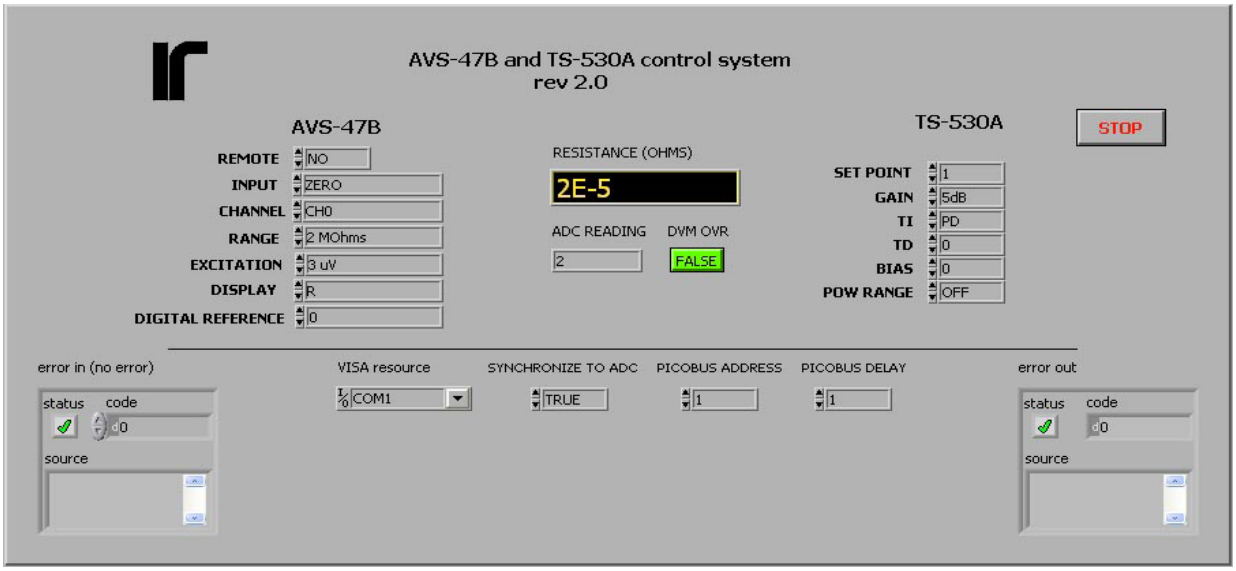
APPLICATION NOTE

LABVIEW VI:s

rev 2.0



For connecting the AVS-47[A,B] to a PC via the Picobus Primary Interface

written for LabView version 7.1

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

CONTENTS

1. INTRODUCTION

The **AVS-47B AC Resistance Bridge** has a very simple "primary computer interface" called Picobus, which is used for connecting the bridge to a PC either via the optional **AVS47-IB Two-Stage GPIB Interface** box, or directly, without any options and additional costs. Direct connection is limited to PC type computers running **LabView 7.1** or higher and having a free standard COM1: or COM2: serial port. On the contrary, the AVS47-IB box relies only on the **IEEE-488 standard** and is thus less dependent on computer types, operating systems or application software. While the AVS47-IB is supported by a very extensive set of GPIB commands (**IEEE-488.2**) and a full-featured **LabView Driver**, direct connection is supported only by a set of less formal **LabView VI:s ("Virtual Instruments")**. Together with some programming examples and this application note, these VI:s should make direct interfacing accessible also for a less experienced LabView owner.

Direct interfacing can be a cheap but suffient solution if

1) The user has an "IBM type" PC computer
2) The computer runs LabView 7.1 or higher
3) The computer has a free COM1: or COM2: serial port with full hardware handshaking capability
4) One wants to avoid using GPIB, either because it is noisy, because the PC does not have an GPIB controller card, or because the distance is too long for GPIB.
5) Advanced features of the AVS47-IB are not needed
6) Saving money is important.

National Instrument's LabView (abbreviated by **"LV"**) software seems to be now popular enough, and we believe that it will continue its success and good support also in the future. Therefore we selected LabView, and did not even try to guess any other programming language that would have an equally good installed base among our customers.

This set of VI:s is not a commercial product. It is given for free, without any kind of warranties, and the VI:s can be freely modified by the customer to better suit his/her needs. These VI:s and this application note may contain errors, and we would be glad to get feedback , corrections and suggestions for improvements. You can download the VI:s from our WEB site at www.picowatt.fi.

2. WHAT IS PICOBUS

2.1. HISTORY

When the AVS-47 design was started in 1993, minimizing radiated interference was one of our primary concerns. The type **DC900 computer interface option** of the **AVS-46**, our previous resistance bridge model, had been found quilty of emitting some high frequency pollution. Therefore we decided to divide the computer interface of the new AVS-47 into two sections. The first section, called **"primary interface"**, had to be designed inside the resistance bridge, so it had to be made as silent as possible. As the second section, we designed an isolated, self-powered, external protocol converter that can be located far away from the bridge (**"Secondary Interface"**). Thanks to the isolation barrier, long physical distance and optional optical fibre link (**"Picolink Option"**), this concept of "two-stage" GPIB interfacing can solve many EMI problems.

In order to be noiseless, the primary interface had to be implemented without any digital intelligence. This prevented the use of an asynchronous protocol, like RS232. Instead, we developed a new synchronous, serial protocol. This **Picobus protocol** uses a PC:s **COMx: port** - not its data lines but the four handshake lines, two of which are inputs and two are outputs. Any program that uses Picobus must be able to access these handshake lines indepently of each other.

When the AVS-47 was launched, direct connection to the PC's COMx: port was supported by some **Turbo Pascal** and **Quick Basic** programs and library routines. But when the **DOS operating system** was replaced by the never-ending succession of new **Windows** versions, those pieces of software became useless. Moreover, direct connection was quite complicated, requiring more than basic programming skills and a lot of time and effort. Consequently, we ceased to support direct interfacing, but offered the AVS47-IB and its LabView Driver as the only solution.

National Instrument's LabView software offers a method, called **"property nodes"**, to access the individual handshake bits of a COM1: or COM2: serial port. This fact inspired us to rehabilitate direct interfacing - but now only for LabView users. The VI:s that are described in this Application Note (abbreviated by "**AN**") enable remote control of both the AVS-47B and **TS-530A Temperature Controller**. Because they do not constitute a complete "**Instrument Driver**" according to NI's standards, we call them just a "**Set of VI:s**".

DISCLAIMER

RV-Elektroniikka Oy Picowatt makes no representations or warranties with respect to the contents or operation of the **Virtual Instruments (VI:s) included in this Set of VI:s**, and specifically disclaims any implied warranties or fitness for any particular purpose. **Picowatt** shall under no circumstances be liable for incidental or consequential damages or related expenses resulting from the use of these VI:s, even if it has been notified of the possibility or danger of such damages. **Picowatt** reserves the right to revise these VI:s from time to time without obligation to notify any person or institution of such revisions, and without obligation to maintain any degree of compatibility between the revisions.

2.2. DESCRIPTION

Picobus Primary Interface is a synchronous, serial interface for the AVS-47B Resistance Bridge. Although it makes use of a PC computer's serial port (usually COM1: or COM2:), it does not follow the standard asynchronous RS232 transfer protocol. Instead, Picobus depends solely on the status outputs and inputs of the serial port. These lines are used in a special way in order to implement a strictly synchronous and bullet-proof operation. Main advantages of the Picobus protocol are:

* The interface does not contain a microprocessor nor any other noisy high-speed digital circuits, which could produce EMI heating in low temperature applications. This was the main reason to develop Picobus.
* Data transfer can be made as slow as desired. Usually, one wants to increase the speed of a protocol. When working with very low temperatures, however, RF filters may be necessary in wires that enter the shielded cryostat room. Fast digital signals cannot be filtered effectively, whereas slow signals can.
* Several AVS-47B bridges, each having a different device address, can be connected to a single COMx: port by extending the **Picobus cable** from one bridge to another. Contact factory, if you are interested in this possibility. This feature is not needed with GPIB, and it is therefore not described in any of our documents.

Unlike asynchronous interfaces, synchronous operation is not time-critical. Depending on software, Picobus may be interrupted for an arbitrarily long time. Further, Picobus can recover easily from errors like pulling off the cable. Synchronous logic never needs to be rebooted.

The output side of the Picobus Primary Interface section has its own power supply inside the AVS-47B and the interface is **optically isolated** from the bridge regardless of whether it is connected to a PC or to an AVS47-IB box (NOTE: the same does NOT apply to AVS-47 or AVS-47A)

The Picobus protocol consist of the following four signal lines:

Picobus signal name	RS232 signal name
Clock Pulse (CP)	RTS (Ready To Send)
Data From Computer (DC)	DTR (Data Terminal Ready)
Data From Instrument (DI)	CTS (Clear To Send)
Alarm (AL)	DSR (Data Set Ready)

The CP and DC lines are outputs from the PC and they are found in the **Modem Control Register** of the corresponding COM port. The DI and AL lines are inputs to the PC and they exist in the **Modem Status Register**.

2.3. TRANSACTIONS

Communications via Picobus are based on "**transactions**". One transaction consist of sending all programming information from the PC to the bridge and of reading the full configuration of the bridge plus the result of the latest A/D conversion from the bridge to the computer. Because these two things happen at the same time, the response from the bridge indicates the old state, that was effective just before the transaction. Information is sent across the Picobus signal lines as high and low voltage levels, corresponding to bit values of ones and zeros. As the number of bits is rather high, 48 in case of the AVS-47B, the bit stream is more conveniently understood as a string of 48 characters ("0" or "1") than as a 48 bits long binary number.

A transaction consists of four parts:

1) Send the **Picobus Address (PBA)** from the PC to the AVS-47B. The address string consist of 8 bits. PBA can range form 1 to 15. AVS-47B factory default is 1, and all resistance bridges have been originally set for this PBA. The address is determined by a DIP switch on the "AVS47E" circuit board. Only if a second bridge is added to the same Picobus line, the PBA of

that second bridge should be altered.

2) **Strobe the address**. Strobing tells to the AVS-47B that the computer has sent all the 8 address bits. If the received PBA is the same as the DIP switch address of the bridge, input and output ports of the AVS-47B primary interface are opened for receiving and transmitting the actual data. If the PBA is not recognized, the ports remain closed. No Picobus traffic whatsoever can then reach the bridge, and all bits that one might try to read out from it will be zeros.

3) **Send the data and read the response**. The configuration data is sent as a 48 bit long string, called "**TXSTR**" (transmitted string), and the received data is an 48 bit long string called "**RXSTR**" (received string or response string). In addition to the new programming data, TXSTR contains the new digital REFERENCE setting for the ΔR difference amplifier. In addition to the old configuration data, which corresponds to the front panel indicator lights, RXSTR contains the result of the latest A/D conversion in BCD format.

4) **Strobe the data**. Now strobing tells to the AVS-47B that all 48 data bits have been transmitted. Strobing data closes the input and output ports of the bridge, and they remain closed until a valid PBA is strobed and recognized for the next time. The new configuration comes into effect as soon as data has been strobed.

The Picobus address and configuration data are sent using the **Clock Pulse (CP)** and **Data From Computer (DC)** lines. The response is read using the same clock pulses and the **Data From Instrument (DI)** line. Data is clocked by the rising edges of CP. Depending on the speed of the computer and the application program, one transaction may take few milliseconds or longer. The drive capability of the handshake outputs is, however, limited, and if the signals are clocked too fast, they do not have enough time to settle. A programming concept of "PBD" (Picobus Delay) is therefore required. The simplest (although perhaps not the best) way to implement PBD are processor loops, which are inserted in a few places for creating delays between the rising edges of DC/DI and CP. The length of the required delays depends on the drive capability of the COM port, cable length and on possible filtering. A too short delay causes data transfer to be unreliable or to fail completely. A typical delay could be in the range of ten microseconds or so.

The number of available output-type handshake lines is two. One is needed for clock (CP) and the other for data (DC). Strobing the PBA and the data is therefore made by interchanging the roles of the CP and DC signals. The normally toggling CP line is set permanently to zero whereas the DC signal toggles up and down for three cycles. This situation will never happen in normal operation, and therefore it can be recognized reliably as the "**strobe condition**".

2.4. ALARM

The integrating dual-slope 7135 DVM circuit needs 0.4 seconds for each conversion. Completion of the conversion could be found by polling the AVS-47B continuously via Picobus, but because Picobus is slow, this would waste much of the computer's power. Therefore, the Picobus protocol includes a special **Alarm (AL)** signal line, which resembles the "**Service Request**" line of the **IEEE-488.2 standard**. The AL line is asserted when a conversion result becomes available, and it is reset by each completed transaction. Instead of polling the bridge via the slow Picobus, the program can now check the state of one bit in the Modem Status Register - the AL bit - which is a much faster operation. The AL signal also reduces problems with emitted interference by eliminating unnecessary traffic in the Picobus cable, .

By synchronizing transactions to the low-to-high transitions of AL, one can make sure that each A/D conversion result is read no more than once. This is also the practice suggested by these VI:s.

2.5. MORE ABOUT PICOBUS

Please refer to detailed descriptions of the **Picobus Utilities VI:s.**

2.6. PICOBUS CABLE

Before starting to make yourself familiar with the Picobus VI:s, you must have a PC-type computer running Lab-View 7.1 or later, and you must connect its COMx: port to the primary interface of the AVS-47B. The **factory supplied Picobus cable** has 25-pin DB25S and DB25P connectors, and it is intended for use between the AVS-47B and the AVS47-IB. For direct connection to any modern PC computer, the female DB25S must be replaced by a 9-pin DE9S. You can:

1) Ask us to mail you the required adaptor without charge

2) Buy a 9(female)-to-25(male) pin adaptor piece, e.g.
   Farnell 960-640 adaptor
   Farnell 960-573 0.3m long cable with connectors
   RS 218-273 adaptor (also other types)
   RS 243-0352 small adaptor

3) Replace the female DB25S connector of our standard cable by a DE9S-pin connector. Then you cannot use the same cable for connection to the AVS47-IB box any more. See the pin assignments below.

4) Build a new cable. The cable must have a female DE9S and a male DB25P connector. It must have a shielding jacket and at least 5 inner wires. Note that the shielding jacket is connected to the enclosure of the AVS-47B but not to the ground of the PC. Building a new cable is a good solution, if you want to have a very short or very long cable (in the latter case, you may need to increase the Picobus delay factor). You can also divide the cable in two parts and make a filtered feedthrough where Picobus enters the shielded room.

The Picobus cable is wired as below:

Computer end DE9S		**Interface end DB25P**	
Pin	RS232 signal	Pin	Picobus signal
Not connected		1	SHIELD Chassis gnd
7	RTS	4	CP Clock pulse
8	CTS	5	DI Data from Instrument
6	DSR	6	AL Alarm Status Line
5	GND	7	GND Signal Ground
4	DTR	20	DC Data from Computer

(The RS232 names are here only for help, if you want to monitor the bus operation with an RS232 tester.)

Prevent the Picobus cable from acting as an antenna by connecting its shield to the wall of the shielded room at the entering point. For that, remove a short section of the outer insulation.

2.7. DIFFERENCES BETWEEN THE AVS-47B, AVS-47A AND AVS-47

YOU CAN SKIP THIS PARAGRAPH IF YOU ARE USING THE AVS-47 "B" REVISION.

READ THE FOLLOWING TEXT CAREFULLY, IF YOU ARE USING AN AVS-47 EITHER WITH OR WITHOUT "A". FAILURE TO FOLLOW THE INSTRUCTIONS CAN LEAD TO POWER SUPPLY OVERHEATING AND POSSIBLE DAMAGE.

AVS-47, AVS-47A:

These revisions do not have an internal, isolated power supply for Picobus. Therefore, the required +5V power must taken either from the bridge, with the consequence that the interface cannot be isolated, or from an external floating power supply.

The **Picobus cable shield** is grounded to the resistance bridge by default. The ground connection can be opened by pulling the short yellow-green wire out from its socket J201 (Fig. 1. the "E" circuit board). While this eliminates ground currents flowing via the cable shield, it impairs immunity against electrostatic discharges. The shield must be grounded somewhere: either to the AVS47-IB via the 25-way D-connector, to the AVS-47(A) using J201, to the wall of the shielded room, or to a PC in case of direct interfacing. In the last case, one must check with an ohmmeter, whether the 9-to-25 way adapter (which is required for all modern computers) carries the shield ground or not. From the ground currents point of view, the Picobus cable should be grounded only at one point. The best immunity against ESD:s is obtained, on the contrary, when both ends are grounded.

A short-circuit piece must always exist in JP204. Whether the places JP201 and JP202 are also jumpered, depends on the application as below:

1) AVS-47(A) is connected to AVS47-IB: Short-circuit pieces must be removed. Signal lines are always isolated. Isolation of the cable shield is enabled by removing the ground wire from J201, and this can be done because the shield is connected to the AVS47-IB box. Place the short circuit pieces to single pins as shown by Fig. 2, so that they do not get lost.

2) AVS-47(A) is connected to a PC, no external power supply exists: JP201 and JP202 must be jumpered with short-circuit pieces, Fig. 3. Optical isolation of the signal lines is not possible. Decision whether to ground or not the cable jacket to the bridge is made solely by considering ground currents. The shield must remaing grounded at least at one point.

3) AVS-47(A) is connected to PC and optical isolation is desired: A floating, stabilized, external +5V power supply, capable of delivering at least 200mA, is connected between pins 11 (+5V) and 7 (0V) of the Picobus DB25 connector. You can connect these voltages to the existing DB25P connector of the Picobus cable, Fig. 4. Alternatvely, if the **battery-power option** for the AVS-47(A) is not needed, you can pull off the 4-way connector J403 (three wires, to the left of the mains transformer), and put it to the 2-way connector J203 in the way shown by Fig.5. Use pins 1 and 2 of the 4-way rear panel DIN connector ("BATTERY INPUT") for +5V and 0V, respectively. Please note that the Picobus power input is NOT protected against wrong polarity or overvoltage. Both will destroy the output side of the Picobus circuitry.

If the shield of the Picobus cable is grounded elsewhere, detach the ground wire from J201. If the shield floats, provide one grounding point by keeping J201 inserted.

If you later decide not to use direct interfacing any more, please remove jumpers from JP201 and JP202 and store them as in Fig 2. Power supply either in the AVS-47(A) or in the AVS47-IB can suffer from serious overload, if these two devices are connected together while the jumpers exist.

AVS-47B (no action is usually required):
The current bridge revision, AVS-47B, has an isolated DC/DC converter that supplies power for the output side of the primary interface. The four Picobus lines are optically isolated regardless of whether they are connected to a PC or to an AVS47-IB box. A short-circuit piece exists in JP204 whereas places JP201-JP203 must not be jumpered (AVS47E circuit board). **By default, J201 is not inserted in order to enable full optical isolation between the AVS-47B and the AVS47-IB box.** In order to prevent the cable from acting as an antenna, the cable shield can be connected to bridge (beware of possible ground currents) or to the conducting wall of the cryostat room. If the AVS-47B is interfaced directly with a PC using such a 9-to-25 pin adapter that does not carry the protective ground, then J201 must be inserted. Some "E" circuit boards have a dummy connector J202 for storing the detached ground lead.

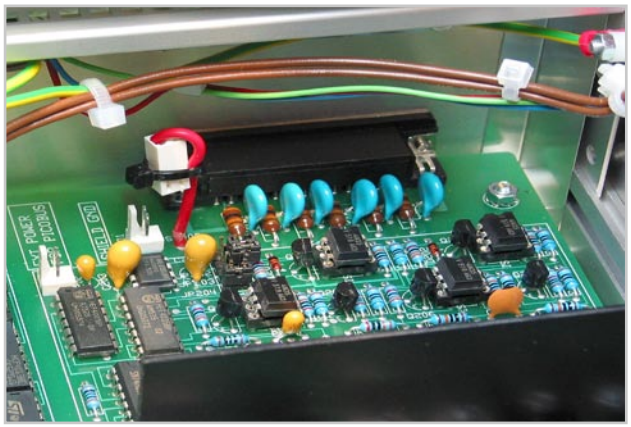


Fig.1: AVS-47 and AVS-47A: The Picobus Cable shield must be disconnected from the bridge ground in order to enable full galvanic isolation from the AVS47-IB interface box. The shield remains grounded at one end, namely to the IB box via the 25-way connectors. If the 25-to-9 pin adapter for direct PC connection is such that it does not connect ground between the metal parts of the connectors, then J201 must exist.

AVS-47B: The Picobus Cable shield is NOT connected to the bridge by default (J201 not inserted). This enables full optical isolation between the AVS47-IB and the bridge without any user considerations. In case of direct interfacing with a PC, one must check whether the 9-to-25 way adapter carries the protective ground or not. In the negative case, J201 must be inserted.

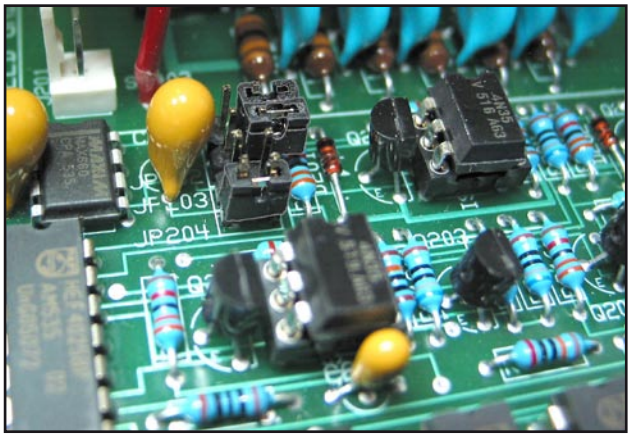


Fig.2: AVS-47 and AVS-47A: Short circuit pieces JP201 and JP202 must be removed, when Picobus gets external power (optoisolation is enabled if also the ground jacket is disconnected from bridge). They can be stored as in this photo. JP204 is always needed.

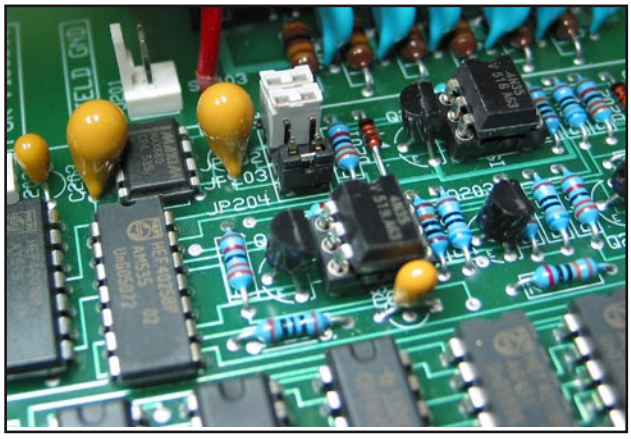


Fig.3: AVS-47 and AVS-47A: If no external +5V power is available for the output side of Picobus, power must be taken from the bridge by placing jumpers JP201 and JP202 as in this photo. Optical isolation is not possible, as the bridge does not have an isolated supply for Picobus. The Picobus cable shield can also remain grounded.

AVS-47B: This revision has an isolated DC/DC converter for Picobus. WITH THE AVS-47B THESE TWO JUMPERS MUST NEVER EXIST, ONLY JP204.

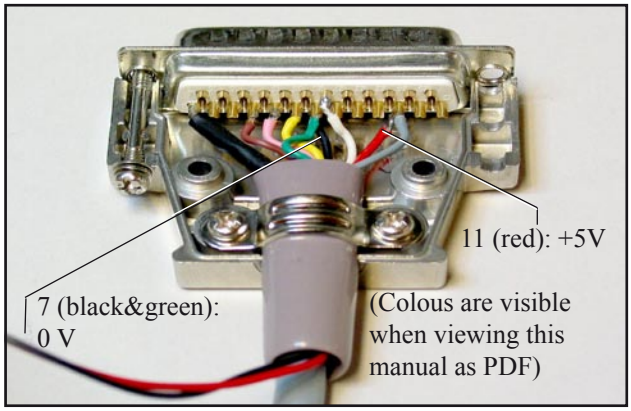


Fig.4: AVS-47 and AVS-47A: External, stabilized +5V 200mA power supply can be connected to pins 11 (+5V) and 7 (0V) of the Picobus DB25P connector. Both pins are on the lower (longer) row.

AVS-47B: External power cannot be used because AVS-47B has an internal, isolated power supply for Picobus.

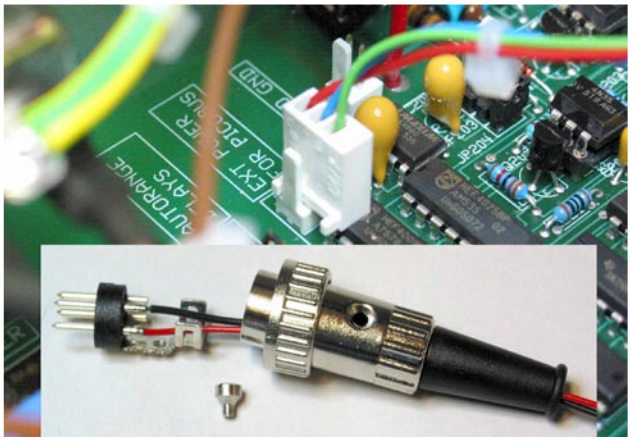


Fig.5: AVS-47 and AVS-47A: External power can also be wired to the 4-way DIN connector, which is normally used for battery power input. Move the 4-way Molex socket with three wires from J403 to pins 1 and 2 of J203 as shown. Solder the power leads to pins 1 (+5V) and 2 (0V) of the 4-pin DIN plug.

AVS-47B: External power cannot be used, and it is not needed.

3. ABOUT THIS PACKAGE

3.1. INSTALLING THE VI PACKAGE

The Picobus VI -package is downloaded from our WEB site as one compressed .ZIP file. Create a new folder inside the "instr.lib" folder of your LabView 7.1 or later, and give it a descriptive name like "Picobus". Download or move the already downloaded ZIP file into this new folder. Unzip the files. Note that the name of the ZIP envelope contains the revision number of this VI package. If we update the VI:s, only the new package will have a different name (higher revision number). Save the ZIP file on your disk, then you can always return back to an older revision, should some compatibility issue in your application prevent using the newer package.

Start your LabView. From the "Open.." -tab, navigate to the new Picobus folder, which might be found, for example, as "C:\Program Files\National Instruments\LabVIEW 7.1\instr.lib\Picobus". Select the AVS47B Tree_b.vi and open it. **Show the block diagram and turn on context-sensitive help**.

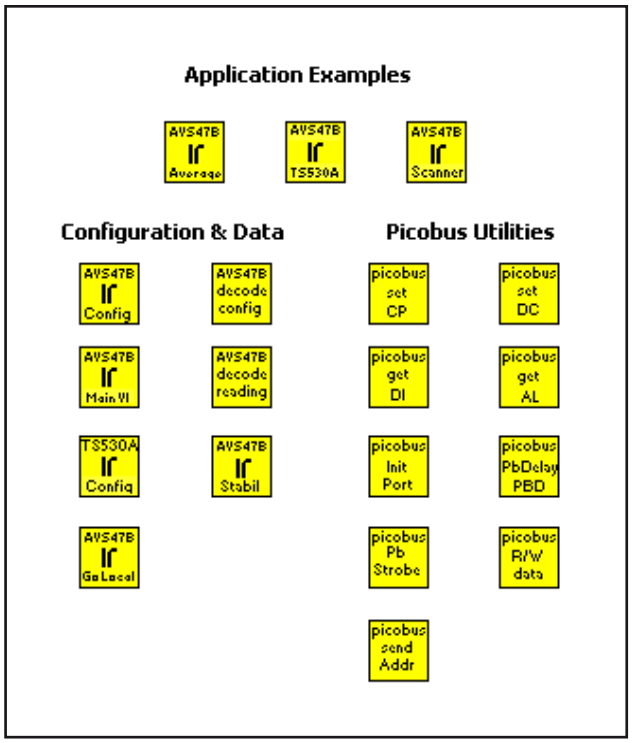


Fig. 6: AVS47B Tree_b.VI block diagram

3.2. INTRODUCTION TO THE VI:S

If possible, install this VI package on your PC and use this Application Note together with your LabView for studying the various blocks. The on-line VI descriptions and context-sensitive help texts complement the discussions of this AN.

Application Example VI:s
This set of VI:s consists of three Application Examples, of seven Data&Configuration VI:s and of nine Picobus Utility VI:s.

The Application Examples demonstrate how to control both the AVS-47B and the TS-530A via the Picobus and how to use the supplied VI:s programmatically.

Average VI is a simple program that takes a predetermined number of A/D conversions, and calculates the average of their results.

Avs47B and Ts530a VI combines these two instruments into one basic control system. In order to keep the example as simple as possible, no calculations of heater power etc. are included.

Scanner is a more complicated VI that uses the AVS-47B as an 8-channel scanner. It repeats taking averages of sensors to be scanned. Every sensor has its own measuring range, excitation and average count settings. In addition, the program allows each channel to stabilize before starting to take readings.

Configuration&Data VI:s
Because the Picobus transactions consist of both writing and reading of settings of the AVS-47B, one must always transact using a TXSTR that contains an appropriate configuration.

The **Avs47BCfg** VI constructs a proper TXSTR to be sent to the bridge, based on various front panel settings (another way to get a proper TXSTR is to read the configuration in local mode). You may need this VI only in the beginning of an application.

Using the above constructed TXSTR, one can communicate with the instrument. This is the task for **Avs47B** VI, the main transaction VI.

After a transaction one may want to know the previous configuration, which was valid until completion of the last transaction. The **AVS47B Decode Config** VI extracts and decodes this information from the RXSTR, the response string.

The A/D conversion result is a 4 1/2 digit number in BCD format. It is obtained from the bridge as a 4*4+1=17 bits long "binary" string. In order to express such an integer number as a floating point resistance, one must use information about the measuring range. Such information was provided by the **AVS47B Decode Config** VI, and the

resistance is calculated by **AVS47B Decode Reading** VI.

The TS-530A Temperature Controller gets its programming information via the AVS-47B. Sending programming messages to the TS-530A using the **TS530A Config** VI configures the resistance bridge at the same time. The transmitted string TXSTR must therefore be constructed so that it does not alter the bridge settings by mistake. This limits the ways how TS530A Config can be used.

AVS47B Stabil VI constructs TXSTR, transmits this string to the AVS-47B without using the response string for anything. Then this VI waits for a given number of seconds until control is passed to next VI in the application. The purpose is to give the bridge time to settle before starting to take readings. The time required depends on excitation and usually between 5 and 15 seconds, depending also on required accuracy and on the possible average count.

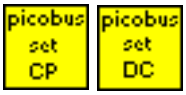
One should always leave the AVS-47B in the local mode. This makes it possible to re-start VI:s so that the existing settings of the bridge are kept intact. The trick is to read the settings in the local mode and to use them for constructing the initial TXSTR. After that, one can change over to remote mode without altering the settings. The **AVS47B GoLocal** VI is typically used only once, in the end of an application.

4. VIRTUAL INSTRUMENTS

4.1. InitPort.vi

This VI is used once in the beginning of an application. It opens a VISA session that uses the denoted serial port (VISA resource name). Typically the port is either COM1 or COM2. Do not set the resource name to LPT. Because the RXD and TXD data lines are not used at all by Picobus, the serial parameters are of no importance. But it is very important that Flow Control is set to “None”. If flow control is anything else, the property nodes fail to drive the handshake lines.

4.2. SetCP.vi and SetDC.vi



These two VI:s are exactly similar, except that SetCP sets the state of the RTS (Request to Send) and the SetDC sets the state of the DTR (Data Terminal Ready) RS232 handshake output. These VI:s are based on a “property nodes” that control the serial port, which is determined by the VISA resource input. State can be either 0 or 1. The property node works only, if no error is coming in, otherwise the error code and description are passed through and CP/DC state is not changed. If you merge your own errors to the VISA error cluster, take into account that the rest of the VI may behave unexpectedly after an error has occurred.

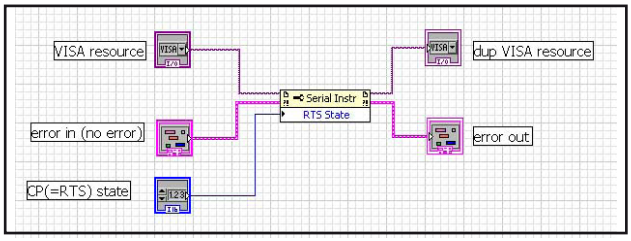
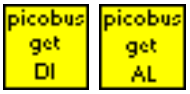


Fig.7: SetCP.vi

The RTS and DTR signals are found in the Modem Control Register of the selected COM port at (base I/O address + 4). Standard I/O addresses are 3F8 for COM1 and 2F8 for COM2. Bits in the register have the following meanings:

7 0
6 0
5 0
4 loopback test
3 out2
2 out1
1 RTS (request to send)
0 DTR (data terminal ready)

4.3. GetDI.vi and GetAL.vi



These two VI:s are exactly similar, except that GetDI reads the state of the CTS (Clear To Send) and GetAL reads the state of the DSR (Data Set Ready) RS232 handshake input. Like above, these VI:s are based on the property node of the previously opened VISA session.

The CTS and DSR signals are found in the Modem Status Register of the VISA-opened COM port at (base I/O address + 6). Bits in this register have the following meanings:

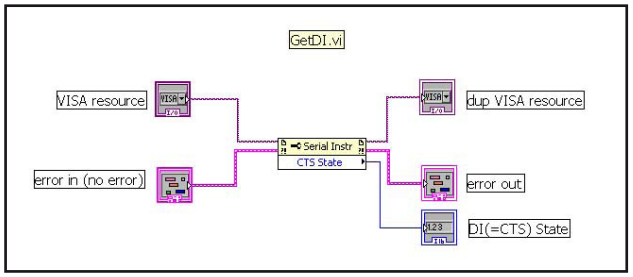


Fig.8: GetDI.vi

7 DCD (data carrier detect)
6 RI (ring indicator)
5 DSR (data set ready)
4 CTS (clear to send)
3 DCD has changed
2 RI has changed
1 DSR has changed
0 CTS has changed

4.4. SendPbAddr.vi

This VI sends the 8-bit PICOBUS address PBA to the AVS-47B and strobes it in.

The integer PBA is first converted to a string. Then the following sequence is run 8 times (8=length of the address string):

1) CP is set to zero, 2) DC is set to a value determined by the n'th bit in the address string, 3) an optional delay lets the DC line to settle, 4) CP is set to one (this rising edge clocks the bit in), 5) optional delay, 6) CP is set to zero, 7) DC is set to zero. After all 8 repeats have been done, the address is strobed. Strobing forces the AVS-47B to compare the transmitted address with its DIP switch setting. If they are the same, the address is accepted and data ports are opened for the next phase, i.e. sending and receiving of the actual data.

The address, which is usual1y 1 (default), is converted into a string of 8 characters ("format into string", length=8, preceding empty places are patched with zeros). Inside the FOR loop, "scan from string" extracts one character in turn from the string and interprets it as a bit, which is used to control the SetDI.vi.

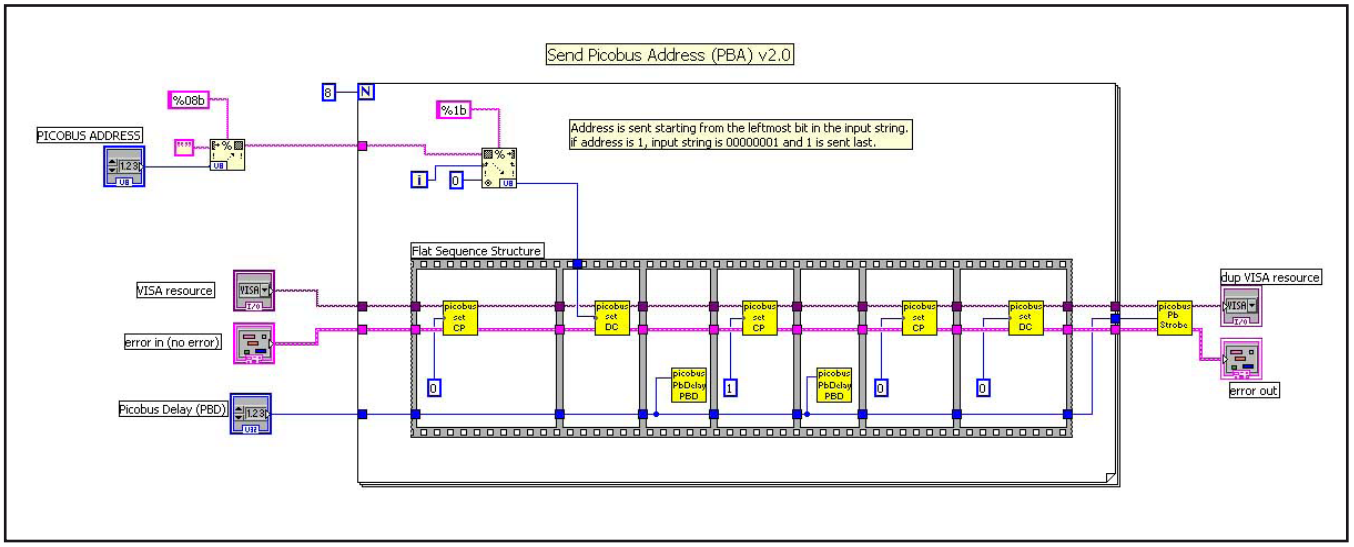


Fig.9: SendPbAddr.vi

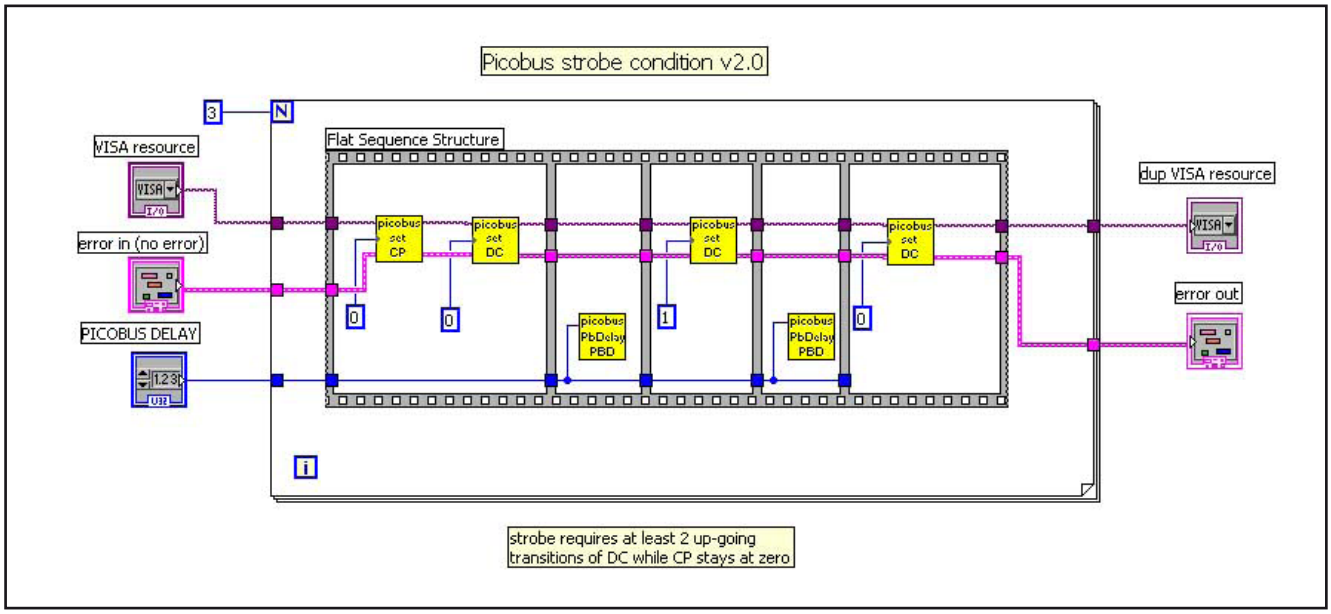


Fig.10: PbStrobe.vi

4.5. PbStrobe.vi

This VI is used to strobe the Picobus Address (PBA) after the address string has been transmitted, and the actual data string TXSTR once it has been sent. Strobing tells to the AVS-47B that the computer has transmitted all bits of those strings, and that the bridge can now transfer this information further from shift registers to the parallel latches that control the analog circuits.

Because the COMx: ports do not offer more than two handshake outputs (RTS and DTR), which are required for the CP and DC signals, strobing has been implemented in an unusual way:

The roles of the clock and data signals are interchanged so, that the clock is held permanently at zero whereas the data toggles between 0 and 1 for three cycles. This is a situation, which will never happen in normal operation, and it is quite easily detected by using sequential logic (flip-flops).

The **PbStrobe.vi** requires only two inputs, the VISA resource name of the COMx: port and the Picobus Delay (PBD). A flat sequential structure takes care for toggling the DC up and down for three times.

4.6. RwPbData.vi

This VI sends the 48-bit data string TXSTR to the AVS-47B and reads the response string, RXSTR.

The TXSTR is received either from the Avs47Cfg.vi configuration block or by reading from the AVS-47B. Each one of the 48 bits in turn is used for running a flat sequence structure. The structure is similar than in SendPbAddr.vi, except that there is one additional sub-VI, **GetDI.vi**. It reads the DI (Data from Instrument) response bit that corresponds to the DC bit that is just being sent.

The trick is then to construct RXSTR from the DI bits. Here it has been made by using local variables. First, the incoming initial value of TXSTR is written to its local variable “TXSTR”. This value is led to a hidden indicator “OUTSTR”. When execution of this VI enters the FOR loop, the value of the “OUTSTR” indicator is written to the two instances of the “OUTSTR” local variable inside the loop. Now these two instances have been initialized.

On each of the 48 turns, one bit in “OUTSTR” is replaced by a bit, whose value depends on the corresponding DI bit and whose position depends on the turns counter “I”. The two instances of the local variable are used for transferring the modified string back to be the initial string for the next turn.

The RwPbData.vi ends with applying the strobe condition after all 48 bits have been sent.

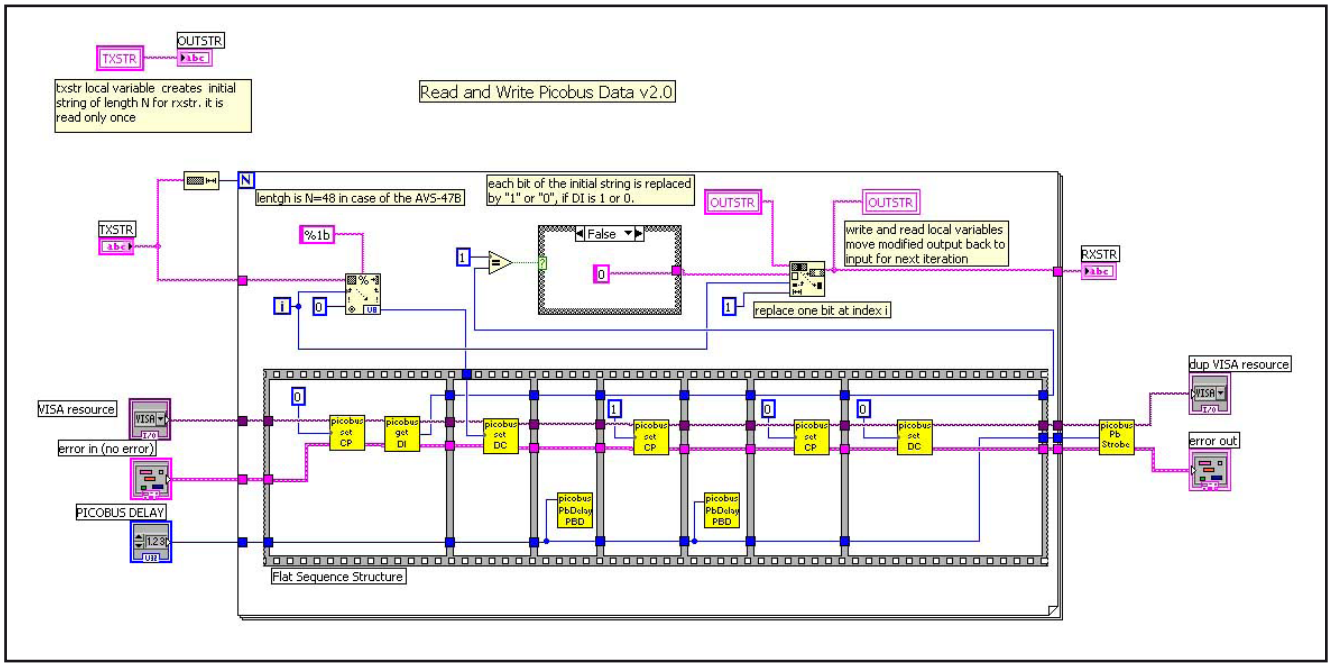


Fig.11: RwPbData.vi

4.7. PbDelay.vi

The purpose of this VI is to generate short time delays. LabView's own delay timer cannot produce delays shorter than one millisecond, whereas PICOBUS may require delays in the ten-to-hundred microsecond range. Using a PBD (PicoBus Delay) is not always necessary. But if the interface cable is long, if it has RF filters, or if the drive capability of the COM port is poor, then a delay may be necessary for reserving time for the DC signal to reach its final level before the rising edge of the clock signal CP. If you find random behaviour when programming the bridge or when reading the resistance values, or if these completely fail to work, try to increase the delay. Start from a large value, say 1E5, and if this solves the problem reduce the PBD until the problems re-appear. Then increase PBD by a factor of two or so in order to reserve some safety margin.

The simple processor loop is not intelligent in any way. It wastes processor time and therefore the delay should be as short as possible. 0 is the minimum value and 1E6 is the maximum. Then the transactions are really slow.

If you need delays longer a couple of milliseconds, consider replacing PbDelay by LabView's own "Wait Until Next ms Multiple".

4.8. Avs47Cfg.vi

This VI "configures" the AVS-47B in the meaning, that it prepares TXSTR, the data string to be transmitted. It does not yet make any kind of transaction with the bridge. That has been reserved for the **Avs47b.vi** transaction VI.

Data is located in the TXSTR so, that the first 4 highest bits are unused and the next 12 bits contain the remote set point as a "binary string". The most significant bit (MSB) is the leftmost bit in TXSTR. Because strings are indexed from left to right, this MSB will be the first bit to enter the AVS-47B.

For ease of use, the digital reference is given as an integer from 0 to +19999. However, a 12-bit DAC accepts numbers only up to 4095, and therefore the given number is first divided by five and then rounded to the nearest integer. The result is converted into a 16-bit string, where the four highest bits are identically zero. This data fills two of six 74HC164 shift registers in the AVS-47B (U407, U416).

The next 8-bit part of TXSTR stores the address for the reference data (U402). In the case of the AVS-47B, this address is 3 (binary string " 00000011"). After two unused bits, TXSTR is further built from input (2 bits), channel, display, excitation and range (3 bits each). This 16-bit long part of TXSTR corresponds to registers U306 and U310.

The last of the six shift registers (U206) contains only two pieces of information, namely remote/local mode and disable AL. Knowing the structure of TXSTR is by no means important. It is explained here only, because some VI front panels show TXSTR.

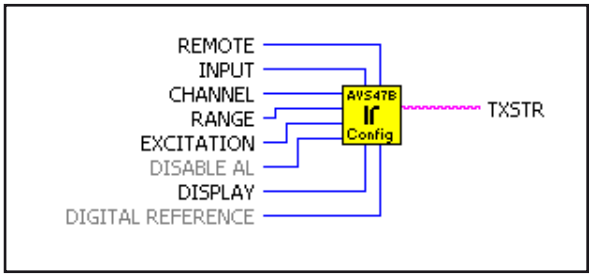


Fig.12: Avs47cfg.vi connector

4.9. Decodeconfig.vi

This VI extracts and interprets the configuration bits from the RXSTR response string. It is a straightforward application of the “Scan from String” LabView function.

The VI outputs integer numbers for programmatic use and clearly understandable texts that are readable on the VI front panel.

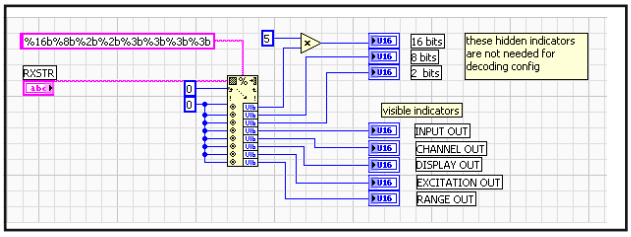


Fig.13: Decodeconfig.vi block diagram

4.10. Decodereading.vi

This VI extracts and interprets the A/D conversion result, which is stored in 1+4*4=17 bit BCD format in the RX-STR response string.

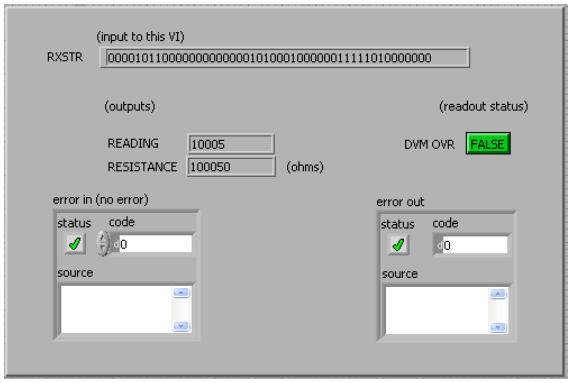


Fig.14: Decodereading.vi front panel

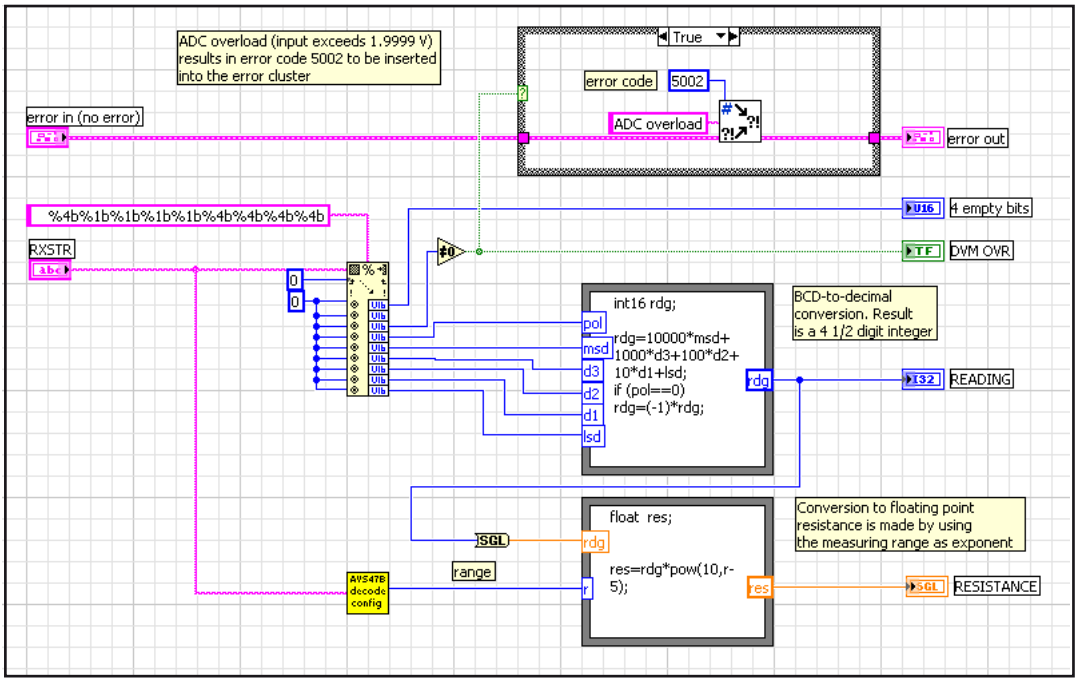


Fig.15: Decodereading.vi block diagram

BCD-to-binary conversion is made in a formula node, which yields a 4 1/2 digit integer. The sensor resistance is calculated from this with the help of the measuring range (r, obtained from Decodeconfig.vi), which is used as an exponent in $R=rdg*10^{(r-5)}$.

If the overload status bit in RXSTR is set, the VI inserts an error message in the error cluster.

4.11. TS530Cfg.vi

This VI configures the TS-530A Temperature Controller. A little confusing after AVS47Cfg.vi, but now the configuration information is really sent to the controller, whereas the configuration VI for the resistance bridge only constructed the TXSTR.

This apparent lack of logic comes from the fact, that an AVS-47B can be used alone, but the TS-530A can be used only together with the resistance bridge. The same 3*8 first bits of the TXSTR, which can be used for setting the AVS-47B's remote reference, are also available for programming the TS-530A. Which one happens, depends on the address contained in the third byte. Address 3 causes data to be written to the AVS-47B remote reference DAC, whereas addresses 9..15 store the PID parameters to their registers in the TS-530A.

The programming information that is intended for the TS-530A passes through the AVS-47B, which must be set in the remote mode before it can relay the 3*8 bits further via the 37-way data cable to the temperature controller. Therefore, the TXSTR must contain proper settings for the bridge. This, in turn, is ensured by transacting with the bridge before configuring the controller. This is not a limitation, because the two instruments are always used together anyway.

The following 6 PID parameters can be configured: Set point, proportional gain, integrator time constant, derivator time constant, power bias and power range.

Configuration is made by transacting 6 times, once for each PID parameter and using a different address. The PICOBUS strobe condition generates a signal that is used to strobe data also into the TS-530A. In order to avoid unnecessary PICOBUS traffic and unnecessary slowering of transactions, TS530Cfg.vi is based on a CASE structure, which runs only if a front panel setting has changed (a sum of all settings is calculated and it is compared with the previous sum). This works well when the VI is used from its front panel, but in programmatic use, this simple method may fail (if you change a control one step upwards and another control one step downwards, the sum remains the same and the VI does not run). Then you must either remove the CASE or design a solution that is more suitable for your application.

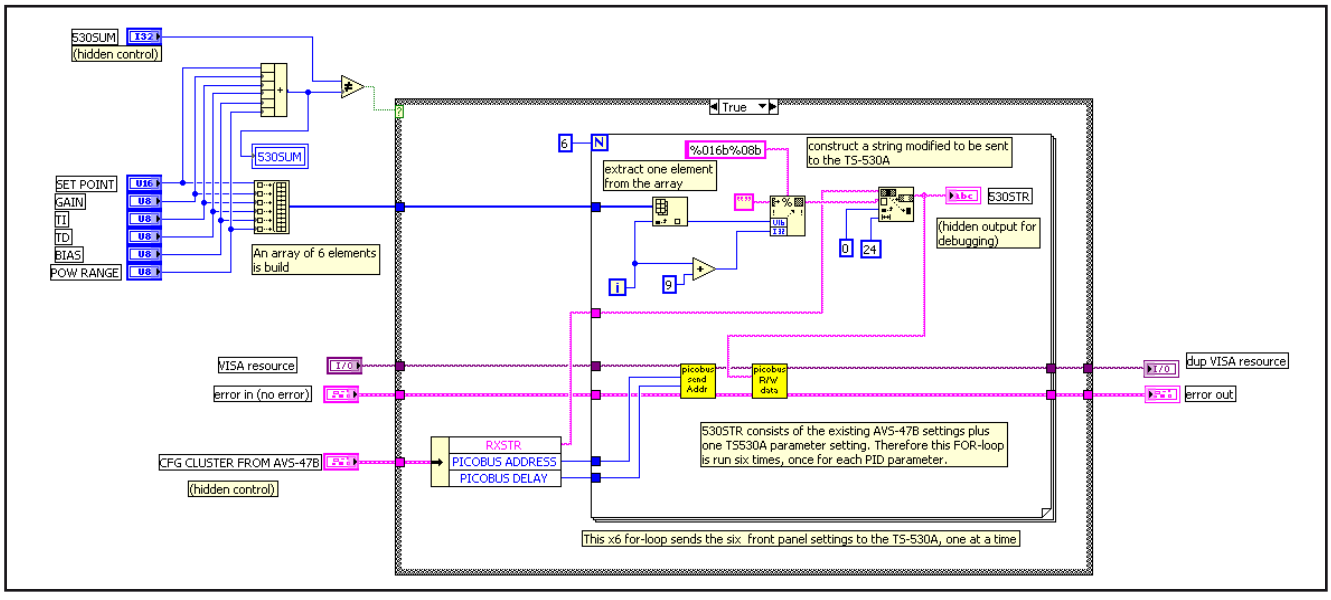


Fig.16: Ts530Cfg.vi block diagram

4.12. Stabil.vi

This VI is used as a sub-VI in the Scanner.vi application example. It configures the AVS-47B (constructs TXSTR), and sends this configuration to the bridge. After that, the VI waits for a given number of seconds (STAB DELAY) before terminating.

The purpose of STAB DELAY is to reserve time for the bridge to settle after changes in the channel number and other measurement parameters. Then the subsequent measurement should be free of transient effects.

4.13. GoLocal.vi

This VI sets the AVS-47B in the local mode. It needs the last used TXSTR or last obtained RXSTR as input.

The action is to reset the remote control bit in the TXSTR and then transact with the bridge.

It is a good practice to leave the AVS-47B in the local mode. Then it is possible to start a new application without introducing unwanted changes in the setup.

4.14. Avs47B.vi

This is the main transaction VI of this package. It can be used as such, from its front panel, or programmatically by wiring suitable values to its inputs. Refer to LabView online help for data ranges. Developing your own application should start with playing with Avs47B.vi until its operation is well understood.

This VI has four “control inputs”. VISA resource is usually either COM1 or COM2, unless the PC has more than two COM ports installed. The VISA resource box shows the available serial ports. **“Synchronize to ADC” should normally be TRUE**. Transaction is faster, if synchronize is set to FALSE. You can use this, if the program takes readings slower than 2.5 times in a second, so that no result can be read more than once. Picobus address is 1, unless a second (or more) bridge uses the same bus. Picobus delay can be 1 or 0 in the beginning. It should be higher only, if required by computer speed or effective filtering.

This VI should be started with REM=No, when the AVS-47B is also in local mode. The existing settings are read, and if REM is then set to YES, the bridge state will not change when entering the remote mode.

The REFERENCE input is actually required only, if one wants to use the DR (delta-R) display.

In the beginning, a VISA session **is opened, but it is not closed upon termination**. This is because one may want to program also the TS-530A, which is possible only if the session remains open. This should not increase the

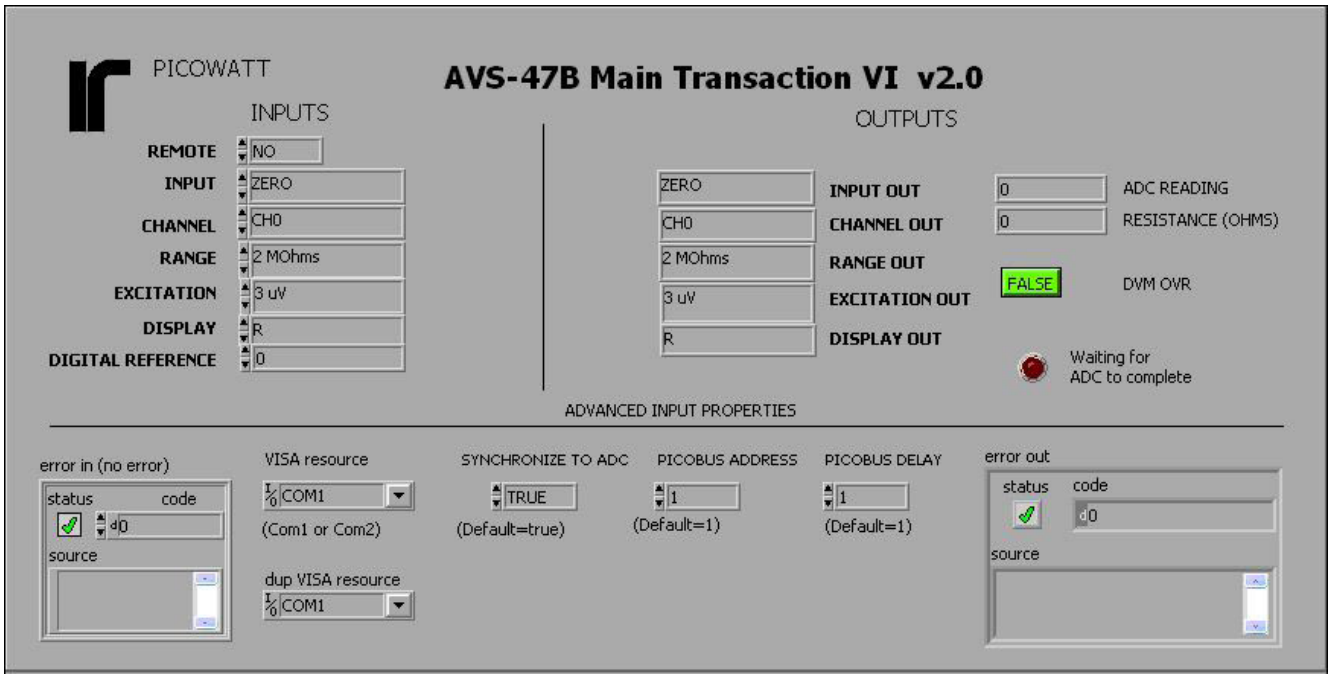


Fig.17: Avs47b.vi front panel

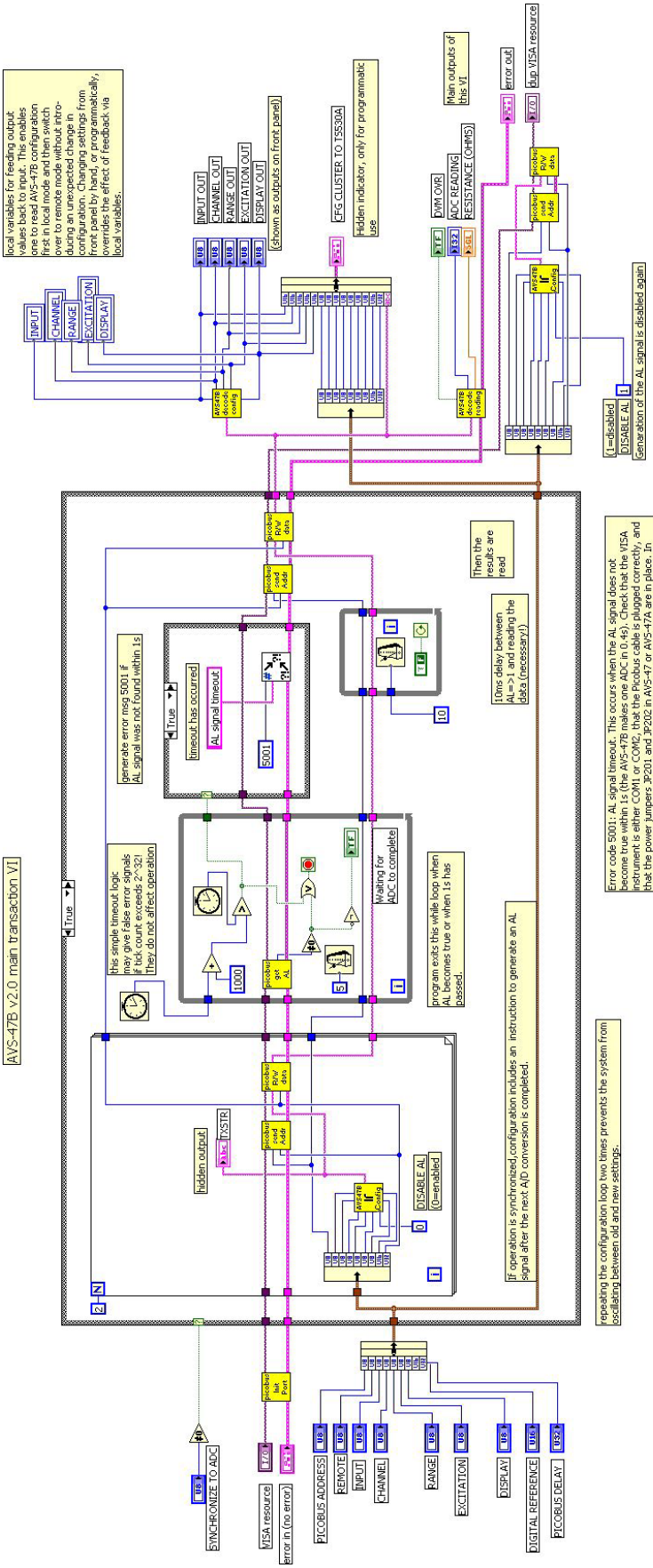


Fig.18: Avs47b.vi block diagram, synchronization = TRUE

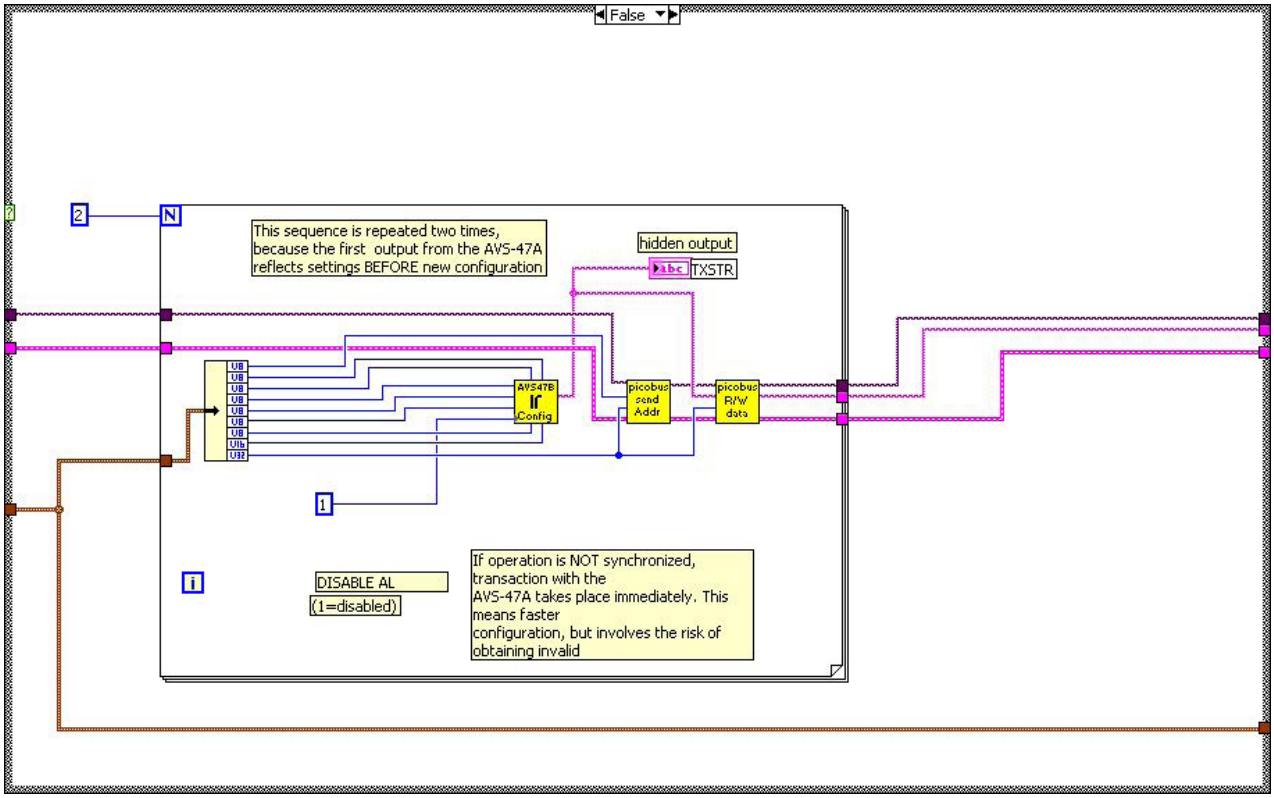


Fig.19: Avs47b.vi block diagram, synchronization = FALSE

number of open sessions, because the VI tries to open the already-open VISA session. It may still be a good and safe practice to add the **Instrument I/O/Serial/VISA Close** in the end of an application, or at least instruct LabView to automatically close VISA sessions.

In case of an ADC overload, both the integer ADC conversion result and the calculated resistance are exactly zero while the OVR indicator is TRUE. Never think that a zero reading means zero resistance without checking the overload indicator!

Following are some remarks about the operation of this VI, starting from the **more important case, when transactions are synchronized to A/D conversions.** Refer to Fig.18, which shows the VI when synchronization is set to TRUE.

The control inputs are first bundled into a cluster so, that there are less wires to handle. Inside the FOR-loop, these controls are unbundled and used for configuring the AVS-47B, i.e. for constructing the TXSTR. Note that generation of the AL alarm signal is enabled (constant 0 is wired to the input). We need now only a very simple transaction with the bridge. Therefore, the PBA is sent first and then the newly built TXSTR is transmitted. The response string, RXSTR, tells the previous state of the bridge, and therefore this transaction is made two times. Then RXSTR corresponds to the new state.

Generation of AL being enabled, we have to wait until AL becomes true before reading the latest A/D conversion result. The WHILE-loop in the middle of the diagram polls the AL bit in the Modem Status Register at 5 millisecond intervals, until AL is asserted. If something goes wrong, and AL never comes true, the WHILE loop would never end. Therefore it has a one-second timeout (AL should be detected in 0.4 seconds at most). A tick count is stored when entering the WHILE loop, and 1000 ms is added to the count. This figure is compared against a new tick count, which is updated on every iteration of the loop. The loop exits either when 1 second is exceeded or AL is detected. A timeout flag is output to the next stage.

If the WHILE loop terminated because of timeout, the TRUE side of the CASE structure adds an error code of 5001 and a short error description to the error cluster. For simplicity and because of the rarity of errors, we have used one single error cluster throughout the VI. This approach has a problem: If timeout has made the error cluster non-empty, the VISA property nodes and some LabView's functions do not work properly. Therefore, one can never see a situation with this VI where everything else works except AL detection. LabView's error indicator shows only the first error that has occurred and it must be corrected before subsequent errors can be seen.

A 10 millisecond delay is inserted between detection

of AL and starting to read the result. During this delay, the AVS-47B moves the new result into its output shift registers so that it can be read.

In the end, the configuration and conversion result are decoded out from the RXSTR. Programmatic use of this VI is aided by bundling a cluster from all settings and TXSTR. Finally, generation of the AL signal is disabled.

Note also the use of the 5 local variables. These variables transfer the read configuration (RANGE, CHANNEL etc.) from the end of the VI back to the input controls. With this method, the initial local mode settings are made defaults for the remote control.

If synchronization is set to FALSE, operation of the VI is simpler, as shown by Fig. 19. The AVS-47B is first configured with AL disabled. After having transacted with the bridge two times, the VI jumps to its last section of interpreting the results. No error message can be generated.

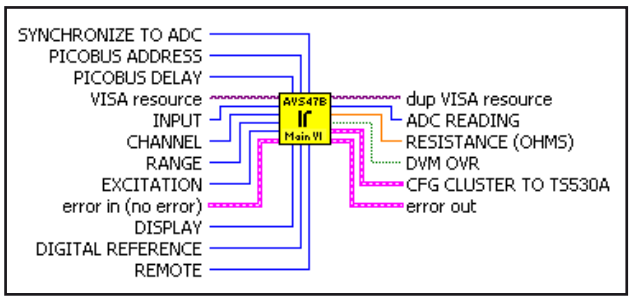


Fig.20: Avs47b.vi connector

4.15. Average.vi

This is the simplest of the three application examples. It makes use of only one sub-VI, namely the Avs47b.vi.

Inside the FOR loop, the Avs47b.vi configures the bridge and reads the conversion result. The loop is repeated AVE COUNT times.

At the output border of the loop, the results are stored in an automatically indexed array. After the measurements, the mean value (average) and the standard deviation (RMS noise) are calculated from the readings in the array.

Use the RMS noise for checking, that the noise performance of your system is not very much worse than what is suggested by the AVS-47B specifications.

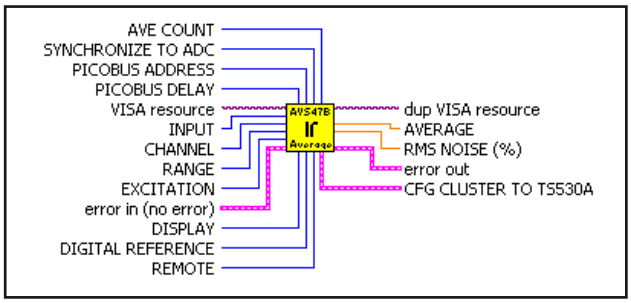


Fig.21: Average.vi connector

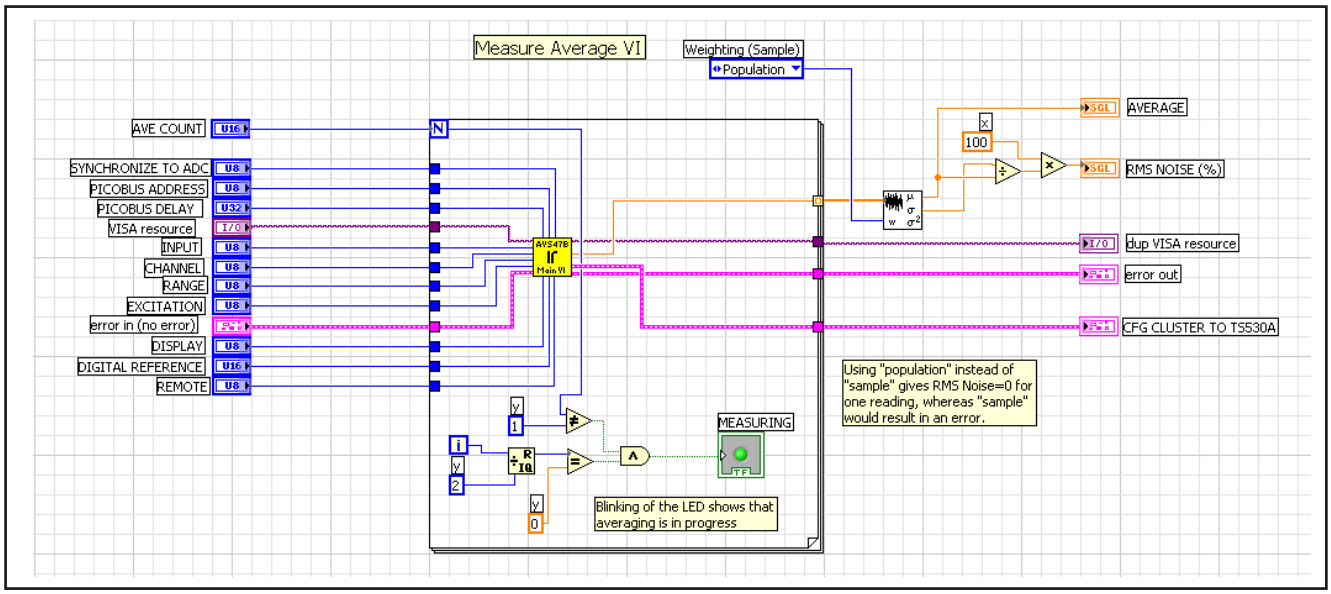


Fig.22: Average.vi block diagram

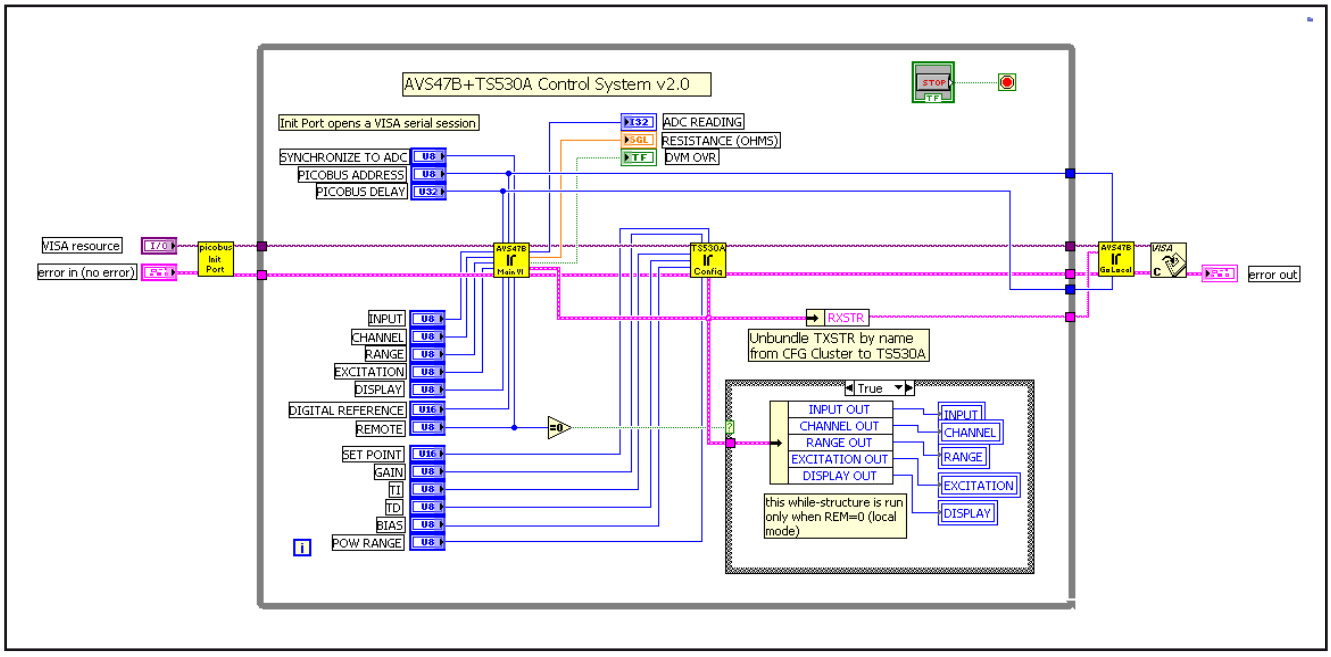


Fig.23: Avs47AndTs530.vi block diagram

4.16. Avs47andTS530.vi

The second Application Example is a simple control system, formed by an AVS-47B and a TS-530A. See the front page of this AN to see the panel of the VI.

The VI starts with opening a VISA session for the denoted serial port. Then the AVS-47B is configured, TXSTR is sent to the bridge and the response is read (Avs47b.vi, the main VI).

After the state of the AVS47B is known (this VI uses the response string RXSTR), one can configure also the temperature controller (Ts530Cfg.vi). The RXSTR from Avs47b.vi is fed to Ts530Cfg.vi using a data cluster. The Ts530Cfg.vi transacts with the AVS47B+TS530A combination several times in order to program each of the PID parameters (one transaction is needed for each PID parameter).

This control system runs as a WHILE loop until terminated. Before the VI exits, the AVS-47B is set into local mode and the VISA session is closed.

The TS-530A is a “digitally managed analog controller”, whose control loop is purely analog. Therefore temperature is controlled quite independenly on whether the VI makes repeated transactions or not, as long as no system settings are changed.

4.17. Scanner.vi

This is the largest application example an it should be studied inside the LV environment.

Now the AVS-47B is used as an 8-channel scanner. it measures continuously those sensors that have been included in the “scan range”, i.e. a contiguous range between and including the first and last channel to be measured. Each channel has its own measuring range, excitation and averaging count. The AVS-47B needs some time to find a new balance after the channel and other settings have been changed. The time that is required varies with excitation and, of course, on what are the required accuracy and average count of the next channel. Therefore each channel has also its own “stabilization delay”. A safe guess for this delay is 5..15 seconds.

If all 8 channels are not required for cooled sensors, one can use the remaining channels for checking the offset (Input=ZERO) and scale (Input=CAL , range=200R), and use these data for digital calibration or checking measurement quality. Similary, one can check the offset and scale of the DR or 10xDR displays using two empty channels and the DR display mode.

5. PICOBUS WAVEFORMS

The five oscilloscope screenshots show, how the various signals on the Picobus look like. Computer is a 1MHz Pentium.

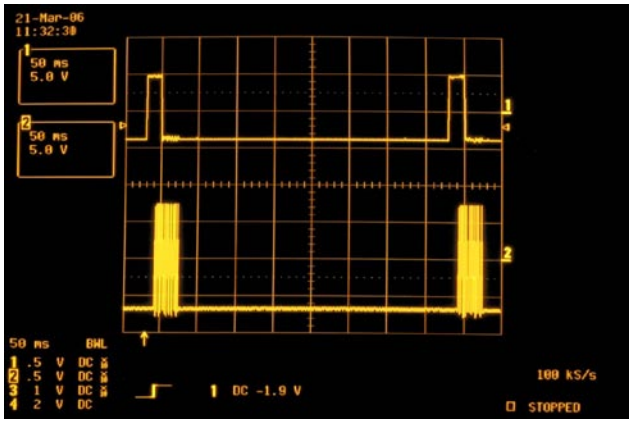


Fig. 24: When an A/D-conversion completes, the AVS-47B asserts the AL signal (trace 1). After about 10ms, transactions with the bridge starts (trace 2). All these screenshots have been taken with Avs47AndTs530.vi, and therefore each transaction consists of more than one address-data sequences. The time interval between two successive transactions in synchronized mode is 0.4 seconds, which is the A/D conversion rate.

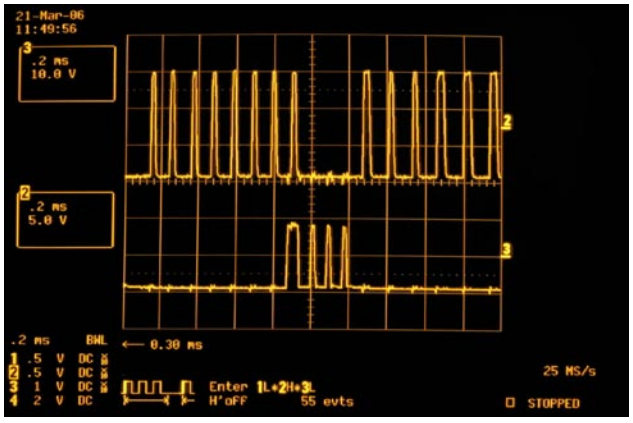


Fig. 25: A transaction starts by sending the address. Trace 2 shows the eight clock (CP) pulses. Because the trace moves from left to right, the 8th pulse is for the least significant bit LSB. The default AVS-47B address is 1, so data (trace 3=DC) goes from 0 to 1 before the LSB clock pulse . A strobe condition is implemented by keeping CP down and toggling DC for 3 cycles.

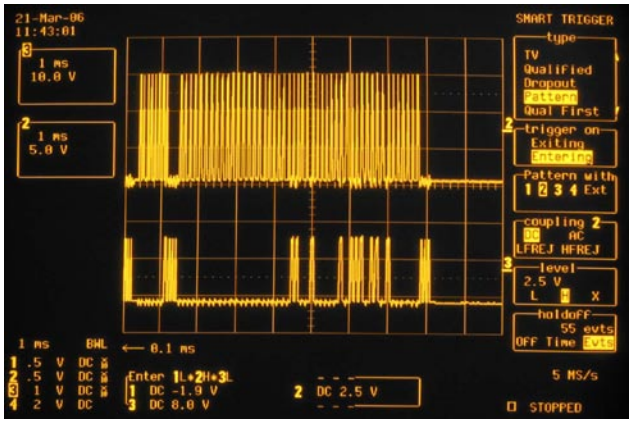
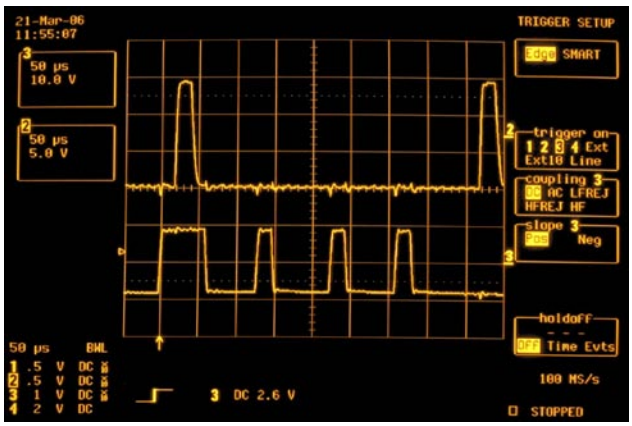


Fig. 26: One complete transaction consist of 8 CP pulses for clocking the Picobus Address (PBA) and of 48 clock pulses for clocking in the configuration data. Trace 2 shows the clock pulses and trace 3 is the data from computer (DC). It shows 4 pulses after address (one for address=1 and three for strobing). The transaction ends with three strobing DC pulses after the data string.



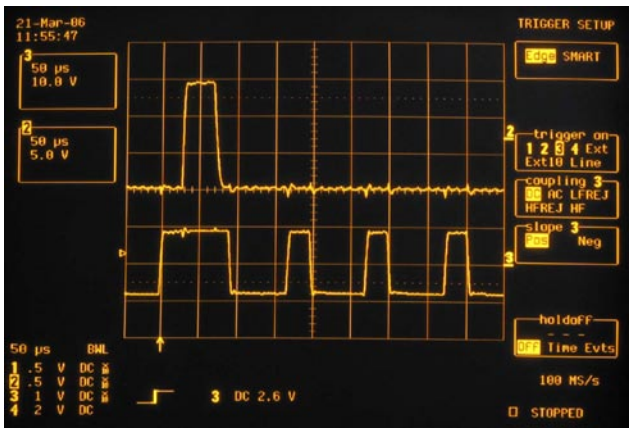


Fig. 27: The effect of the Picobus Delay (PBD) depends on the speed of the computer. Here the DC signal has a settling time of 20us when PBD=1 and 40us when PBD=100.

INDEX

Q

R

S

T

V

W

Z