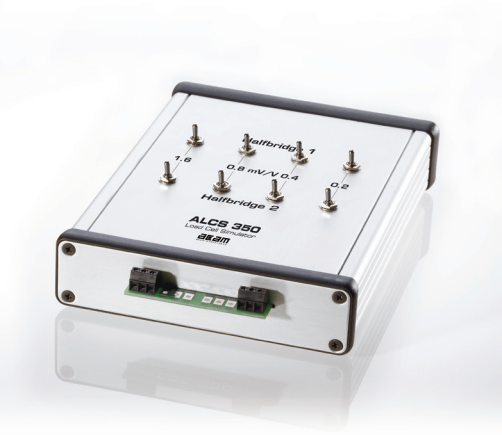
aeam
mess • electronic

PICOSTRAIN

Data Sheet

ALCS-350-V2

Load Cell Simulator



17. Juli 2009
Document-No.: DB_ALCS_V2_en V0.1

Published by acam-messelectronic gmbh

Limited Warranty

The ALCS load cell simulatror is designed and offered by acam-messelectronic. The hardware are warranted against defects in materials and workmanship for a period of 12 months from the date of shipment, as evidenced by receipts or other documentation. acam-messelectronic will, at its option, repair or replace equipment that proves to be defective during the warranty period.
The information provided herein is believed to be reliable. However, acam-messelectronic assumes no responsibilities for inaccuracies or omissions. acam-messelectronic assumes no responsibility for the use of this information, and all use of this information shall be entirely to the user's own risk. No patent rights or licenses to any of the circuits described herein are implied or granted to any third party. acam-messelectronic does not authorize or warrant any acam-messelectronic product for use in life support devices and/or systems.
Except as specified herein, acam-messelectronic makes no warranties, express or implied, and specifically disclaims any warranty of merchantability or fitness for a particular purpose. Customer's right to recover damages caused by fault or negligence on the part of acam-messelectronic shall be limited to the amount theretofore paid by the customer. acam-messelectronic will not be liable for damages resulting from loss of data, profits, use of products, or identical or consequential damages, even if advised of the possibility thereof. This limitation of the liability of acam-messelectronic will apply regardless of the form of action. Any action against acam-messelectronic must be brought within one year after the cause of action acruses. acam-messelectronic shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failure caused by the owner's failure to follow acam-messelectronic's installation and operation instructions. Further the warranty does not cover damages due to owner's modification of the product, owner's abuse, misuse, or negligent acts and power failure or surges or other events outside reasonable control.

Support

For a complete listing of Direct Sales, Distributor and Sales Representative contacts, visit the acam web site at: http://www.acam.de/company/distributors or refer to chapter 7.2 in this datasheet
For technical support you can contact the acam support team in the headquarter in Germany or the Distributor in your country. The contact details of acam in Germany are:
support@acam.de or by phone +49-7244-74190.

The product ALCS350 comply with EMC directive 89/336/EEC, applied standard DIN EN 61326, Equipment for Control and Laboratory (For use in electromagnetically controlled environment). Generic immunity standard part 2 (EN 61000-4-4: 0,5KV, -4-6: 1V), In case of strong electromagnetic distur-bances there might be a deviation of the output signal from the specification, but only for the duration of the disturbance.

1 Introduction

1.1 General Description

The ALCS-350-V2 is a high-precision load cell simulator, based on accurate resistor networks with 350 Ohm base resistance. It is well suited for all kinds of weighing electronics for strain gauge load cells, independent of the excitation principle that is used. The simulator is compatible with electronics with DC voltage excitation and AC voltage excitation. It is ideally suited to work with acam's PICOSTRAIN devices. The ALCS-350-V2 simulates two separate half-bridges or one full-bridge. The full-bridge can be connected in a Wheatstone or PICOSTRAIN mode. The ALCS-350-V2 has very high precision and high temperature stability and is ideal for testing, qualifying, calibrating and batching weighing instruments. It can generate a very high precision analysis of converter electronics linearity due to the internal structure of the simulator. The ALCS-350-V2 comes with a software which has a graphical user interface that copies the mechanical switches and offers user-friendly device control. Simple ASCII code commands allow the ALCS-350-V2 to be easily embedded in an automated test system.

1.2 Features

- High Precision resistor network based on 350 Ω resistors
- Simulates 0 to 3 mV/V outputs in steps of 0.1 mV/V
- Suitable for all kind of excitation (DC, AC, PICOSTRAIN)
- Excitation voltage –10 V to +10 V, gavanically isolated
- Simulates 1 full-bridge or 2 separate half-bridges
- High temperature stability
- Very well suited for linearity analyses
- 3 interfaces (Human, RS232, USB)
- Powered by USB or separate 9 to 12 V power supply

1.3 Applications

- Weighing electronics evaluation
- Systematic qualification tests of load cells
- Automatic batch testing

2 Technical Data

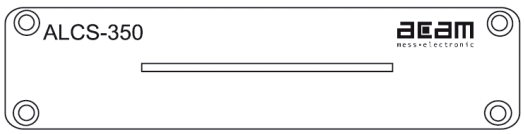
2.1 Technical Specification

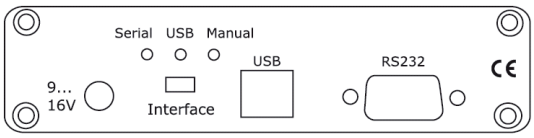
Table 2.1 Technica l specification

Parameter	Value
Supply Voltage	9V to 16 V DC or via USB Interface (requires USB-Mode)
Resistance	350 Ω
Full Scale Output / Output Adjustment	0 to 3 mV/V in steps of 0.1 mV/V (or 0.2 mV/V with half-bridges)
Strain precision	typ. ± 0.005% of F.S. or ±0.1µV/V (PICOSTRAIN wiring) typ. ± 0.03% of F.S. or ±0.6µV/V (Wheatstone wiring)
Zero point precision	typ.± 0.002 mV/V
Linearity	typ. ± 0.01 µV/V (PICOSTRAIN wiring) typ. ± 0.04 µV/V (Wheatstone wiring)
Offset drift	typ.< 0.5 µV/V/K
Gain Error	typ.< 3 ppm / K
Case Material	Aluminum
Operating Temperature	10°C to 50°C
Weight	~ 250 g

2.2 Mechanical Dimensions

Figure 2.1 Mechanical dimensions





Dimensions: W x H x L	130 mm x 40 mm x 158 mm

3 Wiring Schematics

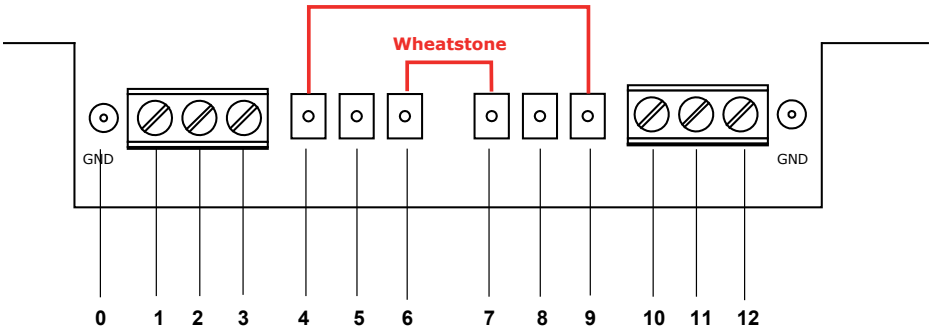
The ALCS-350-V2 is very flexible and can be connected to all kinds of electronic instruments for measuring strain gauges. It can simulate a conventional Wheatstone bridge, 1 half bridge or 2 half bridges forming a full bridge (PICOSTRAIN wiring). The following section shows the various wiring configurations. The electrical connection of a strain gauge amplifier can be done by screw connections using the terminal blocks or via soldering pads.

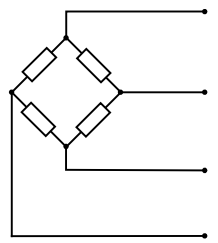
Note:

For high precision applications it is recommended to use the soldering pads instead of the terminal blocks. The recommended wiring is the PICOSTRAIN full bridge, as the accuracy is best. Closing the connection between port-pair 4/9 and 6/7 sets the Wheatstone mode. If open, the PICOSTRAIN wiring is active.

3.1 Wheatstone Bridge

Figure 3.1 Connencting a Wheatstone bridge

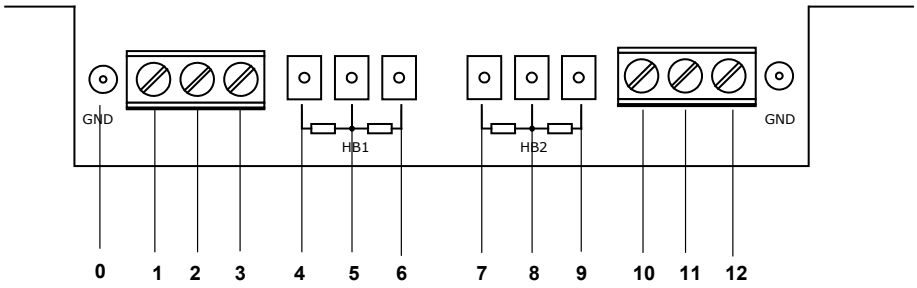


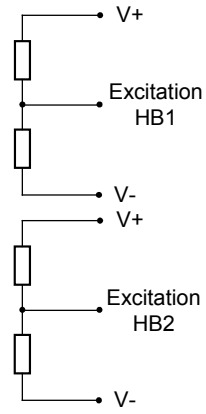


Pin-Nr.	Descripton	Function
0	GND	GND / Shield
1,4	V+	pos. supply voltage
2,5	Excitation-	neg. signal voltage
3,6	V-	V-, neg. supply voltage
7,10	V-	V- ,bridged with pin 7 or 10
8,11	Excitation+	pos. signal Voltage
9,12	V+	V+, bridged with pin 1 or 4

3.2 Standard Full Bridge = 2 Separate Half Bridges

Figure 3.2 Connectins a full bridge

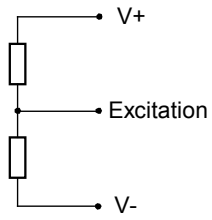




Pin-Nr.	Descripton	Function
Halfbridge 1		
0	GND	GND / Shield
1,4	V+	pos. supply voltage
2,5	Excitation-	neg. signal voltage
3,6	V-	V-, neg. supply voltage
Halfbridge 2		
0	GND	GND / Shield
7,10	V+	pos. supply voltage
8,11	Excitation-	neg. signal voltage
9,12	V-	V-, neg. supply voltage

3.3 One Half Bridge

Figure 3.2 Connecting a half bridge



Pin-Nr.	Descripton	Function
0	GND	GND / Shield
1,4	V+	pos. supply voltage
2,5	Excitation-	neg. signal voltage
3,6	V-	V-, neg. supply voltage

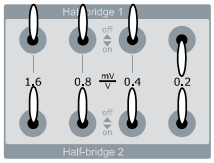
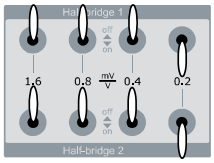
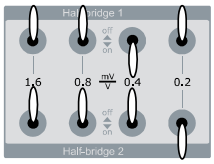
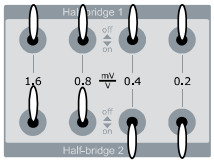
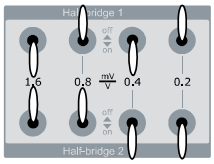
4 Operating principles

The ALCS generates the strain by adding accurate resistors via switches. Due to the operating principle the switches can be set arbitrarily which allows a high flexibility to select a specified strain. The strain is given by the sum of the values depending which switch sets are on.

4.1 Excitation

When operating a full bridge (Wheatstone or PICOSTRAIN) the simulated strain is set separately for each bridge arm in opposite direction. The 4 switches within each row refer to a bridge section (indicated as 'Halfbridge 1' and 'Halfbridge 2'). The total strain is given by half of the sum of the full bridge. The minimum step size of strain variation for each bridge section is 0.1 mV/V, the minimum step size for the strain variation of the full bridge is 0.2mV/V.

Figure 4.1 Switch positions

Switch position	Strain full bridge	Strain Half bridge 1	Strain Half bridge 2
	100ppm	200ppm	0ppm
	200ppm	200ppm	200ppm
	300ppm	400ppm	200ppm
	300ppm	0ppm	600ppm
	800ppm	1000ppm	600ppm

The labels 0.2 / 0.4 / 0.8 and 1.6 mV/V are related to the full bridge excitation.

Note:
As in Wheatstone mode the excitation of our PICOSTRAIN products like PS08 / PS081 is reduced by one third, the sensitivity needs to be set to 1.3333mV/V in the evaluation software in order to get the here explained results.

4.2 Only Half Bridge

The full bridge is formed by bride sections / half bridges. It is also possible to connect only 1 half bridge and set the excitation in 0.2mV/V or 200ppm steps. Therefore, the switches of the connected half bridge section have to be set accordingly. Please note, that only half bridge operation is possible, but not the recommended operation mode because of less accuracy.

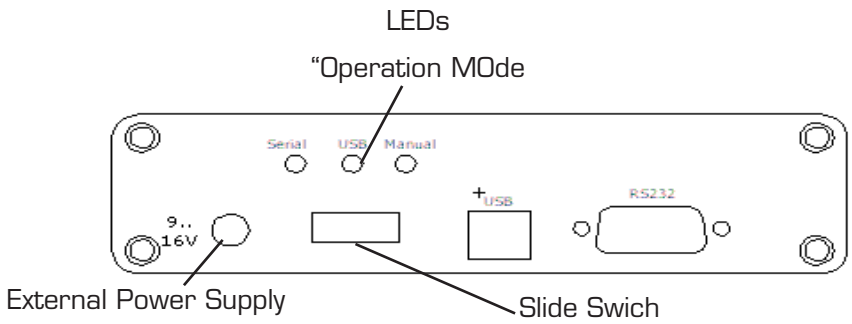
5 Operation Modes

The ALCS comes with software that simulates the mechanical switches and offers user friendly device control. It can be operated by manual switches, through an RS232 interface or through the on board USB interface. The active mode is indicated by the LEDs above the slide switch on the back of the simulator. The ALCS needs a separate power supply of 9V to 16V DC connected to the 3,5mm Jack Socket, when being operated by the manual switches or the RS232 interface. In these two modes the slide switch needs to be set to position “serial” or “manual”.

If the ALCS is operated through the USB interface set the switch to USB. The USB bus provides the power through the USB connector and no extra power supply is needed.

The default mode after power-up is the manual mode where the ALCS is controlled by the mechanical switches on the top of the housing.

Figure 5.1 Back view



5.1 Manual Mode

Altogether eight manual switches arranged in two rows enable to apply the simulated strain.

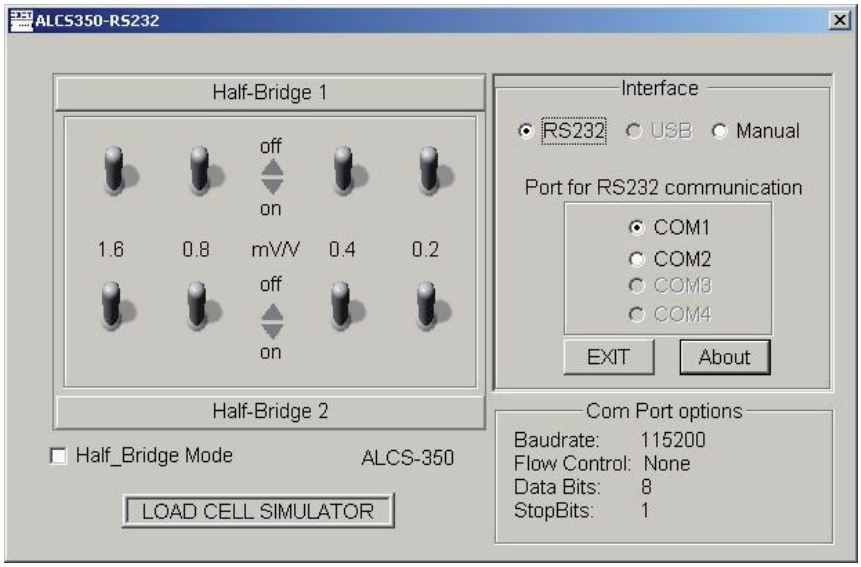
Figure 5.2 Manual switches

The simulated strain is set separately and in the opposite direction for the two bridge arms. This way the simulated strain can be set from 0 to 3 mV/V in steps of 0.1 mV/V in full bridge mode or 0.2 mV/V in half bridge mode.

5.2 RS232 Mode

In this mode the manual switches are disabled. The ALCS has to be configured by the graphical user interface by sending appropriate commands over the RS232-interface. The COM-port has to be selected in the “Interface”-section. Each switch position within a row is represented by a special ASCII command (see also section 6.3)

Figure 5.3 Grafical user interface in RS-232-mode



By default the user software runs in Wheatstone/full bridge mode. The corresponding switches on rows half bridge 1 and 2 are tied together and change simultaneously. So the same strain is simulated on both bridges arms.

In order to simulate two separate half bridges “Half-Bridge Mode“ has to be activated by checking the corresponding box underneath the switches. In half bridge mode the simulated strain can be set independently for each half bridge.

When setting half bridge mode while working with a Wheatstone- or full bridge the minimum step size can be reduced to 0.1 mV/V.

5.3 USB Mode

In USB mode the manual switches are also disabled. The ALCS has to be configured by the user interface or by sending a corresponding command via USB-interface. Each switch position within a row is represented by a special ASCII command. In this operation mode the ALCS is powered by the USB port and no separate power supply is required.

Figure 5.4 Grafical user unterface in USB-mode

By default the user software runs in Wheatstone or full bridge mode. The corresponding switches on rows half bridge 1 and 2 are tied together and change simultaneously. So the same strain is simulated on both bridges arms.

In order to simulate two separate half bridges “Half-Bridge Mode“ has to be activated by checking the corresponding box underneath the switches. In half bridge mode the simulated strain can be set independently for each half bridge.

When setting half bridge mode while working with a Wheatstone- or full bridge the minimum step size can be reduced to 0.1 mV/V.

6 RS232 User Software

6.1 Installation

Copy the folder *ALCS350_RS232* to your local hard disc drive. Run the *ALCS350-RS232.exe* software. Select the correct COM port.

6.2 Accessing the RS232 Interface in VC++

Access to the serial COM Ports of the PC for RS232 communication is accomplished by using a pre-compiled readily available Library file called *RSAPI.dll*.
This library file handles the Windows API portion of accessing the ports in Windows 98, Windows 2000, XP and Vista.
It allows the user to access the Ports by calling a pre-defined set of functions which execute a set of commands.
In order to use the Library functions provided in *RSAPI.dll* a copy of the file has to be in the working directory of the VC++ workspace and the dll has to be loaded in the user code.
The following example code defines the constant *THEDLL* to be the library file and the declaration of a function to load the dll. The function returns 1 if the load was successful and 0 if the dll could not be loaded.

```
// This code is located in the
// MyProjectDlg.cpp : implementation file
//

#define THEDLL "RSAPI.DLL"

HINSTANCE hDLL;

loaddll(char *name)
{     char s[256];
      hDLL = LoadLibrary(name);
      if (hDLL == NULL)
      {     wsprintf(s,"%s not found.",name);
            MessageBox(GetFocus(),s,"ERROR",MB_OK|MB_ICONSTOP);
            return 0;
      }
 return 1;
}
```

The *loaddll()* function has to be called once in the code before the functions for port access can be utilized.

Following is a list of the functions included in the library file *RSAPI.dll* which are relevant for the communication over the serial ports of the PC.

```
// OPENCOM
// Open serial interface
// Parameter: zero terminated character string
// Return: Zero in case of fault.
UINT opencom (char * string)
{
typedef UINT (CALLBACK * LP2INT)(char *);
LP2INT p;
p = (LP2INT)GetProcAddress (hDLL, "OPENCOM");
return p (string);
}

// CLOSECOM
// Close serial interface
// Parameter: None
// Return: None
BOOL closecom (void)
{
typedef int (CALLBACK* LP2INT)();
LP2INT p;
p = (LP2INT)GetProcAddress (hDLL, "CLOSECOM");
return p ();
}

// SENDBYTE
// Sends one byte via serial interface. Serial interface has to be opened
// before
// Parameter: One Byte (0..255)
// Return: None
int sendbyte (BYTE value)
{
typedef int (CALLBACK* LP2INT)(BYTE);
//typedef int (CALLBACK* LP2INT)(WORD);
LP2INT p; // function pointer
p = (LP2INT)GetProcAddress (hDLL, "SENDBYTE");
return p (value);
//return p (wValue);
}
// READBYTE
// Reads on Byte from serial interface. Serial Interface has to be opened
// Parameter: None
// Return: The received Byte or -1 in case of fault
UINT readbyte (void)
{
typedef UINT (CALLBACK* LP2INT)();
LP2INT p; // function pointer
```

```
p = (LP2INT)GetProcAddress (hDLL, "READBYTE");
return p ();
}


// TIMEOUT
// Sets timeout value for the serial interface. Serial interface has to be
// opened before. If no character is received within the specified time-
// interval, the READBYTE-function is aborted and -1 is returned.
// Parameter: Time interval in milliseconds (1..65535)
// Return: previous time interval in milliseconds
int timeout (WORD wTime)
{
typedef int (CALLBACK* LP2INT)(WORD);
LP2INT p; // function pointer
p = (LP2INT)GetProcAddress (hDLL, "TIMEOUT");
return p (wTime);
}

// RI ( Ring Indicator)
// Polls the RI-signal of the serial interface. The interface has to be
// opened
// Parameter: None
// Return: RI-signal state(1/0)
WORD ri (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"RI");
return p ();
}

// TXD
// Sets the serial TxD signal. Serial interface has to be opened
// Parameter: Value(0..1)
// Return: None
int txd (WORD i)
{
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"TXD");
return p (i);
}

// RTS
// Sets the RTS-signal of the RS232-Interface. Interface has to be
// opened
// Parameter: Value(0..1)
// Return: None
int rts (WORD i)
{
```

```
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,”RTS”);
return p (i);
}

// CTS
// Polls the CTS-signal of RS232+-Interface. Interface has to be opened
// Parameter: None
// Return: Signal state (1/0)
WORD cts (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,”CTS”);
return p ();
}

// DCD
// Polls DCD-Signal of the serial Interface.
// Interface has to be opened
// Parameter: None
// Return: DCD-Signal state (1/0)
WORD dcd (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,”DCD”);
return p ();
}


// DTR
// Sets DTR-signal of serial interface. Interface has to be opende
// before
// Parameter: Value(0..1)
// Return: None
int dtr (WORD i)
{
typedef int (CALLBACK* LP1INT)(WORD);
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,”DTR”);
return p (i);
}
```

```
// DSR
// Polls DSR-Signal of the RS232-Interface. Interface has to be opened
// Parameter: None
// Feedback: Signal state (1/0)
WORD dsr (void)
{
typedef WORD (CALLBACK* LP1INT)();
LP1INT p;
p = (LP1INT)GetProcAddress (hDLL,"DSR");
return p ();
}
```

These function declarations must be added to the user code at the appropriate location before they can be called to perform the desired task.
Here is a simple example that opens the COM1 Port, send two bytes and then close the port again:

```
// Function Declarations

#define THEDLL "RSAPI.DLL"

HINSTANCE hDLL;

loaddll(char *name)
{     char s[256];
      hDLL = LoadLibrary(name);
      if (hDLL == NULL)
      {     wsprintf(s,"%s not found.",name);
            MessageBox(GetFocus(),s,"ERROR",MB_OK|MB_ICONSTOP);
            return 0;
      }
 return 1;
}

opencom(char *s)
{ typedef UINT (CALLBACK* LP2INT)(char *);LP2INT p;
  p = (LP2INT)GetProcAddress(hDLL,"OPENCOM"); return p(s);
}

closecom (void)
{
   typedef int (CALLBACK* LP2INT)();
   LP2INT p;
   p = (LP2INT)GetProcAddress (hDLL, "CLOSECOM");
   return p ();
}
```

```
int sendbyte (BYTE value)
 {
    typedef int (CALLBACK* LP2INT)(BYTE);
    LP2INT p; // Pointer
    p = (LP2INT)GetProcAddress (hDLL, "SENDBYTE");
    return p (value);
 }



// Call of the functions in the user code

//   .
//   .

if(loaddll(THEDLL))     // Load dll and access Port if success
{
     opencom("COM1:115200,N,8,1");   // Open COM Port 1 with 115200 baud,
                                     // No flow control, 8 data bits and 1
                                     // stop bit

        sendbyte(0x33);                    // Send first byte
        sendbyte(0x33);                    // Send second byte

        closecom();                              // Close the COM Port again
      FreeLibrary(hDLL);                         // Delete handle to the library
}
else {      MessageBox("Failed to load RSAPI.DLL");   }

//   .
// .
```

For more information please see the source code of the RS232 example program ("ALCS350_ RS232") for the Load Cell Simulator ALCS-350-V2. It is a Dialog Project written in Visual C++ 6.0.

6.3 Set of Commands

The default mode on power-up is the manual mode where the ALCS reacts only to the manual switches. The software commands are all 1 byte commands without any error detection or checksum. In RS232 communication the byte is split up into a high and a low nibble with the high nibble being sent first.

The interface settings for RS 232 communication are:

Baud Rate: 115200

Flow Control: None

Data Bits: 8

Stop Bits: 1

6.3.1 Operation Modes

Tab le 6.1 RS232 commands for operation mode

Operation Modes	RS232 command
Switch Manual Mode on	0x33 0x33
Switch USB communication on	0x34 0x34
Switch RS232 communication on	0x35 0x35

6.3.2 Simulation

Tab le 6.2 RS232 commands for strain simulation

Simulation HB 1		
Strain	**Status**	**RS232 command**
0.2 mV/V (200ppm)	ON	0x31 0x31
	OFF	0x31 0x32
0.4 mV/V (400ppm)	ON	0x31 0x33
	OFF	0x31 0x34
0.8 mV/V (800ppm)	ON	0x31 0x37
	OFF	0x31 0x38
1.6 mV/V (1600ppm)	ON	0x31 0x3D
	OFF	0x31 0x3E
Simulation HB 2		
Strain	**Status**	**RS232 command**
200ppm	ON	0x32 0x31
	OFF	0x32 0x32
400ppm	ON	0x32 0x33
	OFF	0x32 0x34
800ppm	ON	0x32 0x37
	OFF	0x32 0x38
1600ppm	ON	0x32 0x3D
	OFF	0x32 0x3E

7 USB User Software

7.1 Installation

In order to use the *ALCS350_USB.exe* program to communicate with the ALCS-350-V2 device via the USB port of your PC, all you have to do is to copy the executable together with the files *CYUSB.inf* and *CYUSB.sys* to one and the same location on the local hard drive of your PC.
Connect the device to the PC. If Windows prompts for a driver or indicates that it needs a driver, direct the PC to use the *CYUSB.sys* driver by directing it to the *CYUSB.inf* file in the directory you copied it into.
If Windows does not prompt for a driver, it has already matched the device to a driver itself. In this case, you will need to see if the *CYUSB.sys* driver was selected and, if not, manually instruct Windows to use that driver.

1. Right-click “My Computer” and select the “Manage” menu item.
2. In the Computer Management window, select Device Manager.
3. In the right window pane, click the + icon next to Universal Serial Bus controllers.
4. Locate your device in the list and double click on it.
5. Select the Driver tab in the Properties dialog that comes up.
6. Click on the Driver Details button.

If the displayed driver file is *CYUSB.SYS*, Windows has already matched the device to this driver and you should click “OK” and cancel . If not, proceed with the remaining steps.

7. Click “OK”
8. Click “Update Driver”
9. Select Install from a list or specific location (Advanced)
10. Click “Next”
11. Select “Don’t search”. I will choose the driver to install.
12. Click “Next”
13. Click “Have Disk”
14. Click “Browse”
15. Navigate to the directory containing *CYUSB.sys*
16. *CYUSB.inf* should be automatically placed in the File name field
17. Click “Open”
18. Click “OK”
19. Click “Next”
20. Click “Finish”
21. Click “Close”

Don't re-boot your system if Windows suggests that you must. You may need to unplug and re-plug your device, however.

7.2 Accessing the USB-Port in VC++

Communication through the USB port of the PC is accomplished by using two readily available files provided by Cypress. One is *CyAPI.lib* which provides a simple, powerful C++ programming interface to USB devices. More specifically, it is a C++ class library that provides a high-level programming interface to the *CyUsb.sys* device driver. The library is only able to communicate with USB devices that are served by (i.e. matched to) this driver. In the driver for the ALCS-350-V2 supplied to you by acam the correct device is already matched to it. You don't have to do anything to alter the content of the driver or the "**.inf*"-file. You only have to install the driver on your PC.

Rather than communicate with the driver via Windows API calls such as *SetupDiXxxx* and *DeviceIoControl*, applications call simpler CyAPI-methods such as *Open*, *Close*, and *XferData* to communicate with USB devices.

To use the library, you need to include the header file *CyAPI.h*, in files that accesses the *CCyUSBDevice* class. In addition, the statically linked *CyAPI.lib* file must be linked to your project.

You link the file to your project by clicking on the "Project" Menu in Visual Studio and selecting "add to project -> Files". In the explorer window find the *CyAPI.lib* file and double click on it.

The library employs a Device and EndPoints use model. To use the library you must create an instance of the *CCyUSBDevice*-class using the new keyword.

A *CCyUSBDevice* object knows how many USB devices are attached to the *CyUsb.sys* driver and can be made to abstract any one of those devices at a time by using the "Open"-method.

An instance of *CCyUSBDevice* exposes several methods and data members that are device-specific, such as *DeviceName*, *DevClass*, *VendorID*, *ProductID*, and *SetAltIntfc*.

With the following code it is possible to look for a specific device and determine if it is connected to the PC:

```
/////// First we create an instance of the CCyUSBDevice class ///////
CCyUSBDevice *USBDevice = new CCyUSBDevice();

// Look for our device with VID = 194e, PID = 1001
int devices = USBDevice->DeviceCount();
int vID, pID;

int d = 0;
do{
      USBDevice->Open(d); // Open automatically calls Close( ) if necessary
      vID = USBDevice->VendorID;
      pID = USBDevice->ProductID;
      d++;
```

```
      } while ((d < devices ) && ((vID != 0x194e) || (pID != 0x1001)));

if((vID != 0x194e) || (pID != 0x1001))
{
      USBDevice->Close();     // This is not ALCS Device. Close the port.
      wsprintf(s,"Load Cell Simulator not found on USB Port");
      MessageBox(s,"ALCS-350-V2 USB");
}
delete USBDevice;
```

When a *CCyUSBDevice* object is open to an attached USB device, its endpoint members provide an interface for peforming data transfers to and from the device's endpoints. Endpoint-specific data members and methods such as *MaxPktSize*, *TimeOut*, *bIn*, *Reset* and *XferData* are only accessible through endpoint members of a *CCyUSBDevice* object.

Communication to the Load Cell Simulator is performed by using control transfers only. *CCyControlEndPoint* is a subclass of the *CCyUSBEndPoint* abstract class. Instances of this class can be used to perform control transfers to the device. Control transfers require 6 parameters that are not needed for bulk, isoc, or interrupt transfers. These are:

- Target
- ReqType
- Direction
- ReqCode
- Value
- Index

The parameter "Value" contains the command for the Load Cell Simulator.
All USB devices have at least one Control endpoint, endpoint zero. Whenever an instance of CCyUSBDevice successfully performs its Open( ) function, an instance of CCyControlEndPoint called ControlEndPt is created. Normally, you will use this ControlEndPt member of CCyUSBDevice to perform all your Control endpoint data transfers.
The following example shows the creation of an instance of the CCyUSBDevice class and the transfer of a command to the ALCS-350-V2 via USB:

```
#include "CyAPI.h"
//          .
//          .
CCyUSBDevice *USBDevice = new CCyUSBDevice();   // Create an Instance of the
// CCyUSBDevice class
```

```
// Just for typing efficiency
CCyControlEndPoint *ept = USBDevice->ControlEndPt;
                              // Create Pointer to Control Endpoint in
                              // order to easier set parameters

if(USBDevice->IsOpen())       // A USB device has to be present
{
      ept->Target = TGT_DEVICE;
      ept->ReqType = REQ_VENDOR;
      ept->Direction = DIR_TO_DEVICE;
      ept->ReqCode = 0xB0;
      ept->Value = 0x44;            // Command to switch interface to USB comm.
      ept->Index = 0;

      OK = ept->XferData(0, buflen);       // Send the control package

      ept->Value = 0x11;            //applies 200 ppm on halfbridge 1
      OK = ept->XferData(0, buflen);       // Send the control package

      ept->Value = 0x21;            // applies 200 ppm on halfbridge 2
      OK = ept->XferData(0, buflen);       // Send the control package
            .
            .
}

delete USBDevice;
```

For more information please see the source code of the USB example program (“ALCS350_USB”) for the Load Cell Simulator ALCS-350-V2. It is a Dialog Project written in Visual C++ 6.0.

7.3 Set of Commands

The default mode on power-up is the manual mode where the ALCS reacts only to the manual switches. The software commands are all 1 byte commands without any error detection or checksum.

7.3.1 Operation Modes

Table 6.1 USB commands for operation modes

Operation Modes	USB command
Switch Manual Mode on	0x33
Switch USB communication on	0x44
Switch RS232 communication on	0x55

7.3.2 Simulation

Tab le 6.2 USB commands for strain simulation

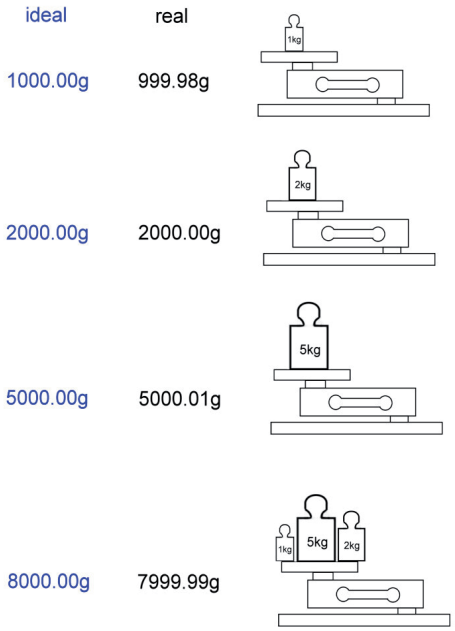
Simulation HB 1		
Strain	**Status**	**USB command**
0.2 mV/V (200ppm)	ON	0x11
	OFF	0x12
0.4 mV/V (400ppm)	ON	0x13
	OFF	0x14
0.8 mV/V (800ppm)	ON	0x17
	OFF	0x18
1.6 mV/V (1600ppm)	ON	0x1D
	OFF	**0x1E**
Simulation HB 2		
Strain	**Status**	**USB command**
200ppm	ON	0x21
	OFF	0x22
400ppm	ON	0x23
	OFF	0x24
800ppm	ON	0x27
	OFF	0x28
1600ppm	ON	0x2D
	OFF	0x2E

8 Use the ALCS for linearity tests

8.1 How it principally works

Linearity of an ordinary scale with a load cell as sensor can be determined by charging the scale with different weights, adding their values together and see if this is the same result when putting on the corresponding weights together. Basically this is how it is done with the load cell simulator, too. Several strains (=weights) are given to the electronics and the results are written down. The results of the individual strains can be summarized and then compared with the results obtained by setting the corresponding strain. The following picture shows the principle:

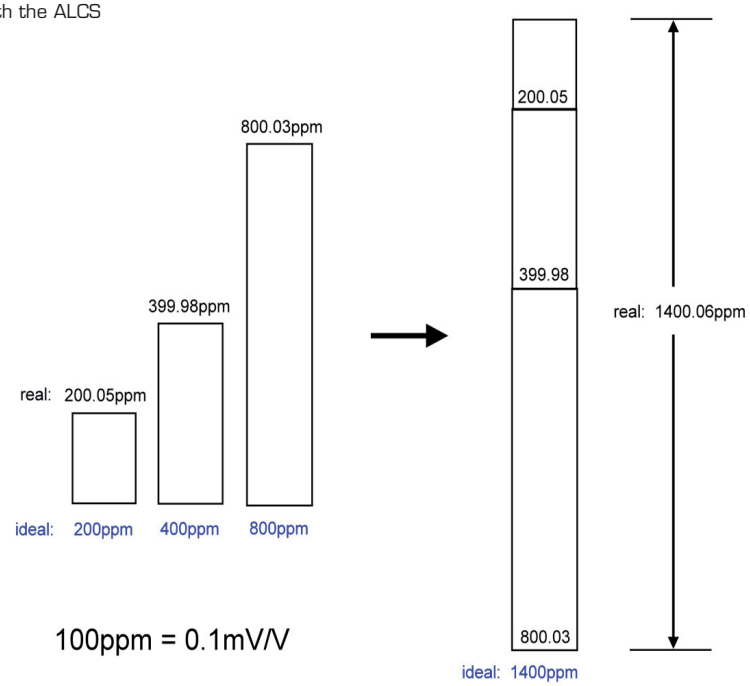
Figure 7.1 Principle of linearity tests



Ideally, the weights match exactly and also the sum of the weights together. In reality there may be some deviations from the ideal weight, e.g. 999.98g instead of 1000.00g. Important is, that the sum of the weights corresponds to the real values of the weights summed up together.

In practise, the weights have a quite high accuracy, higher than shown in the example so that the deviation will not be as high as shown above. Nevertheless, the linearity of the sensor – here a load cell – is limited. Therefore, the linearity of the PICOSTRAIN electronics can not be investigated sufficiently, as the linearity of the electronics is much higher than the one of the load cell. Here comes the ALCS to help out: with the load cell simulator the linearity can be investigated up to 1:200000 (± 0.01µV/V) The way of making the measurement is very similar, the following picture shows what is basically done:

Figure 7.2 Simulating strain with the ALCS



The strain is applied by switching the ALCS (0.2 / 0.4 / 0.8 / 1.6 mV/V). The strains will differ much more than the weights do, so instead of exactly 200.00ppm you will get 200.05ppm for example. Furthermore, the strain precision also varies with the wiring – the accuracy is different from PICOSTRAIN wiring (± 0.1µV/V) to Wheatstone wiring (± 0.6µV/V). That means in practise, that the deviation to the nominal value will be obvious like in the following example:

Selected setting:	1600.00 µV/V
Real value with PICOSTRAIN wiring:	1600.08 µV/V
Real value with Wheatstone wiring:	1600.58 µV/V

Though the strain has its deviation from the nomimal value, it's stable and repeatable. As always the same resistors are switched internally, the linearity can be proven by adding up the strains as you normally do it with weights. The strain can be set with 2 post decimal positions, e.g. 2000.00ppm and therefore investigations about the linearity can be done up to 1:200000 (related to 2mV/V), which is much higher than with a load cell.

Please see also the next chapter about differences between Wheatstone and PICOSTRAIN wiring and its impact on the linearity investigations.

8.2 Difference between Wheatstone and PICOSTRAIN wiring

The calibration of the ALCS is complex process where the main goal is mainly to reduce the parasitic resistances when switching the strain. Even with high precision resistors the accuracy cannot be increased infinitely and is more than that dependend on the mode (PICOSTRAIN or Wheatstone wiring). As the calibration is optimized for PICOSTRAIN wiring, the accuracy is best in this mode and therefore the best linearity is achieved also in this mode. The accuracy differs as follows:

Accuracy with PICOSTRAIN wiring: ± 0.1µV/V
Accuracy with Wheatstone wiring: ± 0.6µV/Vs

Due to the different accuracy also the linearity investigations show different results. The best linearity can be shown in PICOSTRAIN mode, here a linearity of ± 0.01µV/V can be achieved. Compared to A/D converters this corresponds to ± 1.25ppm linearity (with PGA-setting of 128 and related to 2mV/V strain). This linearity is tremendously good and gives you a sensor which is more than 10 times better than an ordinary load cell. Therefore PICOSTRAIN wiring is the preferred mode to prove the good linearity of our PICOSTRAIN products like PS08 and PS081.

With Wheatstone wiring the linearity of the ALCS decreases to typ. ± 0.04 µV/V. This is not as good as in PICOSTRAIN mode, but nevertheless 2-3 times better than good load cells and approximately in the range of good A/D converters. In other words, you can still prove a good linearity of our PICOSTRAIN products, but you will not see the maximum linearity (which can only be seen with PICOSTRAIN wiring).

Note:
Please keep in mind, that the linearity of the ALCS is much higher than that of an ordinary load cell (10 times higher in PICOSTRAIN wiring, 2-3 times higher in Wheatstone wiring). So you have a very good device in your hands to test the linearity of PICOSTRAIN products like the PS08 or PS081. Nevertheless, also the linearity of ALCS is limited so that it can be assumed that the linearity of our products is even higher. In other words, the linearity investigations are limited by the sensor and not by the electronics itself.

8.3 Determine non-linearity (with Excel-Spreadsheet)

For making the tests we recommend to use the evaluation software of PS08 / PS081. Connect the ALCS to the electronics and start the software. Load the configuration "PS081-Linearity.cfg" (available from acam). Important is the Mult_PP setting of 1.25 for PICOSTRAIN wiring and 1.12 for Wheatstone wiring. Also, the cycle time setting has an significant influence and should be set in the range from 85 (=170µs) to 90 (=180µs).

The linearity can be determined by doing the following steps (using Excel to calculate the results):

1. Fill in 1 column all values measured with ALCS, measure all settings from 200ppm to 3000ppm (200ppm / 400ppm / 600ppm / … / 2800ppm / 3000ppm)

Table 1 Measured values

Temperature: 25°C	**Mult_PP=1.25**
Measured Strain [ppm]	
200.48	
400.54	
601.02	
800.59	
1001.07	
1201.13	
1401,61	
1600,89	
1801,38	
2001,43	
2201,91	
2401,47	
2601,96	
2802,02	
3002,49	

2. The values will not match exactly the nominal values (e.g. 200.48ppm instead of 200.00ppm). The accuracy of ALCS is simply not high enough to set the nominal values exactly. However, it makes sense to scale the values to a maximum of 3000ppm, in our example that means to multiply the column by 0.99917 as shown in table 2.

Table 2 Error calculation and scaling

Measured strain [ppm]	Calculated strain [ppm]	Linearity Error [ppm]
200.31		
400.21		
600.52	600.52	0.00
799.93		
1000.24	1000.24	0.00
1200.13	1200.13	0.00
1400.45	1400.45	0.00
1599.56		
1799.89	1799.88	
1999.77	1999.77	0.00
2200.08	2200.08	0.00
2399.48	2399.49	-0.01
2599.80	2599.80	0.00
2799,70	2799.70	0.00
3000,00	3000.01	-0.01
Scaling Factor: 0.99917		

3. You obtain the values as shown in the column 'measured:'. In the column calculated, the values to expect are put. That means, the value 600.52 is a combination of 200.31 and 400.21 for example. All calculated values are on the basis of the 200ppm / 400ppm / 800ppm and 1600ppm values, as these are the basic strains (like the individual weights in the example before).
4. Once you have the column 'calculated' it is easy to determine now the linearity-error, shown in the last column. In our measurement this was only ±0.01 µV/V with PICOSTRAIN wiring and ± 0.04 µV/V with Wheatstone wiring.

8.4 Results of linearity measurement with PS081

We investigated the non-linearity of PS081 in June 2009 here at acam by using the ALCS-V2. As mentioned earlier we got the very low linearity deviation of ±0.01µV/V in PICOSTRAIN mode, which corresponds to 1.25 ppm linearity at A/D converters (with PGA-setting of 128 and related to 2mV/V strain). This is a tremendous high linearity compared to the one of nowadays A/D converters. Ordinary A/D converters have a linearity in the range of ±0.12µV/V (15ppm), better ones have a linearity of ±0.04µV/V (5ppm).

Please see the following diagrams showing the linearity as 'best-fit straight line' over temperature and voltage:

Figure 1 Linearity of PS081 over temperature

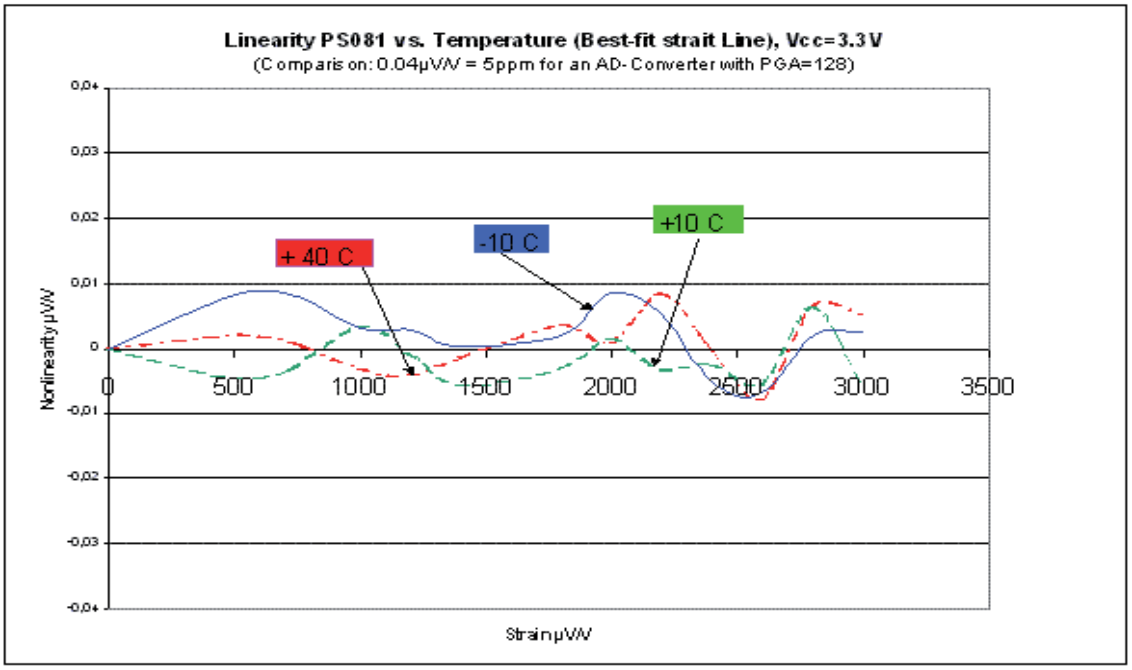
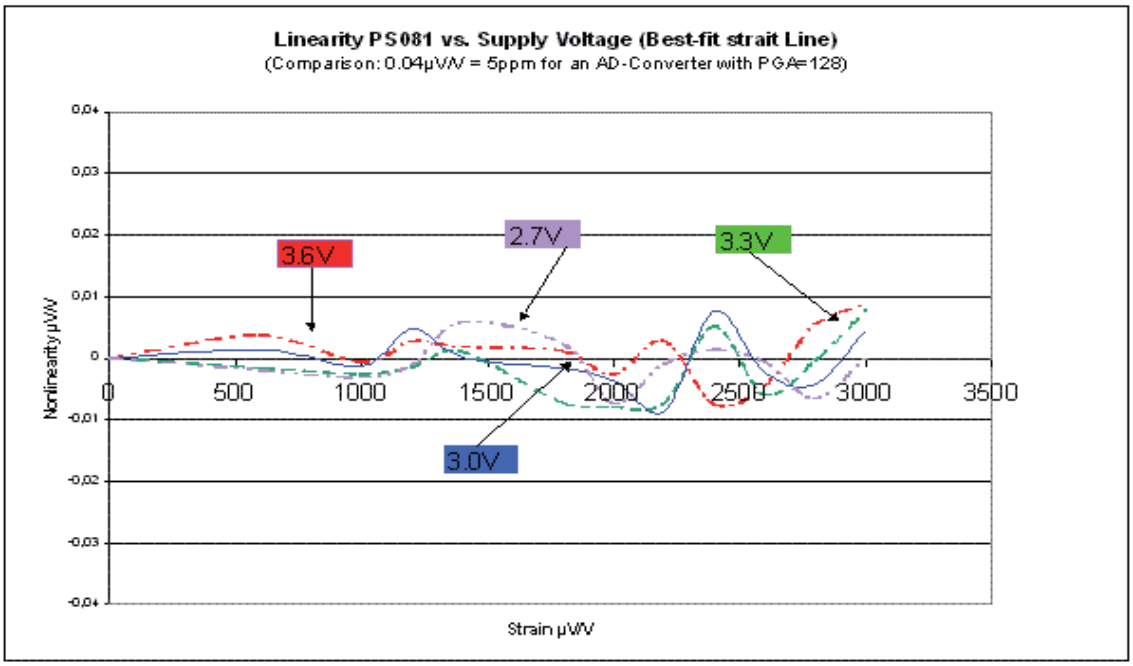


Figure 2 Linearity over supply voltage



Note

The linearity is best in the supply voltage range between 3.0V to 3.6V. Furthermore, the linearity varies over the configured cycle time, the diagrams above were taken with cycle time setting 85 (=170µs)

9 Determine Gain Correction factor (Mult_PP) of your sensor

9.1 What is Mult_PP?

The PICOSTRAIN electronics like PS08 or PS081 has a gain error which is very low, approximately in the range of –4ppm/K. By following some hardware recommendations and setting the Mult_PP (gain correction factor) properly, this remaining gain error can be reduced to < 1ppm/K.

Meanwhile the gain drift of an A/D converter comes mainly from the mismatching of the resistors of the operational amplifier, the source of the gain drift at PICOSTRAIN is entirely different as we don't have a preamplifier at all. The gain drift at PICOSTRAIN comes mainly from the delay time of the comparator and this delay time mainly depends on the values of the low pass resistor and capacitor as well as their material. The recommended values are:

3k3 for the collector resistor

3n3 for the parallel capacitor, must be COG material

If the values are changed, the Mult_PP value changes as well. Of course there is also a dependency on which comparator is used, internal or external. So generally there are standard values we recommend, but they are based also on the recommended hardware and can slightly differ with your hardware. Therefore, the determination of the proper Mult_PP factor for your hardware makes sense, especially because it can be done very easily with the ALCS-V2.

9.2 Determine Mult_PP

Basically there are 2 temperature runs with 2 Mult_PP settings necessary to determine the proper Mult_PP factor. Starting from the recommendations given from acam, the Mult_PP factor is set at first to:

Mult_PP = 1.25* (PICOSTRAIN wiring)

Mult_PP = 1.12* (Wheatstone wiring)

* our measurements showed, that the Mult_PP settings with ALCS are slightly lower than with the plug-in modules of the evaluation kit, e.g. 1.25 instead of 1.28 with PICOSTRAIN wiring.

So you start the 1st run with these values, recording the gain drift from e.g. 40°C to 10°C (with maximum strain of 3000ppm). Let's assume the gain drift would be –2ppm/K (GD1). As the gain decreases with rising temperature, Mult_PP needs to be increased a bit, e.g. from 1.25 to 1.35.

With this second setting for Mult_PP = 1.35 (MPP2) you start the 2nd run and determine the gain drift again. Let's assume it would be 4ppm/K (GD2). Now you can calculate the proper Mult_PP factor for the hardware according to the following formula:

$$MPPN = MPP1 - \frac{GD1\,x\,(MPP2 - MPP1)}{GD2 - GD1}$$

MPP1 = Mult_PP 1, was here 1.25
MPP2 = Mult_PP 2, was here 1.35
GD1 = Gain drift 1, was here –2ppm/K
GD2 = Gain drift 2, was here 4ppm/K

In this example, the correct Mult_PP factor would be 1.2833.

Please see also our Application Note 018 "Metrological Investigations of PS08", p. 15/16 for further explanation of Mult_PP and how to determine it.

10 Limits of ALCS

In this section we want to point out what ALCS-V2 is suited for and where the limits of the device are.

The ALCS is perfectly suited for doing the following measurements:

- Linearity measurement of PICOSTRAIN products (PICOSTRAIN wiring shows best results)
- Linearity measurements of PICOSTRAIN products over temperature and voltage
- Determination of proper Mult-PP factor for your hardware
- Can be used with all PICOSTRAIN products and all conventional A/D converters

The limitations of ALCS are as follows:

- Accuracy of strain is limited
- Linearity and accuracy is different to the selected wiring (PICOSTRAIN or Wheatstone)
- Influences of gain and offset drift of ALCS are visible

Note

It is recommended to have the ALCS in a temperature stable environment, preferably at room temperature. You can of course vary the temperature of the electronic unit / PCB, but it is NOT recommended to put the ALCS into the temperature drift chamber nor to have any changes in temperature when placing it outside the chamber. The best is to insulate the ALCS thermically, that means to put it into a box which protects agains temperature changes.
The background is the gain and offset drift of the precision resistors used in the ALCS. Although it is equipped with the best available resistors in regards of accuracy and drift, they cannot reach the low drift values of a converter like PS08 or PS081. So if the temperature of ALCS varies, you will see a high offset and gain drift coming from the ALCS which makes it impossible to investigate the PICOSTRAIN electronics.

11 Contact

Headquarter Germany	acam-messelectronic gmbh	Am Hasenbiel 27 76297 Stutensee-Blankenloch	Tel: +49 (0) 7244 7419-0 Fax: +49 (0) 7244 7419-29 support@acam.de www.acam.de

European Distributors

Belgium (Vlaanderen)	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.cebselect.nl
Estonia	FINTRONIC OY	Ruosilantie 14 B 00390 Helsinki	Tel: +358 9 2512 7770 Fax +3358 9 879 7770 www.fintronic.fi fintronic@fintronic.fi
Finland	FINTRONIC OY	Ruosilantie 14 B 00390 Helsinki	Tel: +358 9 2512 7770 Fax +3358 9 879 7770 www.fintronic.fi fintronic@fintronic.fi
France	microel (CATS S.A.)	Immeuble „Oslo“ - Les Fjords 19, avenue de Norvège Z.A. de Courtaboeuf - BP 3 91941 LES ULIS Cedex	Tel. : +33 1 69 07 08 24 Fax : +33 1 69 07 17 23 commercial@microel.fr www.microel.fr
Great Britain	2001 Electronic Components Ltd.	Stevenage Business Park, Pin Green Stevenage, Herts SG1 4S2	Tel. +44 1438 74 2001 Fax +44 1438 74 2001 c.jones@2k1.co.uk www.2k1.co.uk
Hungary	ChipCAD ELEKTRONIKAI DISZTRIBUCIÓ KFT	Tuzolto u. 31. 1094 BUDAPEST	Tel: +36 231 7000 Fax: +36 231 7011 szfarkas@chipcad.hu www.chipcad.hu
Italy	DELTA Elettronice s.r.l	Via Valpraiso 7/A 20144 Milano	Tel: +39 02 485 611 1 Fax: +39 02 485 611 242 flabrace@deltacomp.it www.deltacomp.it
Latvia	FINTRONIC OY	P. O. Box 99 1099 Riga	Tel: +371 297 18384 Fax: +371 671 8651 www.fintronic.fi harijs@fintronic.lv
Lithuania	FINTRONIC OY	Pramones str. 21 78136 Siauliai	Tel: +370 612 52525 Fax: + 370 41 419373 www.fintronic.fi modestas@fintronic.lt
Netherlands	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.censelect.nl
Poland	W.G. Electronics Sp.z o.o.	ul. Modzelewskiego 35 02-679 WARSZAWA	Tel: +48 22 847 9720, +48 22847 9721 Fax: +48 22 647 0642 tgornicki@wg.com.pl www.wg.com.pl

Russia	Galant Electronics, Ltd.	100, Prospekt Mira, Moscow, 129626, Russia	Tel.\Fax: +7-495-987-42-10, Tel.: +7-095-107-19-62 Mobile +7-916-993-67-57 leonid-k@galant-e.ru www.galant-e.ru
Switzerland	Computer Controls AG	Neunbrunnenstr. 55 8050 Zürich	Tel: +41-1-308 6666 Fax: +41-1-308 6655 info@ccontrols.ch www.ccontrols.ch
Ukraine	Filur Electric	off. 700, 2A Maxima Krivono-sa str. P.O.B. 180 Kiew, 03037	Tel: +380 44 2488812 Fax: +380 44 2493477 asin@filur.kiev.ua www.filur.net

American Distributors

United States of America	Transducers Direct, LCC	264 Center Street Miamiville, Ohio 45147	Tel: 513-583-9491 Fax: 513-583-9476 sales@acam-usa.com www.acam-usa.com

Asian Distributors

India	Brilliant Electro-Sys. Pvt. Ltd.	4, Chiplunker Building, 4 Tara Temple Lane, Lamington Road, Bombay – 400 007	Tel: +91 22 2387 5565 Fax: +91 22 2388 7063 besimpex@vsnl.net www.brilliantelectronics.com
Israel	ArazimLtd.	4 Hamelacha St. Lod P.O.Box 4011 Lod 71110	Tel: 972-8-9230555 Fax: 972-8-9230044 info@arazim.com www.arazim.co.il
Japan	DMD–Daiei Musen Denki Co., Ltd.	10-10, Sotokanda, 3-Chome, Chiyoda-Ku Tokyo 101-0021	Tel: +81 (0)3 3255 0931 Fax: +81 (0)3 3255 9869 sales@daiei-dmd.co.jp www.daiei-dmd.co.jp
P.R. China	Broadtechs Technology Co. Ltd.	3C JinHuan Building, 489 Xiang Yang Road South Shanghai, 200031	Tel.: +86-21-54654391 Fax: +86-21-64454370 info@acam-china.com www.acam-china.com
	Shenzhen SECOM TELECOM Co., Ltd.	Headquarter: 32/F, Block A, ShenFang Plaza, No. 3005 Renmin Nan Rd. Shenzhen 518001 Nanjing Office: Beijing Office: Qingdao Office: Shanghai Office: Chengdu Office: Wuhan Office: Xi'An Office: Xiamen Office:	Tel.: +86 755 25155888 Fax: +86 755 25155880 zorro_huang@secomtel.com www.secomtel.com Tel.: +86 25 84552900 Tel.: +86 10 82336866 Tel.: +86 86 532 85899132 Tel.: +86 21 52371820 Tel.: +86 28 82981751 Tel.: +86 27 87322726 Tel.: +86 29 88323435 Tel.: +86 592 5806950
South Korea	SamHwa Technology Co., Ltd.	#4 4F Kyungwon building, 416-6 Jakjeon-dong GYEYANG-GU, INCHEON 407-060	Tel: +82 32 556 5410 Fax: +82 32 556 5411 www.isamhwa.com minjoonho@isamhwa.com