the REM0 command. Then the system stays in the remote mode, the front panel switches of the AVS-47 are disabled and unintentional changing of the system state is impossible.

4. CONFIGURATION VI'S

According to their definition, configuration VI's change the state of the instrument, but they do not produce any data which should be read. Configuration VI's are used for setting various parameters prior to a measurement so that the measurement can be started by using a simple command.

Because this driver actually covers three devices, the AVS47-IB, the AVS-47 and the TS-530A, the list of configuration VI's is long.

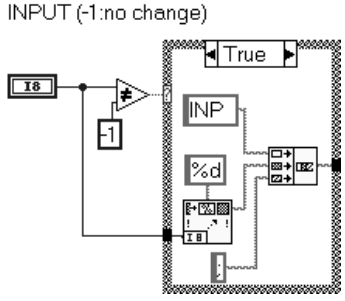
4.1 Configure AVS47

Configure AVS47.vi covers the front panel switches of the AVS-47:

Input: 0..2 (zero, measure, calibrate). The AVS47-IB command is INPx.
Channel: 0..7. The AVS47-IB command is MUXn
Range: 0..7 (no range, 2Ω, 20Ω, 200Ω, 2kΩ, 20kΩ, 200kΩ, 2MΩ). The AVS47-IB command is RANx
Excitation: 0..7 (no exc, 3µV, 10µV, 30µV, 100µV, 300µV, 1mV). The AVS47-IB command is EXCn
Display: 0..7 (R, ΔR, Adj Ref, Reference, Excitation voltage, 530 Heater voltage, 530 Heater current, 530 Set point). The AVS47-IB command is DISx

The inputs of the VI default to -1, which means "no change". Therefore, you can use only one or any of the inputs and leave the rest unconnected. The unconnected parameters will not be configured.

This structure eliminates the need to have five different configuration VI's for the AVS-47. Note that we have used standard digital controls instead of the more readable ring controls. The reason is that the numeric range of the ring controls starts from 0, negative values being impossible. Any other solution seemed to be subject to confusion.



A building block for the Configure AVS47.vi

Looking at the VI diagram, we see that all parameters are handled similarly. The digital control and (-1) are compared. If the control indicates “no change” (default), then the case structure is bypassed. In the TRUE case, a string consisting of the command, argument and a semicolon is added to the previous string. Assume, for example, that the previous string is “INP 1;” After the second case statement, the string could be “INP 1;MUX 2;”. And so on.

After the command string is complete, the string is sent to the AVS47-IB. However, before this is done, the AVS47-IB is serially polled (**poll STB status.vi**). The status must be 0 (idle), otherwise one cannot send commands to the interface. The device status can be other than 0 if, for example, the interface has been previously programmed for scanning and this procedure has not yet ended.

The “not in idle state” output is wired to a LED indicator.

4.2 Configure AVS47IB

Configure AVS47IB.vi can be used to configure one or any of the following:

Printer enabled: Enables (1) or disables (0) printing of responses to ALL queries on a parallel printer (LPT1:) which must be connected to the printer port of the AVS47-IB. This may not be a useful feature in a LabView application, because the message headers must be disabled for this driver to operate, and therefore the printout will be very difficult to read and to interpret.

The AVS47-IB power-on default is “disabled” and the LabView default is “no change”(-1). The AVS47-IB commands are PRN0 and PRN1.

Buffer enabled: Enables (1) or disables (0) saving responses to a RAM disk buffer. Each command line, consisting of commands that are separated by semicolons, results in one response line, where the items are separated by semicolons. Note that this driver requires the response message headers to be disabled. Therefore the response line will consist of only numbers, which appear in the same order as the commands.

The buffer size is 64kB and the remaining space can be read using **Read and save buffer.vi.**

The AVS47-IB power-on default is “buffer disabled” and the LabView default is “no change” (-1). In 47IB: DSK0 and DSK1.

Message headers included: Enables (1) or disables (0) headers in the response messages. Default is “disabled”.

After power-on, the AVS47-IB echoes all command and query headers in the responses. For example, if the combined query is RAN ?;MUX ?, the response will be RAN 5;MUX 0. Once the message headers have been disabled, the response will shrink to 5;0.

A response without headers is more difficult to read by eye, but it much easier to read by the computer. The **Default setup.vi**, which is used inside the **Initialize.vi**, disables the headers which is a prerequisite for this driver to work. Presently, this driver does not contain any situation where the headers should be enabled.

If you want to study the operation of the AVS47-IB by using the **Discuss with AVS47IB.vi** application example, then you may want to turn the headers on by using command HDR 1 from **Discuss**.

FSM enabled: Enables (1) or disables (0) the front panel started measurements. Default is (-1) “no change”.

The AVS-47 offers a possibility to make a standard measurement by lifting the START/ PRINT button momentarily. The measurement consists of a predetermined number of A/D conversions (the number depends on excitation), whose average is calculated. The results are printed, if a printer is connected to the printer port. Also, the results are stored in the RAM disk buffer *regardless* of whether buffering is enabled or disabled. The reason for this decision was to enable printing from the buffer without needing a computer for setting the parameters. The important thing to notice is that the lines in the buffer will contain response message headers. One must avoid situation where the buffer may contain lines both with headers and without them, as parsing would be difficult. Therefore, keep FSM disabled. The **Default setup.vi** in the initialization section disables the headers. The AVS47-IB command is FSM 0.

Autoranging enabled: If autoranging is disabled (0), any measurement, averaging or scanning, will be conducted using the range that was selected prior to the command. The result can be only valid or overrange. It is your responsibility to check the result (using any of the overload indicators that are provided by the appropriate data VI’s).

If autoranging is enabled (1), the AVS47-IB will change range upwards whenever the reading exceeds 19999. If this happens when an average is being recorded, recording is re-started on the new range. Upranging can continue up to the 2MΩ range, then the reading is taken anyway.

If an underrange (reading less than 1800 counts) is encountered, the AVS47-IB changes range downwards. Averaging, if in process, is re-started on the new range. Downranging can continue down to the 2Ω range, then the reading is taken anyway.

Averages that were taken with autoranging enabled can never be invalid due to an A/D converter overload (except if the resistance exceeds 2 MΩ). The same applies also to scanning. Autoranging has been supported by the AVS47-IB also so that all resistance queries calculate the correct value regardless of the range or deviation magnifier.

In a **temperature control application, do not enable autoranging**, as this will lead to unpredictable results.

The AVS47-IB power-on default is ARN 0 (autoranging disabled) and the LabView default is "no change" (-1).

Stabilisation delay: This is the delay between two successive autoranging operations. Its purpose is to allow the bridge some time to find a balance before making the next measurement and before deciding whether still more autoranging operations are needed. In other words, one stabilisation delay follows each autoranging operation.

A too short stabilisation delay may lead to oscillation between two ranges. A too long delay will make the system unnecessarily slow. The length of the suitable delay depends on excitation, and on whether the anticipated jumps are large (like when multiplexing to another sensor).

This stabilisation delay is used only when autoranging is enabled. It has no effect if you change the range manually and then start a new measurement.

The SCAN procedure has its own stabilisation delays for each sensor. These delays can be tailored to suit the excitations used for these sensors. See also **Configure channel parameters.vi**.

The AVS47-IB default value is 15 seconds. This is usually enough for all situations and often a shorter delay is sufficient. The LabView default is "no change" (-1).

4.3 Configure addresses

Configure addresses.vi is used to set the

GPIB device address: The GPIB address can be within range 1..30. The factory default address for the AVS47-IB interface is 20.

Please note that the GPIB address is stored in an EEPROM memory residing in the CPU of the AVS47-IB. The **write count** of this EEPROM **is limited to 10000**. The write count is incremented each time you run **Configure addresses.vi** AND either the GPIB address OR the Picobus Delay Factor is not -1 (no change). Exceeding the write limit may lead to an unrepairable fault (the whole computer must be replaced). Therefore, *limit the use of this VI to temporary occasions*. Under no circumstances include it in any section that may be run routinely.

Picobus address: Picobus address is the address used by the Picobus protocol to identify the Picobus instrument. Valid range is 1..15. The Picobus address is set by a DIP switch inside the AVS-47 resistance bridge. Factory default for all AVS-47 bridges is 1. You need not change this value unless you have more than one AVS-47 connected to a single AVS47-IB interface. In such a case, the two bridges must have different addresses.

After power-on, the AVS47-IB defaults to Picobus address 1 (PBA 1). Use **Configure addresses.vi** for changing this value later, if needed. Changing the Picobus address does not increase the write count, because this number is not stored in the EEPROM.

The LabView default is -1 (no change).

Picobus Delay Factor: Picobus Delay Factor (PBD) determines the end count of a simple for-next program loop, which is used to adjust the Picobus speed so that the waveform shapes in the Picobus interface line are reasonably good. The factory default value has been selected to suit the standard 5 m cable and the speed of the computer in the AVS47-IB. You should not change this value unless your cable is much longer, or if you want to insert RF filters in the Picobus line. In that case, you need to make the delay longer. This need manifests itself by more or less erroneous readings, or by failures in the remote control.

Experiment with other PBD values until you find the speed limit. Select a new PBD so that it is well slower than this limit.

Changing the Picobus Delay Factor will increment the limited EEPROM write count. Refer to the discussion of the GPIB address.

4.4 Configure TS530A

Configure TS530A.vi is used to set values for any or all of the PID parameters of the controller:

Set Point: Valid range is from 1 to 42000 corresponding to 100µV..4.2V.

LabView default is "no change" (-1) whereas the power-on state of the TS-530A set point DAC is random.

Proportional gain: Valid range is from 0 to 15. Values from 0 to 11 correspond to gains of 5, 10, 15...60 dB. Values 12, 13 and 14 are forbidden. Value 15 short circuits the error signal to ground.

If you want to make changes in the input conditions during temperature control, or if you want to measure another sensor temporarily, then short the error signal and latch the integrator (infinite time constant).

Note that the indicated gain values are not overall gains, which depend also on the output stage and the heater. Rather, they indicate the gain steps that are available.

TS-530A power-on default is 0 and LabView default is "no change" (-1).

Integration time constant: Valid range is from 0 to 11. 0 and 10 both indicate the PD (proportional + differentiate) mode where the integrator stays reset. Values from 1 to 9 correspond to time constants 1, 2, 5, 10, 20, 50, 100, 200, 500 and 1000 seconds, respectively.

The integrator can be latched by setting TI=11. Latching means that the time constant is infinite. However, because the integrator uses analog technology, there will be slow drift either upwards or downwards.

TS-530A power-on default is 0 and LabView default is "no change" (-1).

Derivation time constant: Valid range is from 0 to 7 corresponding to derivator time constants of 0, 1, 2, 5, 10, 20, 50 and 100 seconds.

TS-530A power-on default is 0 and LabView default is "no change" (-1).

Power bias: Valid range is from 0 to 5 corresponding to 0, 20, 40, 60, 80 and 100% of the full power of the range. The bias power drive signal is summed to the power drive signal from the PID controller (which may be either positive or negative).

Bias power is useful in the PD control mode, where it can reduce the magnitude of the steady-state error.

TS-530A defaults to 0 at power-on, whereas this VI defaults to "no change" (-1).

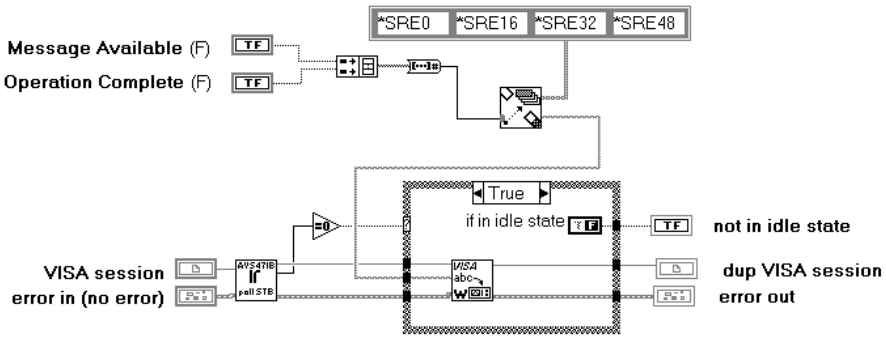
Heater power range: Valid range is from 0 to 7 corresponding to OFF, 1µW, 10µW..1W.

These values apply to a 100 ohm heater. If the heater's resistance is lower than 100 ohms, the maximum output current will limit the power (100mA on the highest range, then divide successively by 3.16). If the heater's resistance is higher than 100 ohms, the heating power will be limited by the maximum output compliance voltage, which is 10 Volts. Refer also to the TS-530A manual.

Power-on default of the TS-530A is 0 and LabView default is "no change" (-1).

You can use **Configure TS530A.vi** to program one or more of the PID parameters by giving values to the corresponding inputs. Those parameters that are left unwired, or whose values are set to -1 are not changed. Note, however, the way how the AVS47-IB interface works. Whenever you change at least one of the PID parameters, all parameters will be sent to the TS-530A. This happens regardless of the fact that this VI may generate only a short command string with just one parameter command, for example.

The purpose of this behaviour is to make sure that all parameters correspond to values that the interface has in its memory, that nobody can make permanent changes in the settings by using the front panel switches. In the case of the AVS-47, this has been accomplished by disabling the front panel controls, but because the design of the TS-530 is much older, this method was not available.



Enabling service requests

4.5 Enable service requests

You can use the **Enable service requests.vi** to enable either one or both of the following events to generate an GPIB service request: Message available (MAV) or Operation Complete (OPC).

Above is the diagram of this VI. The two Boolean controls (MAV and OPC) are first combined into an array of two elements (build array). Then this Boolean array is converted into an integer number. Commands required for programming the SRE (Service Request Enable) mask register are listed upmost in the diagram as array elements (refer to the AVS47-IB manual for more information on the masks and the status reporting in general).

There are four commands corresponding to the four possible combinations of the two inputs. The “index & append” function selects one of these elements to be sent to the AVS47-IB. Whether the command is written depends on the state of the interface. This is found by polling its status (**Poll STB status.vi**). Zero status indicates idle state and writing is possible. A nonzero value results in setting an error flag (“not in idle state”) and bypassing the VISA operation.

This driver does not use service requests nor interrupts. Instead, all waiting loops have been designed by using only serial polling. Then, using a delay, one has complete control over the behaviour of the system. By making the delay shorter, polling is made more frequently and the response toward the instrument is faster. By making the delay longer, polling is made more seldom, but other tasks, processes or programs that are running in the computer get more CPU time with less overhead. LabView 5 offers a VISA function “Wait SRQ”, which could probably have been used. Unfortunately, the behaviour of this function has not yet been described in detail so that it was safer not to use it.

4.6 Configure time and date

Configure time and date.vi takes the current time and date from the computer and initializes the real time clock of the AVS47-IB to these values. This VI uses the “Format date / time string” -function. The format strings make the outputs to suit the interface, and commands for both the time and date are combined. The combined command could look like DAY 2000,1,31;TIM 11,55,59 (31st January 2000, at 11:55:59 am).

You will need this VI if you save scanning result in the buffer. These results have time labels, which will be of best use if the clock time is correct.

The clock inside the AVS47-IB is not battery-backed. Therefore it will loose its value in case of power outage. This driver version (1.1) does not provide means for detecting such a condition.

4.7 Configure scan parameters

The **Configure scan parameters.vi** is used to configure one or any of the general settings that control the SCAN procedure. These settings do not depend on the channel number being measured (the channel-specific parameters are given by the **Configure channel parameters.vi)**. A few rules must be obeyed.

First channel: This is the first channel to be included in the SCAN procedure. It can be anything from 0 to 7. **Except**: If the TS-530A temperature controller is incuded in the system, the control sensor *must* be connected to channel 0, and this channel *must not* be included in the SCAN sequence. AVS47-IB default is 0 and LabView default is “no change” (-1).

Last channel: This is the last channel to be included in the scan sequence. It must be greater or equal to the first channel. If first channel = last channel, then this one sensor is measured using those channel parameter values that were given for this sensor. AVS47-IB default is 0 and LabView default is “no change” (-1)

Scan interval: Scan interval (SCI) is the time between two successive starts to measure the first channel. This parameter is meaningful only in the continuous scan mode. The scan interval must be longer than what is actually needed to measure all the sensors that are included in the sequence. If it is shorter, the new cycle begins immediately after the previous one. Valid range is from 0 to 10000 seconds.

Note that it is not possible to specify the exact times when the sensors are measured. This is due to at least two facts: 1) Possible autoranging operations can cause unpredictable delays and 2) In order to separate a real zero reading from an A/D converter overload, one has to repeat a couple of times a measurement that results in a zero reading. So some measurements can be randomly longer than the others.

If temperature control is enabled, the scan interval must be so long that both the scan sequence and the temperature control period have time to complete before starting the next cycle.

AVS47-IB default for SCI is 600 seconds and LabView default is “no change” (-1).

Enable temperature control: In addition to scanning sensor channels from 0 to 7, it is also possible to allocate sensor 0 for temperature control and to scan one or more of the remaining 7 channels.

If temperature control is enabled, the SCAN procedure is changed so that the temperature control is active for a period called TCP (temperature control period) between measurements of any two successive channels. Temperature control is active also for the time after the scan cycle but before the next cycle starts at the time determined by the SCI .

The combination of the temperature controller and scanning is explained in more detail in the AVS47-IB manual.

AVS47-IB default is “disabled” (ETC 0) and LabView default is “no change” (-1).

Temperature control period: This is the time used for temperature control between scanning any two successive channels. The temperature control period must be chosen experimentally. The idea is to prevent the system from drifting too far from the thermal equilibrium during the SCAN procedure, when the integrator is latched and the controller outputs a constant heating power instead of an active control.

Valid range is from 0 to 1000 seconds. If TCP=0, the controller will be active only between the ends and next starts of the scan cycles.

4.8 Configure channel parameters

Configure channel parameters.vi is used to specify the range, excitation, stabilisation delay and average count for each sensor channel separately.

You must use this VI once for each channel, it is not possible to program several channels at the same time.

The AVS47-IB requires a command string like SCP5;RAN4,EXC3;SDY10;CNT30. This string means the following: Commands that follow on this line are intended for sensor channel No. 5. This channel is to be measured on range 4 (2 kOhms), using excitation 3 (30µV). The stabilisation delay after having selected this channel and its parameters, and also in case of an autoranging operation, must be 10 seconds. The final result shall be the average of 30 successive A/D conversions.

The diagram of the VI is very similar to those where the AVS-47 or the TS-530A are configured, except that the first entry, the channel number, is mandatory. You can leave any of the channel parameters unchanged by not connecting the inputs or by setting them to -1.

The combined command string will be written only if the interface is in idle state.

Channel: 0 to 7. If temperature control is enabled, then the valid range for channel is from 1 to 7. This entry is mandatory, there is no default.

Stabilisation delay: Any time a new sensor channel is selected, the AVS-47 bridge needs some time to settle. The same applies also to an automatic change of range. The settling time depends on the excitation - the bridge is faster when higher

excitation is used. By using different stabilisation delays for different sensors, you can optimise the speed of the scan procedure but prevent premature measurements.

Valid range is from 0 to 100 seconds.

AVS47-IB default for all channels is 15 seconds, which is sufficient even for the lowest excitation. The LabView default is “no change” (-1).

Average count: Generally, use an average count greater than unity. It is meaningful to improve the confidence of the readings by averaging. There are many delays in the SCAN procedure anyway, so that short averages do not make the cycle much longer.

Valid range is from 1 to 1000. Each A/D conversion need 0.4 seconds.

AVS47-IB default is 10, the LabView default is “no change” (-1).

Excitation: Valid range is from 1 to 7. AVS47-IB default for all channels is 1 (3μV). This value was chosen in order to prevent incidental overheating of the sensors. LabView default is “no change” (-1)

Range: In manual range mode, this setting will be used regardless of the result. It is your responsibility to verify that the result is not invalid. Because this is difficult to do while scanning, use autoranging whenever possible. Autoranging guarantees that the results are always valid (except if the resistance is over 2MΩ).

In autorange mode, this parameter defines only the initial range. The initial range is tried when this sensor is measured for the first time. If needed, one or more autorangings are made until a valid reading is obtained. Then the average is recorded. The new range setting will be remembered when this sensor is measured during the next scan cycle. This means that the next scan cycles may take much less time.

AVS47-IB default is 2MΩ for all channels, and the LabView default is “no change” (-1). If you use autoranging, you do not need to select any other initial range value. The highest range is a safe guess that does not cause accidental overheating.

4.9 Configure reference voltage

The deviation reference voltage of the AVS-47 can be set remotely in two ways by using the **Configure reference voltage.vi**:

Programming mode: The default programming mode (0) is to specify the reference voltage. The alternative mode (1) is to read the A/D converter and use this reading as the reference. The latter mode effectively means nulling the deviation signal.

Reference voltage: Valid range is from 0 to 2.0000 Volts. Because the reference D/A converter has only 12 bits (corresponding to 4096 counts), the allowed voltage values must be rounded to the nearest number that is divisible by 5. Therefore the step of this digital control has been set to 5.

5. ACTION/STATUS VI’S

According to National Instruments’ definition, the Action/Status category contains two types of VI’s. Action VIs start or stop measurements. They do not change the settings of the instruments otherwise. Status VIs obtain the current status of the instrument. This driver contains only one status VI, **Poll STB status.vi**.

5.1 Go remote

Go remote.vi sends only one command to the AVS47-IB, and that is REM 1. The interface in turn reads the current settings of the AVS-47 resistance bridge to be its new initial values. During this transaction, the remote bit is held reset so that the bridge remains in local mode. The AVS47-IB then sets the remote bit and makes a new transaction with the bridge. Now the bridge enters into the remote mode, but at the same time all the switch settings are programmed according to their previous values.

As the result, **Go remote.vi** does not change the settings of the AVS-47.

This VI is used only once, in the **Initialize.vi**.

5.2 Go local

Go local.vi sends only one command to the AVS47-IB, namely REM 0. Within this driver, **Go local.vi** is used only once, in the **Close.vi**.

Usually you want to leave the system in the local control state so that the front panel switches are active. If you want to prevent anyone from changing the settings, you can also close you LabView application so that the AVS-47 is left in the remote control mode. Refer to **Close.vi**.

5.3 Poll STB status byte

This is the only status VI in this driver. **Poll STB status byte.vi** can be used regardless of the state of the AVS47-IB. This is because serial polling is handled by the 7210 GPIB controller quite independently of the computer.

The application program in the AVS47-IB updates a byte which we call "serial poll response" whenever the program moves from one phase to another. The serial poll response is always available to the 7210, which uses this value as the Status Byte (STB) if a serial poll is made.

The STB of the AVS47-IB contains the following information. When reading the rather complicated description, please note that we have done all the necessary operations for you. Most probably you will not need to use this VI yourself at all.

bit 2^6 **SRQ**. The system is requesting service. This bit can be set only if one or more events have been first enabled, and any of these enable events occur. Refer to **Enable service requests.vi**. The service request is a physical signal line in the GPIB bus. It is directly wired together with this SRQ bit.

Usually, GPIB controller cards offer a possibility to generate a hardware interrupt from the SRQ signal line. This is one way to detect an SRQ.

It is also possible to repeat serial polling and inspect bit 2^6 until it indicates an SRQ. This is the second way to use SRQ.

This driver does not use SRQ, but the few events that are important can be directly seen from the STB status byte by serial polling.

Serial polling the STB resets this bit (it is required by the IEEE-488.2 standard), which means that SRQ can be detected only once per event. Before an event can generate a new SRQ, the Event Status Register must be reset.

bit 2^5 **OPC**, operation complete. An OPC event can set this bit only if appropriate mask register bit has been enabled. This is done in the **Initialize.vi**.

In order for the OPC event to appear, one must first clear the Event Status Register and also place a *OPC command as the last item in the command line.

bit 2^4 **MAV**, message available. This bit in the STB indicates that the output buffer of the AVS47-IB is not empty. As the interface will never produce output by itself, MAV=1 can only mean that the response to a query is available.

bits 2^0..2^3 This group of four bits (16 values) describes the state of the interface. You may not need to use the **Poll STB Status.vi**, but nevertheless we describe the status signals, as this can help in understanding the behaviour of the instrument.

0: **Idle state.** The 47IB accepts all commands and queries. Idle state means that the unit is not measuring, not scanning, not controlling temperature neither doing anything else. It is just idling.

1: **Averaging.** The unit is making A/D conversions at maximum speed for calculating their average. Only two commands are allowed: STP (stop) and *RST (reset).

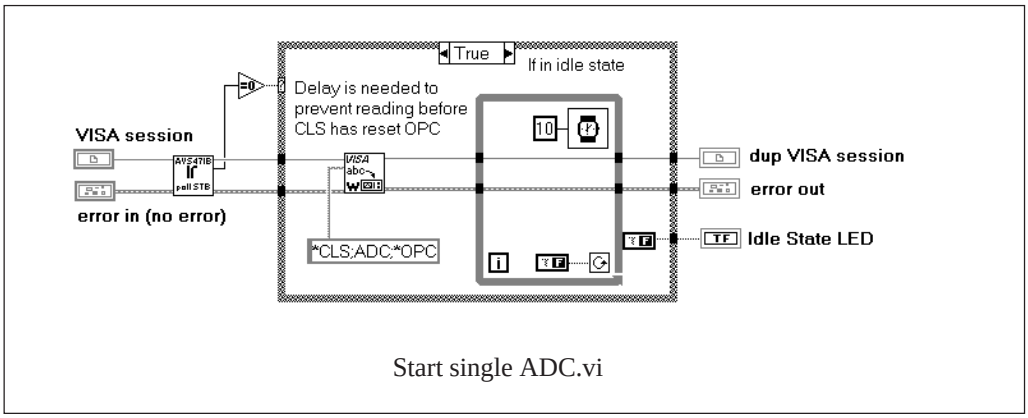
2: **Scanning.** During some phases of the scan procedure, the unit may also show state No. 1. Only STP and *RST commands are allowed.

3: **Scan interval delay**. The unit is waiting for a new cycle to start in the continuous scan mode. The temperature controller is active if temperature control has been enabled. Only STP and *RST are allowed.

4: **Temporary temperature control during scanning.** This means the short temperature control periods (TCP) between measurements of successive channels. Only STP and *RST are allowed.

5: **Waiting for the START/PRINT button in the front panel started scan mode**. You should never see this state in a LabView application! Front panel started measurements are incompatible with this driver. Only STP and *RST are allowed.

6: **Digital filter in progress.** A/D conversions are being taken at maximum speed and the readins are used to fill a circular buffer. It is allowed to make a DFR? (digital filter reading) query, and to stop (STP) or reset (*RST) the interface.



Start single ADC.vi

7: **Printing the buffer on printer**. The interface goes into this state after a PBF command (print buffer file), and it exits back to state 0 once printing is complete. Only STP and *RST are allowed.

8: **Reading the buffer in RBF mode.** The interface goes into this state after an RBF command, and remains there as long as there are lines to be read or until reset. It is allowed to make the NXT? query for receiving the next data line, or to give the STP or *RST commands.

9: **Self-Calibration in progress.** Only STP and *RST are allowed.

10: **Self-test is in progress.** The procedure cannot be terminated. No commands are allowed.

5.4 Start single ADC

The **Start single ADC.vi** is used to make one measurement without any averaging. Note that if the system is in autoranging mode and the initial range is not correct, several conversions may be needed until a valid reading is obtained. Therefore one cannot be sure about the time this VI requires.

Like with almost all commands in this driver, the idle state is first verified. The command string consists of

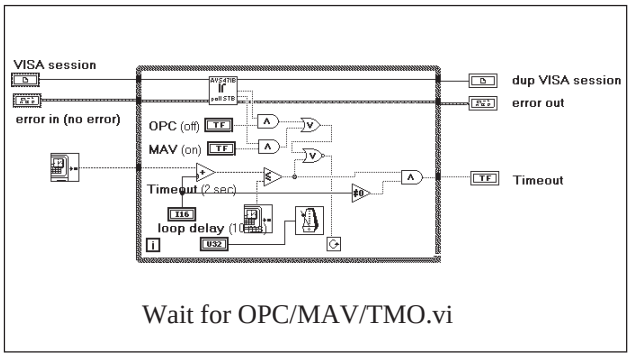
1) *CLS Clears (resets) the status register so that it can reflect new events

2) ADC Tells the AVS47-IB to wait for the completion of the **next** analog-to-digital conversion and to give its result. This command should need a maximum of 0.4 seconds.

3) *OPC When the command interpreter of the AVS47-IB encounters this command, it sets bit 2^0 of the Event Status Register (ESR). This in turn has been enabled to turn on bit 2^5 of the status byte STB (see **poll STB status.vi**).

Once this command string has been sent to the interface, the VI continues with a short delay loop (10ms). This delay is very important: Suppose that the OPC status of the interface was true before the ADC command. If LabView now continues very quickly by starting to wait for the OPC to come true, it will be satisfied immediately. Some time must be allowed for the interface to reset the status before proceeding.

Note that you must verify that this operation has completed before trying to read its result. There is the **Wait for OPC/MAV/TMO.vi** available for this purpose.



Wait for OPC/MAV/TMO.vi

5.5 Wait for MAV/OPC/TMO

The **Wait for MAV/OPC/TMO.vi** is the general waiting loop in this driver. It can be programmed to wait one or more of 1) **Message available**, 2) **Operation complete** or 3)**Timeout**.

The MAV and OPC are Boolean controls. If both are enabled the VI waits for the first of them to occur. If both are disabled of if they fail to occur, the VI waits only for the timeout.

The timeout value (in seconds) is set by a digital control. This waiting loop is not used for generating millisecond delays. Therefore the 1 second step is adequate. The timeout is generated by recording the computer time before entering the while-loop, by adding the timeout number to this reading, and by comparing the sum against future computer time while the loop is running.

If you set the timeout control to zero, the logic prevents the VI from generating the Boolean "Timeout" error signal. This feature is required when reading results from a scan sequence.

The loop delay is an important parameter. During this delay, the while-loop gives its time co-operatively to other simultaneous tasks. The longer is the delay, the better other tasks can run during this loop. On the other hand, the MAV and OPC status will be checked more seldom. The result may be that your application becomes too slow. This is a tradeoff that you have to make.

It seems that Windows 3.11 cannot realize delays shorter than 55 milliseconds. Therefore it may be necessary not to use this co-operative delay at all under Windows 3.11. The loop delay can be turned off by setting the timeout value to 0.

5.6 Start averaging

The **Start averaging.vi** begins a series of A/D conversions whose average is counted and can be queried using a separate VI.

This VI needs one input parameter, which is the average count. The measurement is made with those settings that were valid before this command. The average count can range from 1 to 1000. Each conversion needs 0.4 seconds. Taking an average of 100 conversion needs 40 seconds, so that this is not a very fast procedure.

The combined command string is formed from *CLS (for resetting the status) followed by AVEx where x is the average count, and ending by *OPC. The last command gives us the possibility to wait until all A/D conversions have been made.

There is a short delay after this command, before the VI exits. This delay prevents any attempts to read the STB status before the AVS47-IB has had time to reset it first.

5.7 Start digital filter

The **Start digital filter.vi** sets the AVS47-IB into a special mode, where the selected channel is read as fast as possible (0.4 sec/sample) using the range and excitation settings that were valid before this VI. The readings are maintained in a circular buffer and read using **Read digital filter.vi**. This VI has one input, the filter length. Valid range for the filter length is from 1 to 1000. You can probably find values from 10 to 100 to be most useful.

If the 47IB box is in digital filter mode, it is not idling, but polling the STB status will return 6.

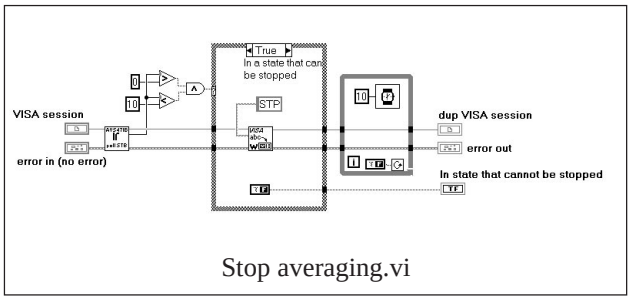
Most commands/queries cannot be used in the digital filter mode. VI's that expect the AVS47-IB to idle are just bypassed and their "not in idle state" error flag is set. The only VIs that will work are

"Read digital filter.vi" for obtaining one reading from the circular buffer. Refer to the description of this VI.

"Stop.vi" for exiting the digital filter mode. Refer to the description of this VI below.

In addition, you can always use the **"Poll STB status.vi"** to find out the present state of the interface.

The digital filter mode is a fast way to get high-quality readings quickly, if you can keep it running. Use a long filter if the excitation is low. The filter function is a simple running average. When the filter is started, after an autoranging operation or after a large sudden change in the reading, the filter is reset. Then the reading is the sum of the cumulated results divided by their number, until the buffer is full.



Stop averaging.vi

Because of its nature, the digital filter mode is not suitable for multiplexing. In such case, use normal averaging, or even better, use the SCAN procedure.

5.8 Stop averaging etc.

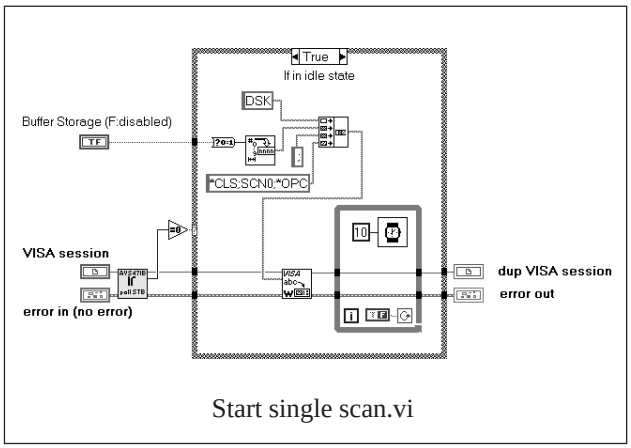
The **Stop.vi** is used to stop the following operations:

- digital filter mode
- averaging
- scanning, especially the continuous scan mode.

The only situations where stopping should be routinely needed are exiting from the digital filter and continuous scan modes, because these modes never stop by themselves.

Stop.vi can also be used to abort printing etc. (see also the description of **Poll STB status.vi).**

The VI starts with polling the STB. The status is checked to be within the range from 0 to 10 (see **poll STB status.vi**). If the system is in a state that can be stopped, the STP command is sent to the interface. In the negative case, the VISA operation is bypassed.



Start single scan.vi

The following delay is very important. It gives time for the AVS47-IB to stop the operation and to update its status byte, which is used for response to a serial poll. If the LabView application continues by reading the status before it has been updated, an error situation will result.

5.9 Start single scan

Start single scan.vi initiates one scan cycle. After all phases of the cycle have been completed, the interface returns to the state that was valid before starting to scan. It means the original channel, range and excitation. A scan cycle can be started only when the AVS47-IB is in idle state. So this is verified at first.

This VI has one Boolean input parameter, Buffer storage, which is disabled by default. When buffer storage is *disabled (false)*, you must read the results after the scan cycle has ended. The **Read scan results.vi** has been designed for this purpose.

If buffer storage is *enabled (true)*, you can also read the scan results later from the buffer. Because results of new scan cycles are appended to the buffer, you can also read the results of several cycles at one time. The results contain time stamps.

The combined command string consists of
- disabling or enabling buffer storage (DSK 0 | 1)
- clearing the interface status (*CLS)
- starting a single scan (SCN 0)
- setting the OPC indicator bit in the STB when the cycle is complete.

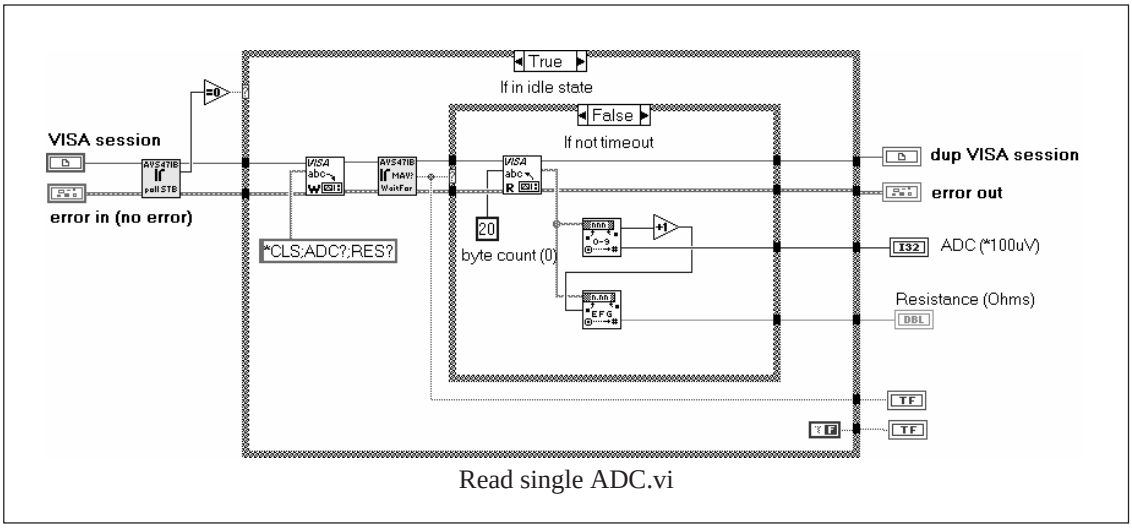
A short delay after the VISA operation guarantees that the status cannot be polled before the 47IB has had time to reset it first.

5.10 Start continuous scan

The **Start continuous scan.vi** differs from the **Start single scan.vi** only in one place: the SCN 0 command is replaced by SCN 2.

When using continuous scan, **it is almost necessary to enable buffer storage**. This is because you *cannot* ask scan results while the system is busy. It is better to use single scan cycles rather than to stop the continuous mode. If the continuous mode is stopped during a cycle, the scan results may belong to different cycles.

Do not use the **Read scan results.vi** after the continuous mode. Instead, use the **Read and save buffer.vi**.



Read single ADC.vi

6. DATA VI'S

According to their definition, data VIs transfer data to and from the instrument. Data VIs for the AVS47-IB all belong to the latter category, there is no data that could be sent to it (except data which is used for configuring it).

The general rule when trying to read anything from the AVS47-IB (or from any other GPIB instrument, for that matter) is that one must always verify there is something in the output buffer. Otherwise, the GPIB bus will hang up until it decides this is a timeout.

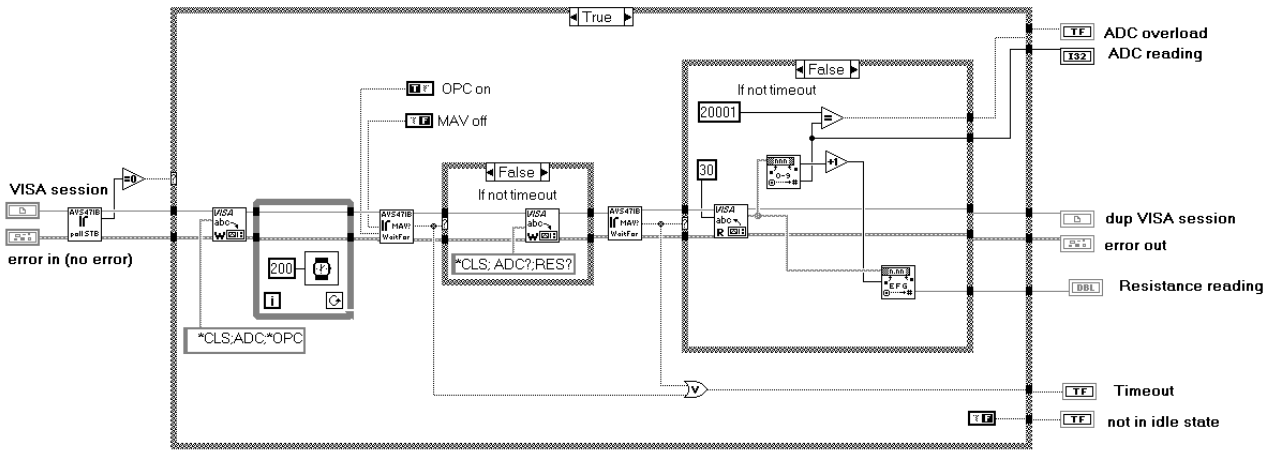
Many AVS47-IB operations take a very long time in the computer scale. One A/D conversion needs 0.4 seconds, taking an average of 100 readings nees 40 seconds, and a longer scan cycle with temperature control enabled may take some tens of minutes. It is necessary to use waiting loops which continuously poll the interface status, but which also give all their spare time to other tasks. If not done so, the computer would be useless during all this time.

The waiting loop that has been used in most data VIs is the **Wait for OPC/MAV/TMO.vi**.

6.1 Read single ADC

The **Read single ADC.vi** is intended to accompany **Start single ADC.vi**. However, you cannot connect these two VIs directly after each other, but you must first verify that the previous ADC operation has completed. Use the **Wait for MAV/OPC/TMO.vi** for that purpose. The ADC conversion result can be read only if the system is in idle state.

The AVS47-IB will receive the following command string: *CLS;ADC ?;RES ? At first, it resets its status. The next query obtains the result of the last A/D



Read one reading.vi

conversion as a signed integer, without any knowledge on range or display mode. Note that a delay between *CLS and ADC? is not needed, because we do not try to poll the status. The RES? query obtains the A/D conversion result as a *resistance*. The resistance is calculated by the AVS47-IB on the basis of the range and the position of the ΔR magnifier switch. Resistance is a floating point number.

NOTE 1: If the display selector is not within 0 and 4, the resistance readings are meaningless. Only the ADC? query shall be used for reading the excitation voltage, heater current etc.

NOTE 2: Because of the limitations of the AVS-47 circuitry, the decimal point on the LED display of the bridge is hard wired to the ΔRx10 switch. If the switch is in the magnifier mode, but the display selector is not in the ΔR position, the decimal point will be in a wrong place. This driver works correctly, however. The same applies also to the AVS47-IB interface, which calculates the results correctly regardless of the switch settings.

NOTE 3: If you have instructed Windows to use a decimal point other than period (.), update your driver from version 1.0.

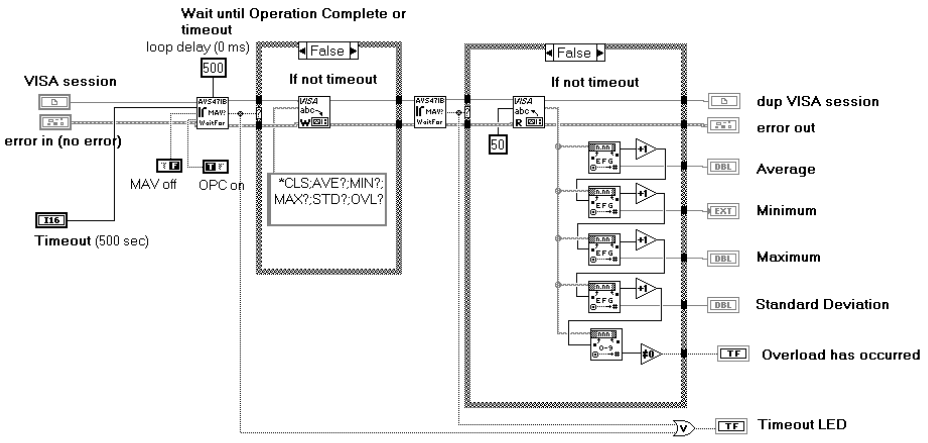
6.2 Read one reading

The **Read one reading.vi** is a self-contained and relatively "easy-to-use" VI for getting one reading from the channel that has been previously configured using other VI's.This is a typical data-VI in that it does not have any inputs at all. On the other hand, the VI contains commands that start an A/D conversion, and these commands would belong to an action-VI. As a matter of fact, **Read one reading.vi** is a combination of both the **Start single ADC.vi** and **Read single ADC.vi** source codes.

If the interface is in idle state, its status is first cleared (*CLS). Then an A/D conversion is started (actually, the AVS-47 has a free-running A/D converter, which cannot be started. Instead, the ADC command instructs the interface to wait for the next conversion to complete and to use its result). The *OPC command requests the AVS47-IB to set the OPC indicator bit in its status register when the conversion has completed.

The short delay prevents the VI from proceeding before the interface has had time to reset the status byte.

By setting the OPC control *true* and the MAV control *false*, the **Wait for OPC/MAV/TMO.vi** is instructed to wait until the end of the conversion. If the conversion does not complete before the default timeout of 2 seconds, an error flag is set, and the rest of the VI is bypassed (such a situation could result from a missing Picobus cable, from a loose optical fibre in the **Picolink**, or if the AVS-47 is simply off).



Read average.vi

The status is cleared again, and the interface is queried for both the integer ADC result and the floating point resistance value.

No delay is required here before entering the waiting loop. This is because the status was cleared once before starting the A/D conversion, and the conversion does not produce any output. So the “message available” -bit cannot be high when the waiting loop starts. If a message does not appear within 2 seconds, the VI exits with the timeout error flag true.

The VISA read -function obtains the integer ADC result and the floating point resistance reading. Note that some display items do not deal with resistance. Then make use only of the ADC result.

The ADC result is compared against 20001. This value, used to indicate overload, is beyond the range of the A/D converter. There is also a separate OVL ? query, but using 20001 is faster as no additional commands are needed.

Note that the resistance has been calculated by the 47IB box: it takes into account the range and the position of the 10*ΔR switch. This reading is always correct.

6.3 Read average

The **Read average.vi** returns, in addition to the average value itself, also the minimum, maximum and standard deviation of all recorded A/D conversions.

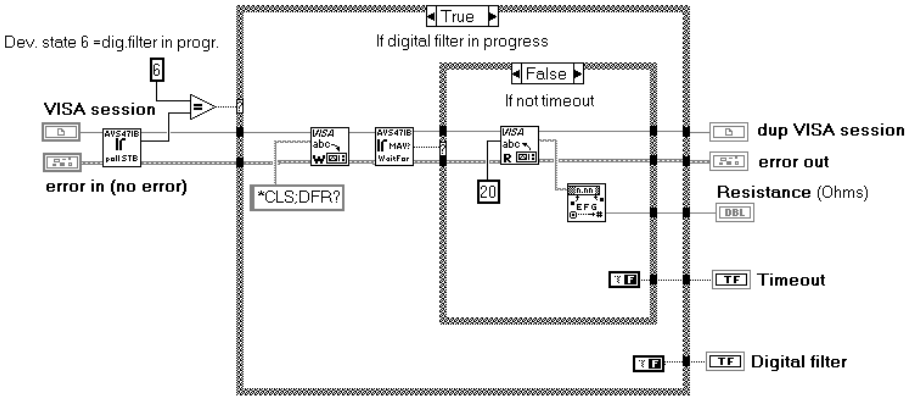
The **Read average.vi** has not been intended to be used with display items other than R or ΔR. The average is a floating point number, corresponding to the resistance in all other respects, but it gives more accuracy.

This VI works as follows: The **Wait for OPC/MAV/TMO.vi** is set to wait for a completed operation, with a timeout of 500 (five hundred) seconds and a loop delay of 500 milliseconds. The timeout value must be really long, so that this VI can be started immediately after having initiated averaging. An average of 1000 readings (this is the maximum) will take 400 seconds.

The loop delay is also long, 500 milliseconds, so that as much time with as little overhead as possible can be given to other tasks. Even a short average will take several seconds, so half a second cannot produce a serious extra delay.

If not overload, the above mentioned readings are queried, followed by the overload query OVL?. Note that now we must use the overload query, because it indicates overload if any conversion resulted overload.

The next WaitFor loop delays operation until the response is ready to be read. The VISA read function receives a string consisting of four floating point numbers followed by the overload indicator which can be only 0 or 1. All items in the response are separated by semicolons.



Read digital filter.vi

6.4 Read digital filter

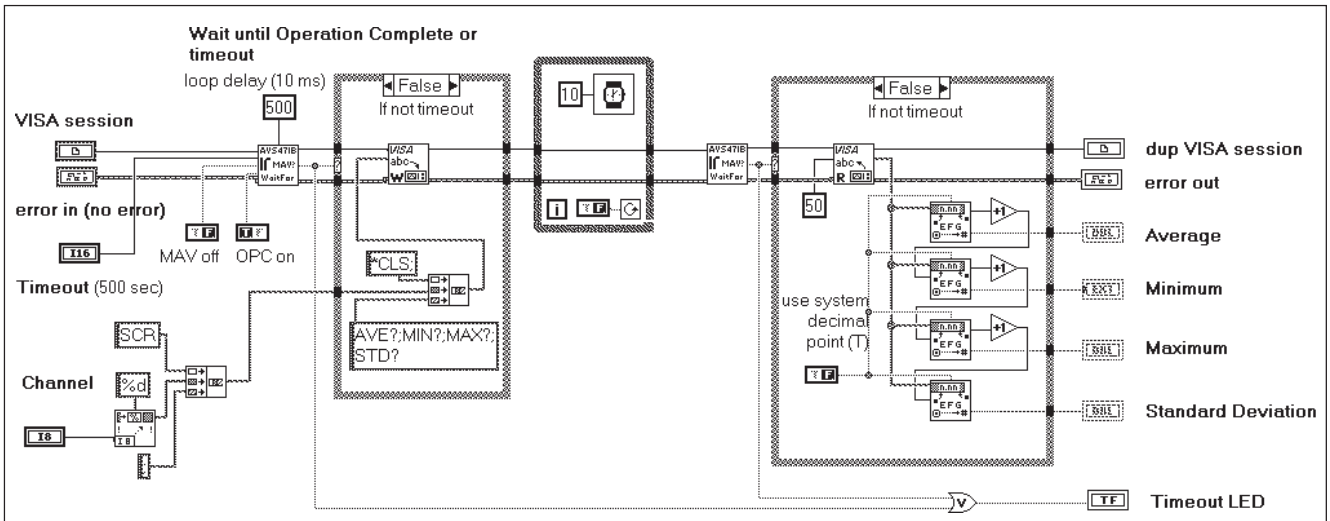
The digital filter reading can be obtained using the **Read digital filter.vi** only if the AVS47-IB is in state 6 (digital filter in progress). Therefore the state is first verified by polling the STB status byte.

The status is cleared and the filter reading is queried by command DFR?.

The response is read once it is available in the output buffer of the interface.

The filter reading is a floating point number, corresponding to the “resistance” and “average” values. Digital filter is not suitable for measuring display items such as excitation voltage or TS-530A set point, where range setting or the position of the ΔR switch must not affect the reading.

Do not use digital filter together with multiplexing. Instead, use preferably the SCAN procedure with first channel = last channel.



Read scan results.vi

6.5 Read scanresults

The **Read scan results.v**i should be used only for reading the results of a single scan cycle. It will not be useful for the continuous scan mode. One should stop the continuous mode for asking the results, but after stopping, the results in the memory may belong to different scan cycles.

This VI has been designed so that it can follow direcly after the **Start single scan.vi**. Compare this VI with the **Read average.vi**, and you see many similarities but also some differences.

The first thing to do is to wait until the scan cycle completes. This is made by programming the **Wait for OPC/MAV/TMO.vi** to wait for completion with a long timeout (500s) and a generous loop delay (500ms). Refer to the discussion of **Read average.vi**.

This VI needs one input parameter, which is the channel number. Note that you must ask the results for each sensor separately by varying the channel number. The part of the command string which is needed here is SCRx, where x is the channel number. The total command string starts with clearing the status, then entering into the reading mode, followed by commands for reading everything that is available:
*CLS;SCRx;AVE?;MIN?;MAX?;STD?

The response is read **once it is available** in the output buffer of the interface. If you use this VI after having started a single scan cycle, the VI waits until the cycle has ended and the results are asked immediately after that. You have to repeat this VI for each channel whose data you want to read.

> When doing so, you must necessarily set the timeout to zero for all other than the first VI. This prevents the VI both from waiting for the timeout (default is 500 seconds), and from generating a timeout error output if the timeout control is set to zero. Note that the OPC will not be available for other than the first read.

The response string consist of four floating point numbers, separated by semicolons.

Note that overload has not been queried nor detected from the output. We want to strongly encourage you to use autoranging together with scanning. Then all results will be valid.

6.6 Read and save the buffer

The purpose of the **Read and save buffer.vi** is to enable reading and saving the buffer as a disk file after the AVS47-IB has been operated in the continuous scan mode. It cannot be used for other purposes because this VI is dependent on the unique data format used for buffering the scan results. Reading is possible if the system is in idle state.

This VI is more complicated than those described so far in this manual. Please refer to the diagram of the VI by opening it in LabView.

After having verified the idle state, the size of the buffer and the remaining free space are queried by BUF ?.The response consists of two numbers, the first of which is the size in bytes while the second number tells the free space. The numbers are separated by a semicolon.

The VI requires two inputs, Mode and Filename. If Mode is 0, only the buffer size and the free space are queried. If Mode is 1 AND the buffer size is greater than zero, the buffer is read and saved in Filename. If Mode is 2, the buffer file is also deleted before exiting the VI.

The interface enters in the "read buffer" mode when it receives the RBF (read buffer file) command, although nothing happens yet, except that the state of the AVS47-IB is now 8. The oldest line in the buffer is placed in the output queue by the query NXT ?

The most important part of the VI is a while loop. This loop starts with a short delay. The STB status byte is polled, and if the state is 8, which means there is still something to read, one line is read from the interface. This line is then parsed and converted into a form, which may be called "spreadsheet form". It means simply that the numbers are separated by blank spaces. Such a file can be easily read into a spredsheet program like Excel or QuattroPro. Once the line is in correct form, it is appended to the output file. Then the NXT ? query is repeated and the loop starts from the beginning with the short delay.

The AVS47-IB stores lines in the buffer as follows:

1993,12,6;4,37,24;1.0006E+02;1;0;3;7;0;0

Which *simply* means the following: this line was saved into the buffer on 6th December 1993 at 4:37:24 am. The resistance value was 100.06Ω, the input switch was in MEAS position, channel 0 was selected, range was 3 (200 Ω), excitation was 7 (3mV), the display selector was in "R" position and no overload was detected.

These are converted to the "spreadsheet" form in a for-next loop. This loop operates as follows:

A new complete line from **VISA Read** enters into the shift register of the for-next loop. The "Match Pattern" function searches for the first occurrence of either a comma (,) or semicolon (;). Once found, it places the part before the search string into a string concatenating block. The part after the search string is placed into the shift register where it replaces the input to the Match Pattern function.

The string concatenate block consists of three elements. The first element is an empty string in the beginning. The middle element is the output from the Match Pattern. The third element is a space character (\s) except when the loop ends.

The first of the two shift registers handles the rest of the line, which shrinks after each turn.The second shift register gathers the converted string, where the commas and semicolons have been replaced by blanks, and the contents of this shift register grows after each turn. After 13 turns (the turns count goes from 0 to 12), the second shift register outputs its contents to the Write File function, which appends the new line to Filename. Now the next data line can be read from the interface.

There is no default for the Path and File names, so you have to specify them according to your computer and application.

In normal operation you should always delete the buffer file after having read it. The possibility to read and save without deleting exists for the purpose of debugging and developing the application. It would be very inconvenient to generate new buffer files frequently, because the only way to do so is to start new scan cycles.

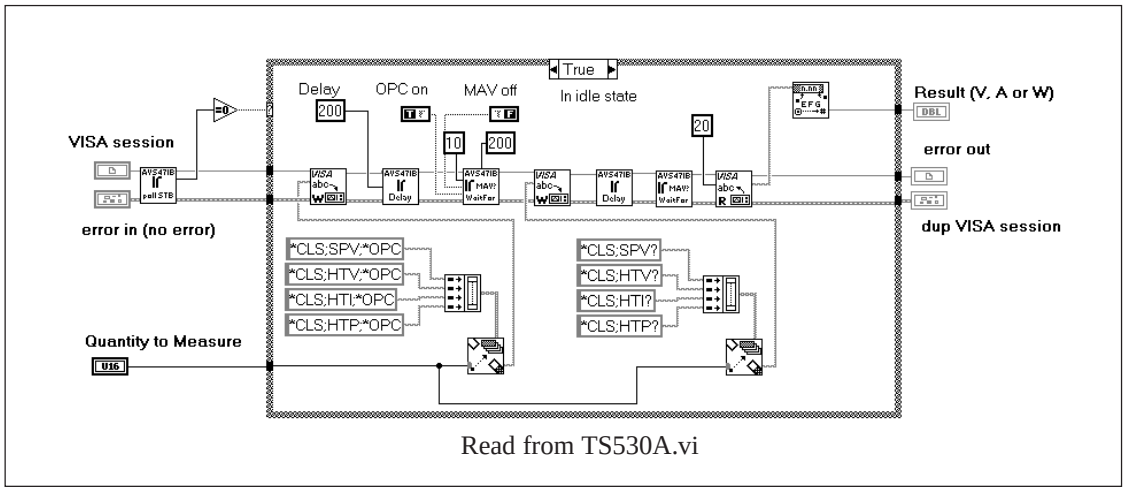
The command used for deleting the buffer file is KIL. A deleted file cannot be recovered.

6.7 Read AVS47 parameters

The **Read AVS47 parameters.vi** is used to determine the positions of all the AVS-47 front panel switches.These positions can then be used as initial values for later remote control.

The AVS47-IB requires one Picobus transaction in order to ask one parameter. This VI queries routinely all 8 parameters and therefore it is quite slow. It should not be used repeatedly in an application.

National Instruments seems to think that a LabView driver does not need VIs for reading any settings, because these should be known by the computer in a remote control application. However, we have designed this driver so that an application can be started without disturbing a running system. This is possible only, if the current state of the system can be read.



Read from TS530A.vi

The **Read AVS47 parameters.vi** gives the following outputs, which all are integer numbers:
- **Remote: 0..1:** Local/Remote control mode
- **Input: 0..3:** Zero/Meas/Cal
- **Channel: 0..7**
- **Range: 0..7:** no range, 2Ω...2MΩ
- **Excitation: 0..7:** no exc, 3µV...3mV
- **Display: 0..7:** R, ΔR.....530 set pt
- **Ref voltage source: 0..1:** from memory, from helipot
- **Deviation magnifier: 0..1:** ΔRx1, ΔRx10

All these items are separate outputs, so you can select which of them to use. **Getting started.vi** shows how the instrument state can be read and turned into initial conditions for later remote control.

6.8 Read from TS530A

The basic design of the TS-530 Temperature Controller dates back to a time where the computer control was limited to few aspects only. Therefore the TS-530A does not contain an A/D converter of its own. Voltages that are proportional to the heater output current and voltage, plus the analog set point voltage generated by the set point DAC, are all fed to the AVS-47 resistance bridge. The A/D converter of the bridge can then be used to measure these quantities.

The AVS-47 display selector positions 5-7 exist for this purpose: The set point voltage can be directly measured and the value displayed by the AVS-47 is correct. The heater current and heater voltage displays are not correct as such - they need some calculations which take into account the power range setting.

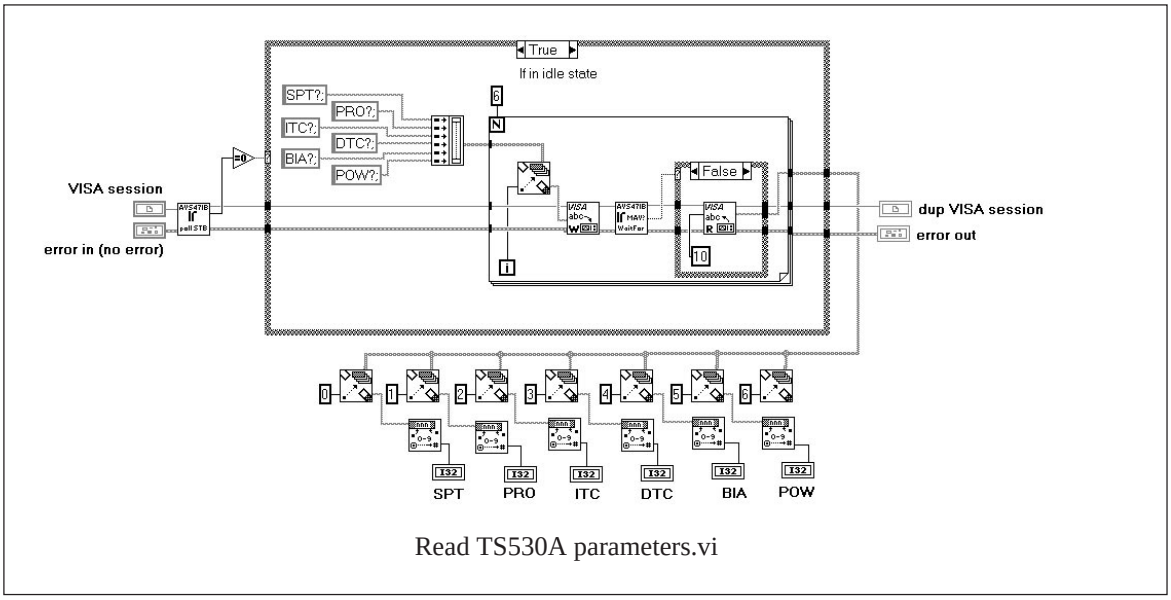
While the AVS-47 display cannot be directly used for measuring the heater current or voltage, the AVS47-IB interface makes all the necessary calculations. In addition, it offers also a query for the heating power.

The **Read from TS530A.vi** uses the macro commands offered by the interface unit. Because these macros need a couple of A/D conversions with settling times between them, their operation is very slow, typically a couple of seconds per query. Therefore this VI was not designed to provide answer to all four queries automatically. You must first specify the input parameter “Quantity to Measure”, which can be

- Set point (0: default)
- Heater voltage (1)
- Heater current (2)
- Heater power (3)

The floating point output is either volts, amperes or watts depending on the input parameter value.
The VI operates as follows: After having verified the idle state, measurement of one of the four items is commanded by using a string that begins with *CLS for clearing the status and ending with *OPC for generating the OPC bit once the measurement is complete. The “Index & Append” function is used to select one of the command strings.

A delay follows the VISA write, so that the interface has time to reset its OPC status before LabView starts to poll it. The **Wait for OPC/MAV/TMO.vi** loop is set for OPC, timeout of 10 seconds and loop delay of 200 milliseconds, which should allow ample of time for multitasking.



Read TS530A parameters.vi

Once the OPC status has been detected, the corresponding item can be queried. Note that these macros are always used in pairs, like SPV;SPV? - a measurement command and the corresponding query. The status is cleared again, and this time the waiting loop, in its default condition, waits for the response to become available (MAV).

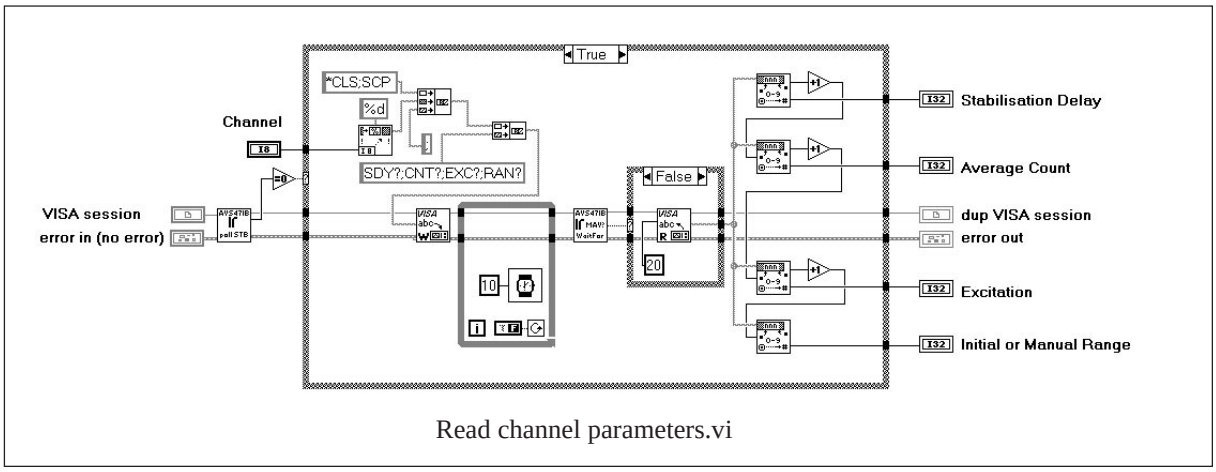
Application example 1.vi shows one possible way to use this VI.

6.9 Read TS530A parameters

It is not possible to read the PID parameters from the TS-530A in the same way as they can be read from the AVS-47 resistance bridge. This is due to the old basic design of the controller. However, it is possible to read the PID settings from the AVS47-IB interface. Although not exactly as useful, also the parameters stored in the interface can serve as good initial values for later control. Generally, these settings are the same as those in the TS-530A, except in a case where none of the parameters has been remotely programmed after they have been changed manually.

The **Read TS530A parameters.vi** is similar to the read VIs for the AVS-47 and for the AVS47-IB parameters. An array of strings is first formed from all the necessary queries, in this case SPT?, PRO?, ITC?, DTC?, BIA? and POW?

The for-loop that follows selects one of the queries in turn and sends it to the interface. Once the response is available, which is verified by the WaitFor VI, it is read. Note that the for-loop in LabView works so that if a turn of the loop produces some output, these outputs are collected together into an array. When the loop finally exits, it is this array that is passed as output. Therefore, the output needs no parsing, we have used several Index & Append blocks in parallel to extract each element.



Read channel parameters.vi

6.10 Read AVS47IB parameters

The **Read 47IB parameters.vi** works exactly in the same way as the previous VI, except that the queries are now PRN?, DSK?, IBA?, PBA?, PBD?, HDR? and FSM? corresponding to
- printer enabled
- buffer storage enabled
- GPIB device address
- Picobus address
- Picobus delay factor
- message headers included
- front panel started measurements enabled

Note that asking the IBA or PBD does not increment the EEPROM write count.

6.11 Read channel parameters

The SCAN procedure uses separate sets of parameters for each sensor channel. These parameters, which are called channel parameters, are the stabilisation delay, average count, excitation and initial range. Channel parameters can be read for one channel per one instance of the **Read channel parameters.vi**. Reading is possible if the system is in idle state.

The command string that is sent to the interface is

*CLS;SCPx;SDY?;CNT?;EXC?;RAN?

*CLS resets the status byte. SCPx selects the desired channel number x, where x is the input to this VI. Then the four parameters are queried.

The response is read once it becomes available. It consists of four floating point numbers, which are extracted using the “from decimal” LabView function.

You may need this VI only for creating correct initial values for the LabView application by reading them from the interface. Whereas the standard range and excitation values can be changed from the AVS-47 front panel in the local control mode, it is not possible to change the channel parameters manually.

6.12 Read scan parameters

The **Read scan parameters.vi** is exactly similar to **Read TS530A parameters.vi**, so please refer to its description. The items that are queried are 1) the first scan channel, 2) the last scan channel, 3) scan interval, 4) temperature control enabled, 5) temperature control period.

You may need this VI only for creating correct initial values for later LabView control.

7. UTILITY VI'S

According to NI's definition, Utility VIs perform a variety of operations that are auxiliary to those VIs of the instrument driver that are used mostly. An example of a utility VI is the **Reset.vi**.

This driver version does not contain a few VIs that should belong to an instrument driver if it is to meet the National Instruments' "standards". Please see "Omitted VIs" below.

7.1 Reset

The **Reset.vi** sends only one command to the AVS47-IB, namely *RST. This command can be sent to the interface regardless of its state with the exception of the self-test which cannot be terminated.

Using the **Reset.vi** will bring the AVS-47 and the TS-530A into their reset states. Most data in the AVS47-IB interface will remain intact, including the self-calibration data and scan parameters. The buffer file is not deleted.

Although **Reset.vi** exists as a mandatory option for the **Initialize.vi**, it has not been used in this driver because of its catastrophic effect on a running system.

7.2 Default setup

The **Default setup.vi** sets the AVS47-IB into a state where it is compatible with this driver. The command string is:

FSM0;HDR0;*ESE1

which means that front panel started measurements are disabled and so are also response message headers. Bit 2^0 (the Operation Complete Bit OPC) of the Event Status Register ESR is OR'ed (enabled) into bit 2^5 (the Event Status Bit ESB) of the STB status byte. This is accomplished by programming the Event Status Enable mask register ESE using the *ESE1 command. OPC is the only event that is enabled into the ESB bit. Therefore this bit tells whether an operation is complete, whereas bit 2^4 (the message available bit MAV) is permanently enabled.

These arrangements belong to the IEEE-488.2 standard. If you want to know more about how the AVS47-IB reports its status, please refer to the AVS47-IB manual.

The **Default setup.vi** is used only in the **Initialize.vi.**

7.3 Self test

The **Self test.vi** initiates a thorough investigation of the system. Please refer to the AVS47-IB manual for the detailed description of the self test.

The **Self test.vi** goes through the Picobus connection, the remote controllability of the AVS-47 and the remote controllability of the TS-530A. Further, it checks the offset and scale factors of the AVS-47 and of the TS-530A set point DAC.

In order to use the self test, you need only to connect the **Initialize.vi**, the **Selft test.vi** and the **Close.vi**. There is no verbose output report - only a number is output - but this report can be found in the interface manual.

The self test needs some minutes to complete. If you do not have a temperature controller in the system, the possible results will always be greater than 64.

*Note that one cannot use the **Discuss with AVS47IB.vi** for starting the self test query. This is because **Discuss** has a 1 second timeout whereas the self test takes some minutes.*

7.4 Self calibrate

The **Self calibrate.vi** makes a digital calibration of the AVS-47. The zero offsets and scale factors are measured in the ZERO and CALIBRATE input modes for all seven excitation ranges. These calibration factors are used to correct the following readings

- resistance, ΔR and ΔRx10
- average (also in the scan mode)
- maximum (also in the scan mode)
- minimum (also in the scan mode)
- standard deviation (also in the scan mode)
- digital filter reading

The A/D conversion result is not modified, neither are the measurements from the TS-530A. **Note that digital calibration does not change the display of the AVS-47.**

The **Self calibrate.vi** has three operating modes

0 (default) Query the self calibration factors. This produces a string output, consisting of seven offsets and seven scale factors separated by commas.

1 Reset self calibration factors. All offsets are set to zero and all scale factors are set to 1. After this operation (or if self calibration has not yet

been conducted) the output displayed by the AVS-47 and the digital results from the interface are the same.

2 Self calibrate.

The self calibration factors are stored in the AVS47-IB as long as it is powered. If you need to check whether the calibration still exists, you can check the offset or scale factor corresponding to the 3µV excitation. Because of noise, these readings are sure to deviate from exact 0 or 1.

Use self calibration if the AVS-47 is subject to large changes in the ambient temperature. Digital calibration is faster and easier than the analog calibration which requires opening the instrument and adjusting a number of trimmers.

You can also use digital calibration to calibrate the deviation mode ΔR. This is because the calibration procedure handles the deviation reference simply as an offset.

8. OMITTED VIs

Some VI's have been omitted from this driver, although they should belong to an instrument driver, according to National Instruments "standards".

Revision Query.vi

The revision number and other necessary information can be found in "Show VI info" for the **VI-Tree.vi**.

Error Query.vi

Many virtual instruments in this driver contain error flags for "not in idle state", "timeout", and sometimes also for "overload". If the interface was not in idle state, the rest of the VI is usually bypassed. The VI front panels contain error indicator LEDs, which can be inspected by opening the VI.

Once the application has been designed and debugged, error situations should be rare.

Error Message.vi

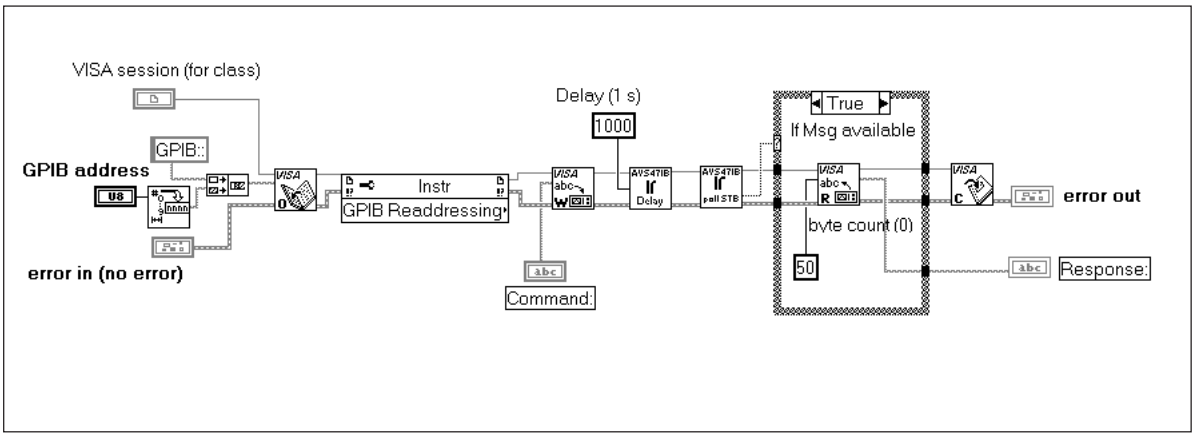
Was not implemented because no error query exists.

9. APPLICATION EXAMPLES

The previous discussion of the virtual instruments concentrated on how these VI's operate and what they actually do. Sometimes the discussion was quite complicated, because the operations were complicated. Very little was said about how to use these VIs simply and effectively, and this is just the purpose of LabView.

We have designed a couple of application examples, or "instrument front panels" if you want to say it that way. These examples should give some idea on how we thought our driver could be used. We have also played with these "non existing instruments" and they seem to work in a way that we think is correct.

Even though you may want to write your application yourself by using the more basic VIs, please take a little time, read through the descriptions and try some of the examples before starting your work.



9.1. Discuss with AVS47IB.vi

This is not actually an application example at all. The **Discuss with AVS47IB.vi** is a simple VI that you can use for sending commands and queries, single or combined, to the AVS47-IB interface and to read the response if such is generated.

You can use this VI when you start to use our interface with LabView for the first time. Some operations, like changing the GPIB address or asking the firmware version, are easier to make using **Discuss** than if you write a VI, which may be never needed again. **Discuss** provides also an easy way to study the behaviour of the AVS47-IB + AVS-47 + TS-530A system.

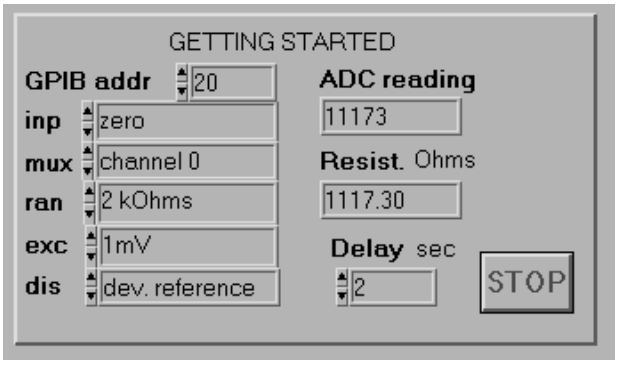
The VISA session is opened at first. The resource name is GPIB::xx, where xx is the GPIB device address that you give from the front panel. The factory default address of the AVS47-IB is 20.

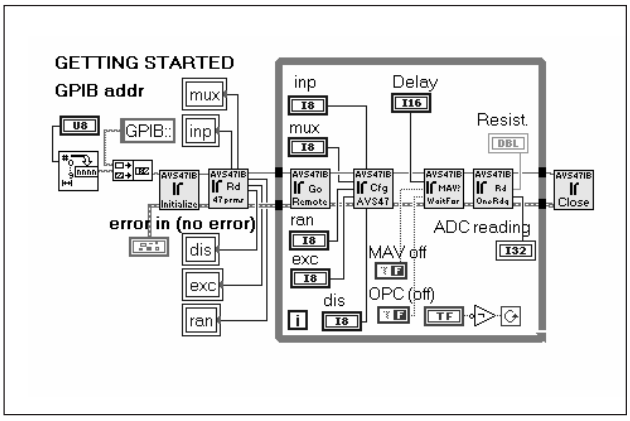
> *You can change the GPIB address once you have first established the connection using the old address. Then issue the command "IBA newaddress". The response to the "IBA?" query has changed to "IBA newaddress", but the interface will still use the "oldaddress" as its GPIB device address. Boot the interface by plugging the power off and on again.*
>
> After booting, the new address becomes valid everywhere, and you must change also the GPIB address on the Discuss front panel.

There is a "property node" after the VISA open function. It seemed to us that some error conditions were avoided by adding the "GPIB readdressing=default=true".

The command line, which may consist of several commands and/or queries separated by semicolons, is written to the interface. A delay loop prevents polling of the status until one second has passed. If a message is available, it is read and placed to the response screen on the front panel. If there is no message, an empty string is placed to the response screen.

If everything works, this VI is easy to use: Open it, write the message, run the VI and read the response.





9.2. Getting started.vi

The **Getting started.vi** shows one way to use the virtual instruments in this driver to build an easy-to-use resistance bridge without any bells and whistles.

The resource name for the **Initialize.vi** has been generated from the header string and the GPIB address that is obtained from the front panel. The integer number is converted into a string using “To decimal” function, whereafter the header and the address are added using the string concatenate function.

The **Read AVS47 parameters.vi** queries the settings of all AVS-47 front panel controls. The outputs are placed in local variables having names like “inp” or “mux”. This is the way how one can read the existing setup of the system into LabView.

After the settings have been saved, the program enters into a while loop, which is run until the “STOP” button is pressed. At first, the system is set into remote mode. You can see that the yellow “remote” led on the AVS-47 front panel lights. Actually, the **Go remote.vi** could have been before the while loop as well.

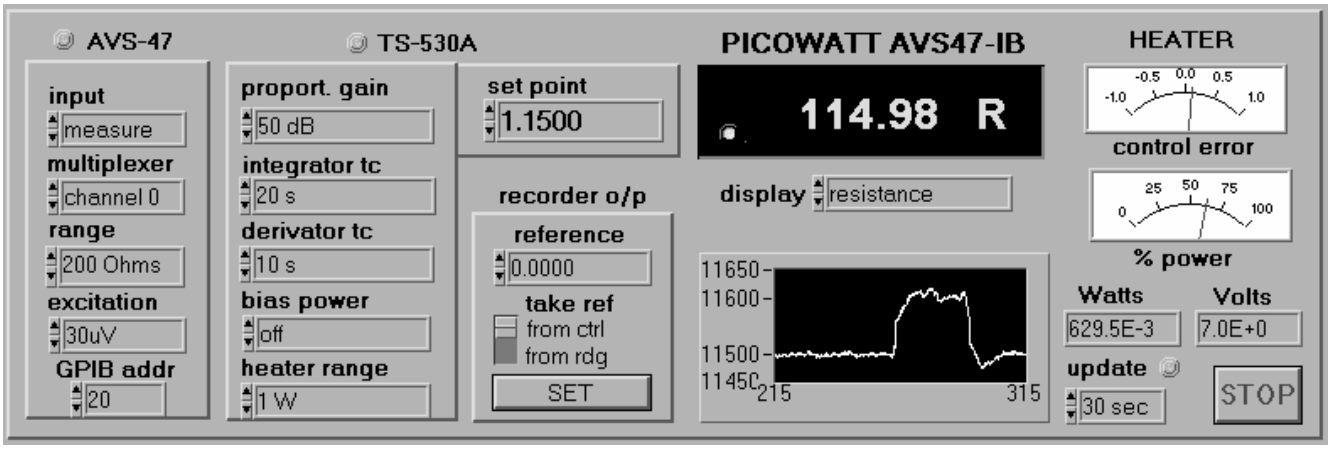
The AVS-47 is now configured. All the real switches on the AVS-47 front panel have their counterparts as digital ring controls. These ring controls have a verbose name for each digital number, which makes their use easy. When the while-loop is run for the first time, values for these ring controls are obtained by copying them from the local variables having corresponding names.

In fact, LabView has supported creating such local variables. Using the wiring tool, point the digital control “inp”, for example. Right click, and you can find a menu item Create/local variable.

Once the controls have so got their initial values, you can start to change them. The local variables are used only once, so your changes remain valid as long as the while loop runs.

The **Wait for OPC/MAV/TMO.vi** is configured to produce a programmable delay. You can experiment with this delay. Set it to zero, the excitation to 10 µV and switch between input=0 and input=100R calibrate. If the delay is zero, you get many readings before the bridge has settled. If the delay is long, the bridge has time to stabilize but the system is slow. The purpose of this delay is just to show that you need a delay after the settings have been changed, not elsewhere.

The **Take one reading.vi** outputs both the resistance and integer ADC values.



9.3. Application example 1.vi

This is a more complicated VI, which suggests how the lower level VIs can be used programmatically to create a new non-existing instrument. Some useful hints may be found from this VI; also one peculiarity of LabView is discussed.

The **Application example 1.vi** combines both the AVS-47 and the TS-530A into a single precision temperature controller for dilution cryostats. It can also be run without the controller, but not without the bridge. This VI offers only manual ranging. Further, you must select the control polarity of the TS-530A manually - it cannot be remotely controlled.

The AVS-47 switch panel on the left side has controls for the input mode, channel, range, excitation and for the GPIB device address. Control for the display item has been moved under the LED display.

The TS-530A switch panel offers controls for the proportional gain, integrator and derivator time constants, bias power, heater power range and the set point in volts.

The “recorder o/p” panel contains input for the deviation reference of the AVS-47 and a pushbutton “SET” switch. In addition, it has a slide switch selecting from where the reference is obtained: either from the digital control above or from the LED display.

Note that the “analogue” output is always proportional to the sensor resistance, you can affect the “recorder” output by setting the deviation reference.

The two analog meters for heater power and error signal, familiar from the TS-530A, have been retained. What is extra to the physical controller, is that also the heater current and voltage are calculated and displayed.

Because measuring the heater power etc. is a slow process, we have provided some choices how often this is made. This is the purpose of the “update” control.

Although multiplexing is possible, its best use is in applications not involving a temperature controller. By shorting the error signal (see proportional gain control) and by latching the integrator (see integrator time constant), you can isolate the controller from the bridge for a while, and measure another sensor. Doing this manually is prone to catastrophic errors, therefore one should build a new application that makes advantage of the SCAN procedure.

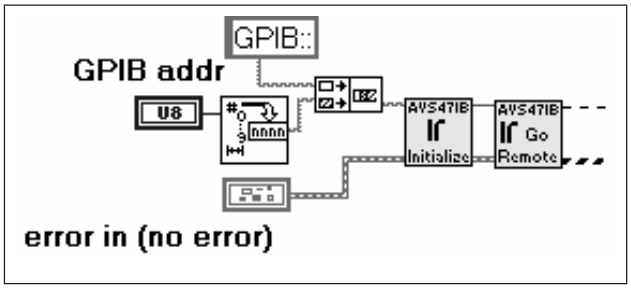
Use of the **Application example 1.vi** is almost intuitive to those who have previously worked with the physical AVS-47 and TS-530A. Assume that the control system is ready with a sensor and a heater and a thermal path between them.

After having configured the AVS-47 properly for the control sensor, the reading should stabilise to some reasonable value. Set the reference to a value that requires some heating. The “control error”- meter deflects to positive direction, if more power is needed. Set the “update” to 10 seconds. Select a suitable heater power range. The value depends on situation and can be something from microwatts to one watt. Increase the proportional gain until the power meter has risen a little.

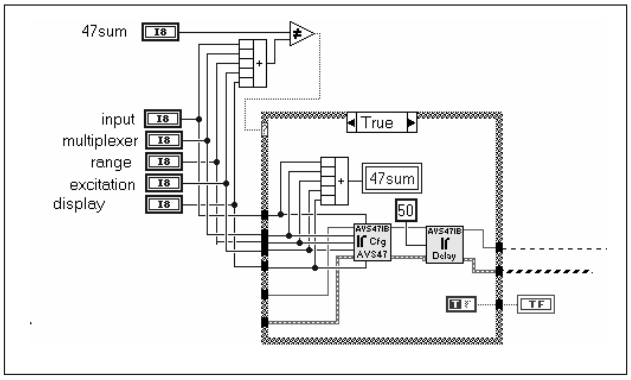
The power meter is updated in every 10 seconds. Once you have found reasonable settings, make the update rate slower, otherwise most of time the system measures the power and not the sensor value at all. As the system approaches equilibrium in the PD mode, enable the integrator with a suitable time constant.

The “History” screen is set for automatic Y-range and the chart history lenght is 1024 points, dx is 0.1 and the update mode is “strip chart”. You must tailor these settings according to your own needs with the help of the LabView manual or online reference.

The diagram is a little complicated. Please open the VI in LabView so that you can better follow this discussion.



After initialisation, the VI forms a big while loop, which is run until the STOP button is pressed. This discussion follows the VISA session "thread".

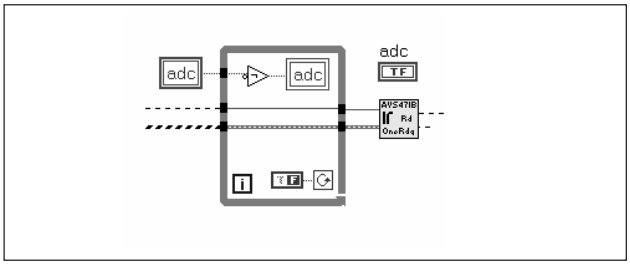


At first the program calculates the sum of all AVS-47 front panel controls. This sum is compared with a hidden control "47sum". If these are the same, it means that none of the VI front panel controls has been changed, and the **Configure AVS47.vi** can be bypassed. If they are not the same, the program enters a case statement. Inside the case statement, a sum of all AVS-47 controls is calculated again. This sum is different to the above mentioned "47sum". The new sum is now stored in a local variable, whose name is also "47sum". When the program exits this case statement, the value in the local variable is stored in the hidden control "47sum". This is a way to prevent configuring the AVS-47 if none of the settings has changed, and to speed up operation. This **Application example 1.vi** does not use reading initial values from the instruments.

In the next phase, a quite similar case loop configures or does not configure the TS-530A, depending on whether the "530sum" has changed since last comparison.

Of course it is possible to cheat this method by toggling one control up and the other control down so that the sum remains constant. However, it is not easy to do this unintentionally with a mouse in a time less than one cycle.

The deviation reference is set if the SET button is pressed. **Configure reference.vi** needs two inputs, the reference mode (from reference, from display) and the reference value.

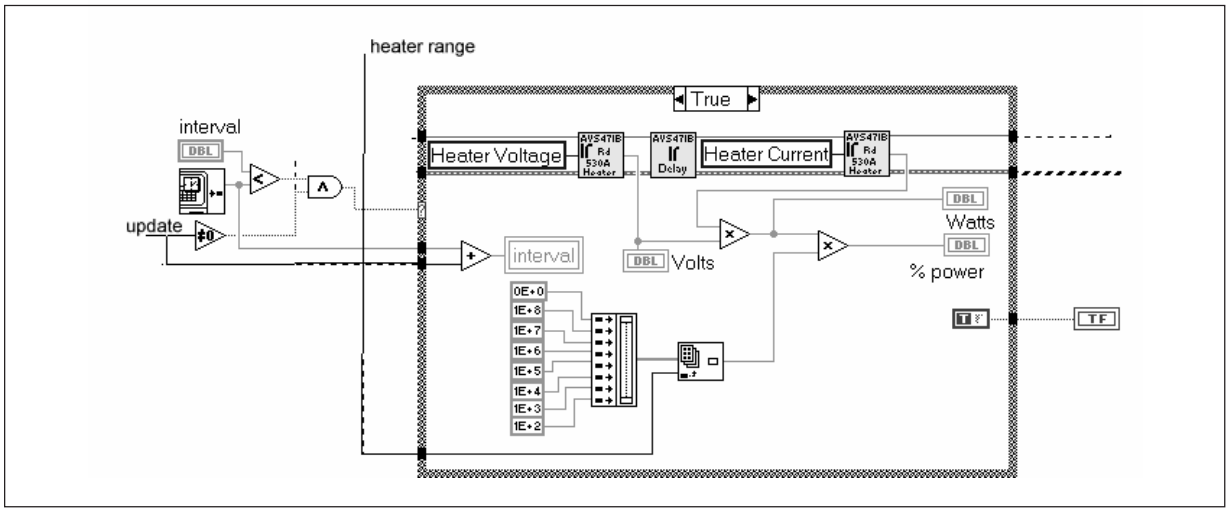


The next small while loop is run only once, because the while-control is always false.

Inside this loop there is one inverter and a *write local variable* "adc". The inverter gets its value from the *read local variable* "adc" outside the loop. The values stored in these two local variables are Boolean, because they have been created from the ordinary Boolean indicator "adc". The write local "adc" gets its value inverted from the read local "adc", and the value in the write local is written to the read local immediately after the case statement has been ended. The effect is that the values oscillate between true and false. The same oscillation is seen in the Boolean indicator "adc" which is the round yellow LED lamp on the left of the display. It changes state on each turn and indicates that the big loop is running, which would otherwise be difficult to see.

One A/D conversion reading is obtained using the **Read one reading.vi.** The integer ADC reading is fed to the "history" chart display. Refer to "Charts" in the LabView manual.

The ADC result is also compared with the TS-530A set point. The difference, which is the control error, is subject to nonlinearising before showing it on the error meter. The error signal is converter to a positive value, 1 is added to prevent it from being zero, and a logarithm is taken. The original sign of the error signal is returned and the result is scaled to suit the meter. The meter is updated in every turn if "update" is not zero. The purpose of this nonlinearising is to prevent the meter from saturating at high control errors while keeping it reasonably sensitive near to equilibrium.



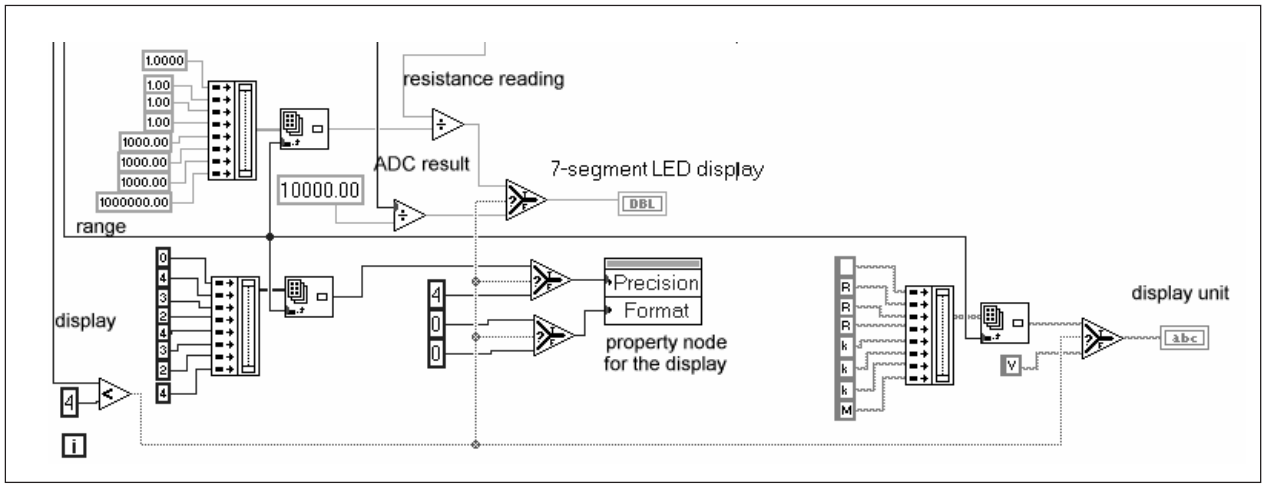
From the **Read one reading.vi** the VISA session goes into a CASE statement, which serves for calculating the TS-530A heater output. Because measuring the heater output is a very slow procedure, one has to make it only seldom. Otherwise there would remain no time left for measuring the sensor itself, which usually more important than knowing the exact output level.

How often the output is measured is determined by input called “update”. If “update”=0 (“no 530”), the case is identically false. If “update” is not zero, and the current clock time of the computer is equal or greater than the value in the hidden control “interval”, the case will be true and the statement will be run. One operation within the case structure is to add “update” to the current clock time and to store it into a *local write variable* “interval”. When the program exits the case structure, the value from the local variable is copied into the hidden control “interval”, which will now represent a new moment when the case will become true again.

The heater output voltage and current are read using the **Read from TS530A.vi** two times. The output power is simply the product of these two values. In order to present the power in terms of % of the full range, one has to scale the actual power. This was made by storing coefficients for each range as array elements and by using the “heater range”-control for selecting the element.

One of the most difficult things in writing this VI was to generate the display. For some strange reason, the writers of LabView have understood the word “precision” so that it means the number of digits after the decimal point. According to this thinking, 0.01 and 100000.01 woud be equally precise.

The A/D converter of the AVS-47 gives 4 1/2 digits, from 0 to 19999. The precision of the reading is limited by the number of significant digits. We wanted the

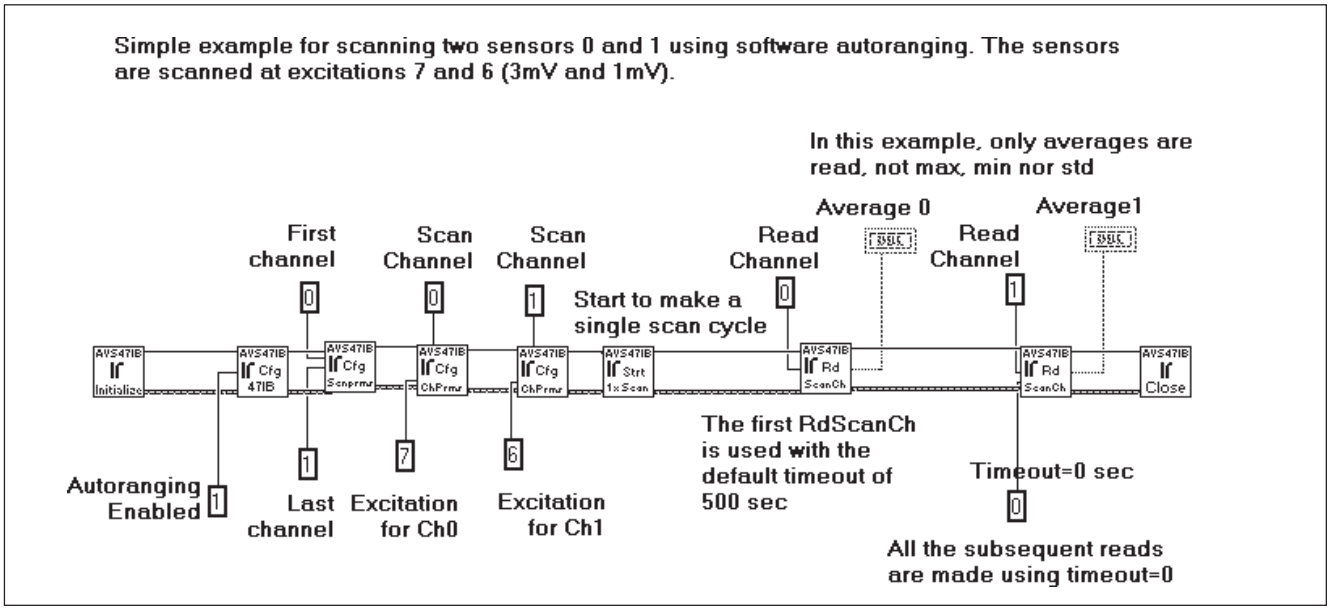


display to indicate these digits, no more or no less regardless of the range setting.

There are 8 possible items to be displayed. First four of them (0..3) can always be expressed in terms of resistance, whereas volts is more suitable for the rest of them. Although the ADC reading could have been used to generate the display with the aid of the range selector, we have used also the floating point R reading, because it takes into account the possible ΔRx10 switch.

R is first divided by 1, 1000 or 1E6 so that it is suitable to be expressed either in Ohms, kilohms or Megaohms. The result is led to the display, whose format is determined by the property node. As long as the display item is less than 4, the number of digits after the decimal point ("precision"?) is taken from an array. For display items from 4 up, this number is fixed to 4.

Last, there is a units indicator letter right of the display. This letter is "R", "k" or "M" for Ohms, kilohms and Megaohms respectively. If the display item is not resistance, the display unit will be "V" for volts.



9.4. Scan2Ch.vi

This is a simple example showing how to start a scan cycle for two sensors and how to read their results.

Autoranging is enabled, and no initial ranges are given. This means that when this VI is used for the first time, the AVS47-IB starts from the 2MΩ range and autoranges as many times as needed to get a proper reading. The correct ranges are rememebered, and the next scan cycle should not require this iteration.

Only channels 0 and 1 are measured. Channel 0 is measured using excitation 7 and sensor 1 using excitation 6. Replace the constant with digital controls for any practical application.

When the single scan cycle is started, the status byte output of the AVS47-IB is nulled. The OPC bit will be set once scanning has ended.

The RdScanCh polls the status until it sees the OPC bit. Then channel 0 is read. In order to read sensor 1, the RdScanCh (=Read Scan Results.vi) is repeated with the timeout control input set to 0. This is the important thing to remember.

INDEX

Symbols

A

B

C

D

E

F

10. Revision History

1R0 => 1R1

Effect of the “system decimal point” on reading operations from the AVS47-IB was removed by forcing LabView to use period regardless of the setting in Windows.

The Wait for OPC/MAV/TMO.vi was modified so that if the timeout input control is set to zero, no timeout error output is generated.

A simple example on scanning two sensors “Scan2Ch.vi” was included in the driver set of files.