ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Agilent E1415A / E1413C* at our website: *Click HERE*

This literature was published years prior to the establishment of Agilent Technologies as a company independent from Hewlett-Packard and describes products or services now available through Agilent. It may also refer to products/services no longer supported by Agilent. We regret any inconvenience caused by obsolete information. For the latest information on Agilent's test and measurement products go to:

www.agilent.com/find/products

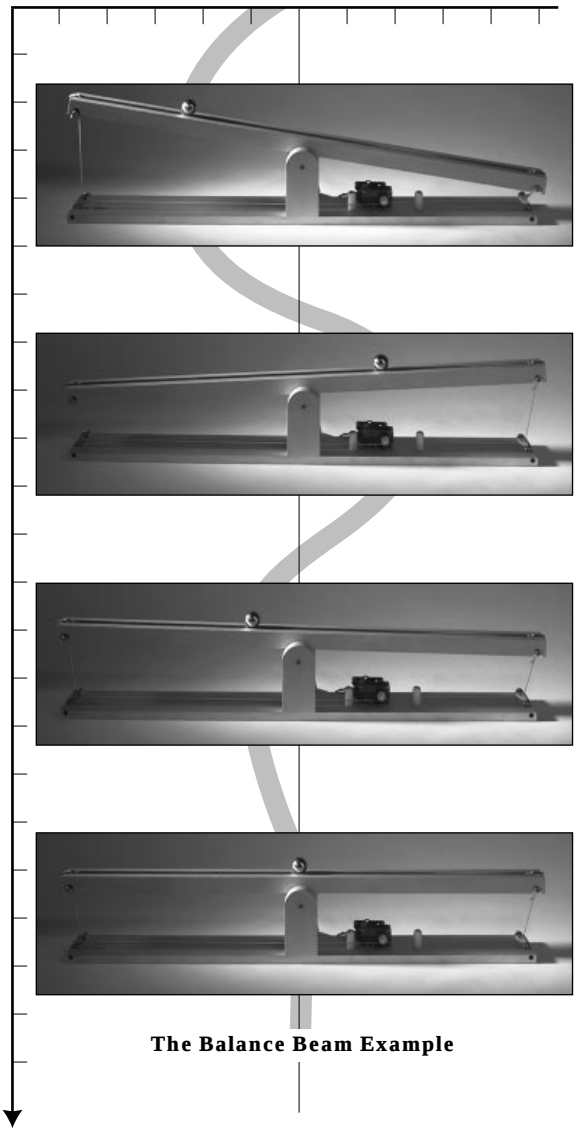
Or in the U.S., call Agilent Technologies at 1-800-452-4844 (8am–8pm EST)

Agilent Technologies
Innovating the HP Way

HP E1415A Algorithmic Closed Loop Controller

HEWLETT® PACKARD

Product Note E1415-1



The Balance Beam Example

Quickly implement control loops ranging from PID to complex custom algorithms that run stand-alone in a single VXI Module

- ❏ Up to 32 control loops
- ❏ Multiple inputs and outputs per loop: analog, digital, discrete, count
- ❏ Up to 2.5 kHz update rate
- ❏ User specified control algorithms written in C, and built-in PID algorithms
- ❏ Control algorithms can run stand-alone, minimizing load on VXIbus and embedded computer
- ❏ On-the-fly change of coefficients and setpoints for easy tuning
- ❏ Algorithm operation can be monitored through Current Value Table (CVT).
- ❏ Loop history can be read from the 65,024 element FIFO buffer



Contents

- Introduction 2
- Description 4
- Simple Examples 8
  - Temperature Bath 8
  - Balance Beam 10
- Complete Listings for Examples 14
- Built-in PID Algorithms 19

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introduction

This product note introduces you to the HP E1415A Algorithmic Closed Loop Controller module. After reading this, you'll have an insight into the flexibility and power the HP E1415A provides as a solution to process control and monitoring applications.

Background

Figure 1 shows some of the elements of a complex control loop. A control algorithm receives a mix of analog and digital signals that represent the state of a running process. The algorithm calculates values appropriate to keep the process operating according to a setpoint value. These calculated values are converted to electrical levels that actually control the process.

The signals measured may be voltage, resistance, temperature, pressure, frequency, or digital states. The control signals can be analog voltage, current, digital pulses or digital states. As shown in Figure 1, a single algorithm may calculate multiple control outputs from multiple process inputs.

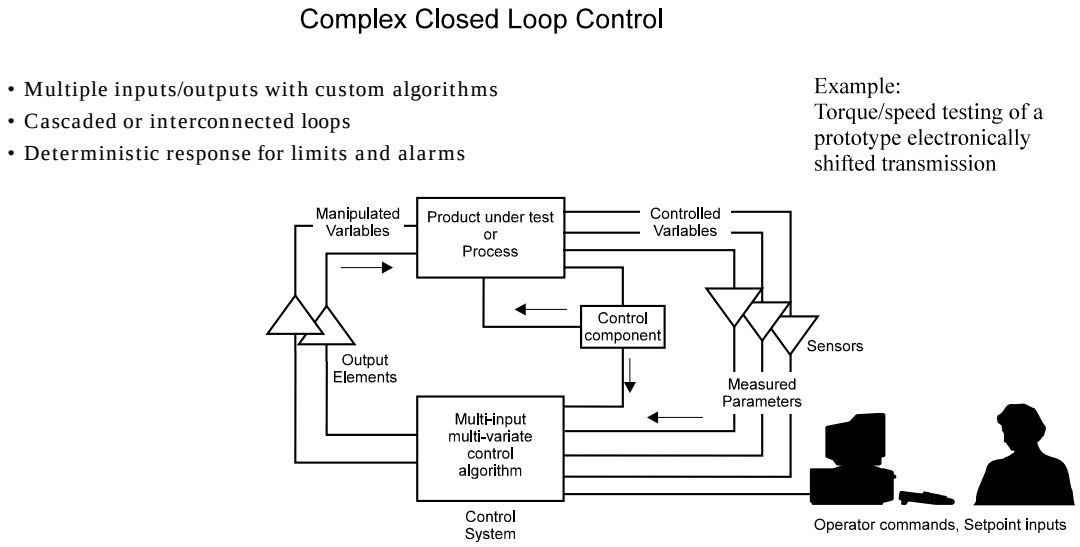


Figure 1

The number and mix of signal types typically means that complex custom loop controllers must integrate several input and output instruments with a control computer. In order to provide deterministic response to process inputs (such as an out-of-limits alarm condition), a real-time computer is often required. Figure 2 shows a typical complex loop controller configuration using individual VXIbus instruments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

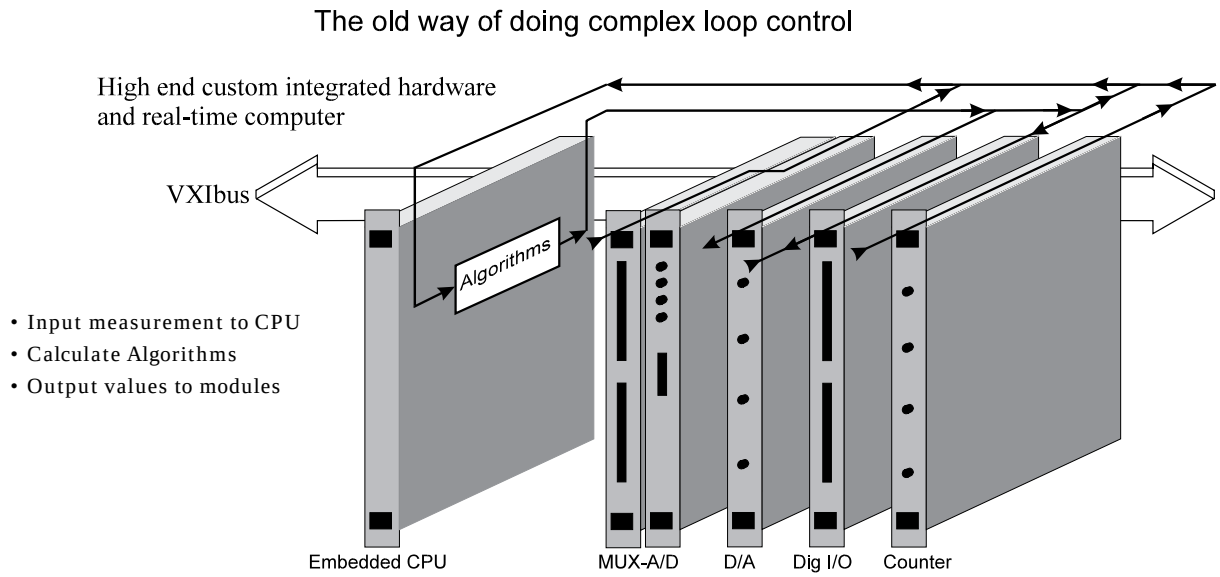
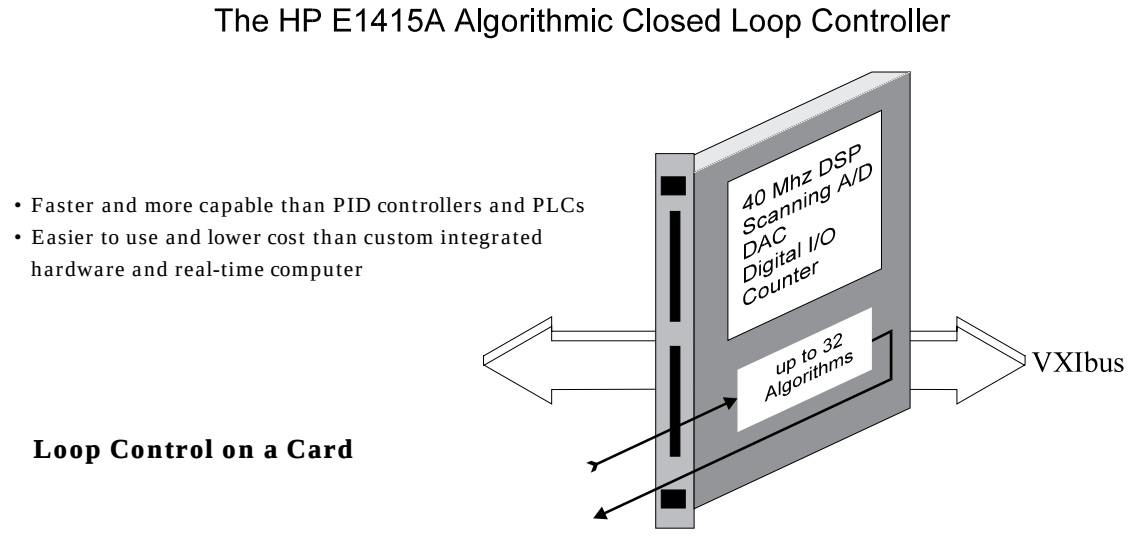


Figure 2

A better Way

Now you can implement complex custom control without having to integrate several individual instruments with a real-time computer.

The HP E1415A provides a choice of configurable input and output channels, and a DSP chip to execute algorithms, all inside a single slot C-size VXI module.



Loop Control on a Card

Figure 3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The HP E1415A is a self contained data acquisition and control platform in a single C-size VXIbus module. Once configured for operation and started (using its SCPI command set), the module's DSP chip executes all currently defined algorithms. The algorithms have access to data acquired from input channels, and from these data they generate values that control the analog and digital output channels. It is the calculation and decision making capability provided by its Algorithm Language that makes the HP E1415A a unique closed loop controller. By placing the control "computer" (DSP chip) inside the data acquisition and control instrument, response to changing input data can be fast and deterministic without burdening an embedded computer with this task.

Features

The HP E1415A executes up to 32 independent closed loop algorithms. Input to, and output from the algorithms is through 64 channels that can be configured as 8-channel groups of analog input, analog output, and digital I/O. These are provided by optional Signal Conditioning Plug-on cards (SCPs) mounted inside the HP E1415A. The HP E1415A can accommodate up to 8 SCPs. The analog input and output SCPs are self calibrated by the HP E1415A on demand. Figure 4 shows the HP E1415A's architecture in simplified form.

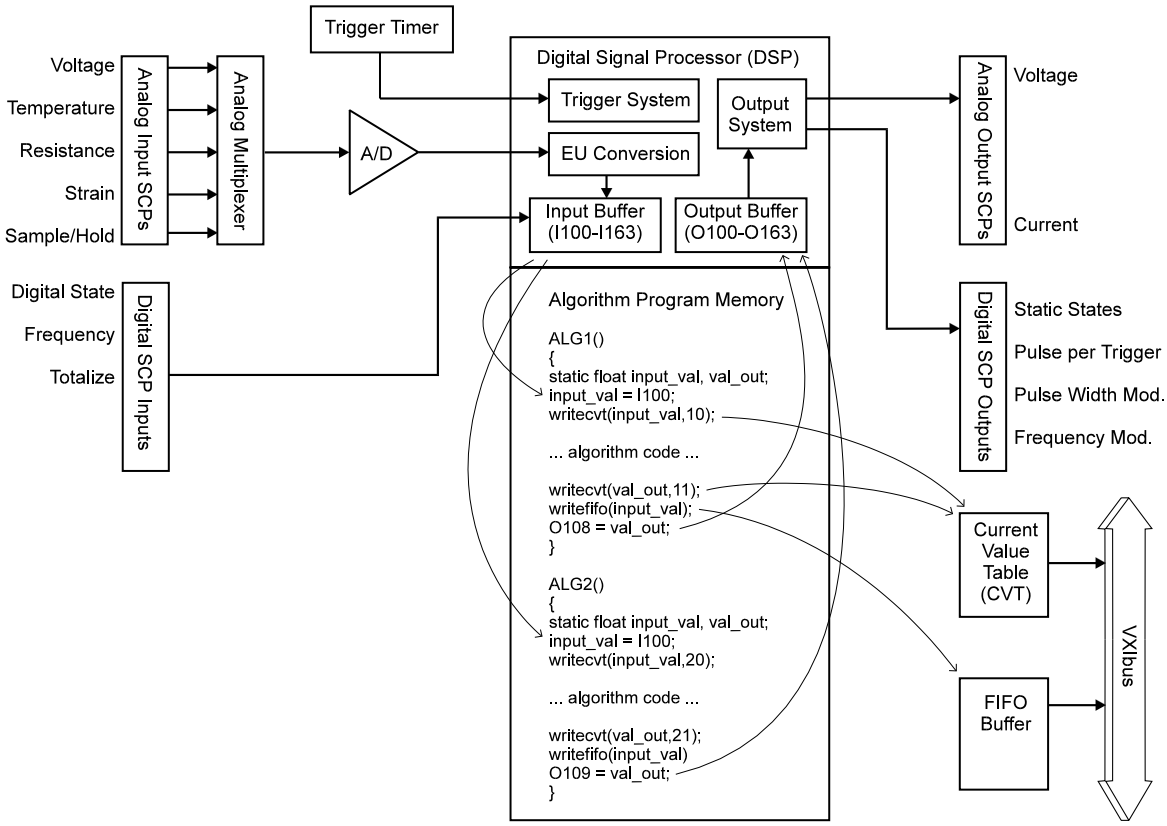


Figure 4 - Simplified HP E1415A Architecture

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Operation Sequence

See Figure 5 for this discussion. Each time the HP E1415A is triggered, the current values of analog and digital input channels are placed in the Input Buffer (phase 1). Between the input scan time and algorithm execution, the VXI controller can send values to update algorithm variables (phase 2). In phase 3 all currently defined algorithms are executed. They take their input values from the Input Buffer, calculate output values and send those to the Output Buffer. Finally, in phase 4 the Output Buffer values are sent to the analog and digital output channels.

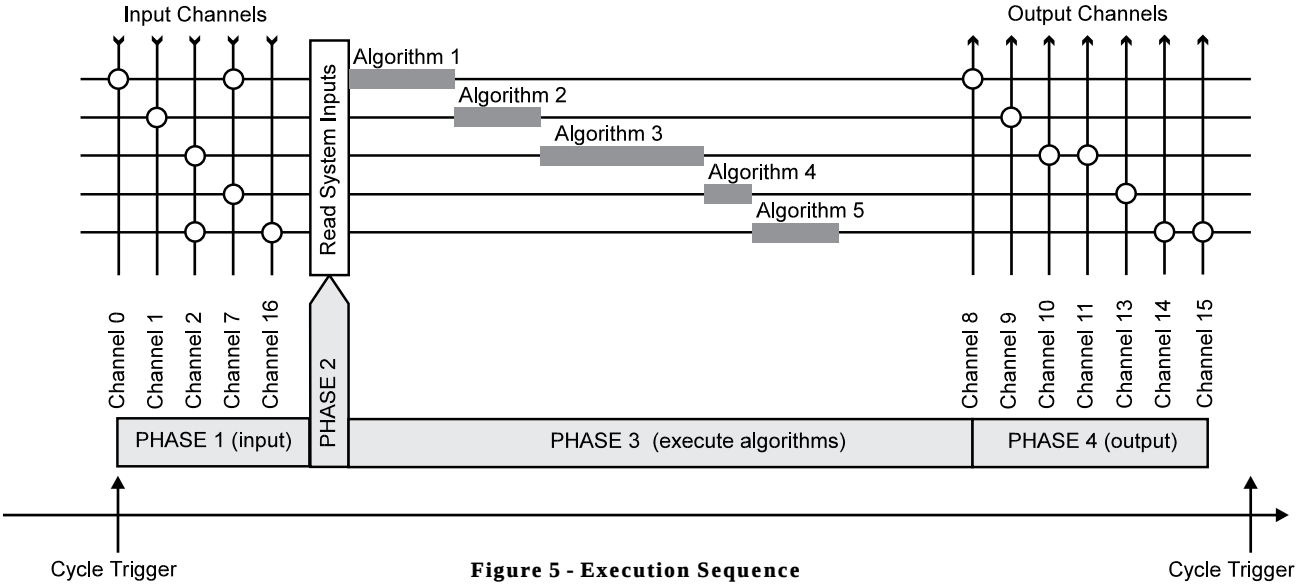


Figure 5 - Execution Sequence

Built-in PID Algorithms

Because many control applications can easily be handled by a traditional PID algorithm, the HP E1415A provides two different built-in PID algorithms. All that is required to define one of these built-in PID algorithms is to execute the SCPI commands:

ALGorithm:DEFine'<alg_name>','PIDA(<inp_chan>,<outp_chan>)'
or
ALGorithm:DEFine '<alg_name>','PIDB(<inp_chan>,<outp_chan>,<alarm_chan>)'

By specifying a pre-defined global variable for <inp_chan> or <outp_chan> instead of an input channel (I100) or output channel (O108), these PIDs can then communicate with a custom written algorithm.

Functional block diagrams for the "PIDA" and PIDB" algorithms are provided starting on page 19. A program listing for PIDA is also included.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating Algorithms

The key feature of the HP E1415A is its unique method of executing algorithms. Here we'll discuss the HP E1415A's Algorithm Language, how to create algorithms, and how to load them into the HP E1415A.

The Algorithm Language

The HP E1415A's algorithm language will already be familiar to many programmers since it is ANSI C. The Algorithm Language is a subset of C that is tightly focused to perform loop control. Figure 6 illustrates the C language elements that the HP E1415A implements.

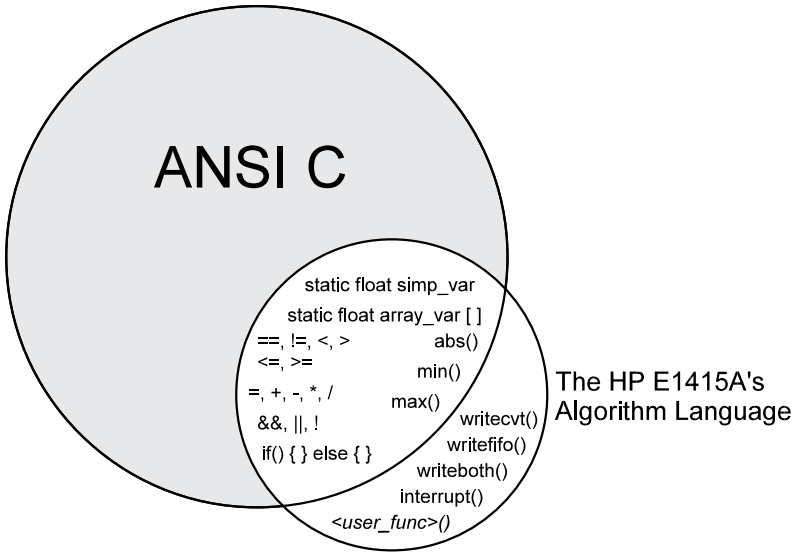


Figure 6

User Defined Functions

Note that at the bottom of Figure 6 there is the <*user_func*>() extension. To maintain high algorithm execution speed, common transcendental and custom non-linear functions are evaluated outside the algorithm execution environment by software supplied with the HP E1415A. These values are stored into piece-wise linearly interpolated tables. During algorithm execution these "look-up" tables return a value much faster than if a transcendental function's equation were arithmetically evaluated using a power series expansion. The HP E1415A supports up to 32 user defined functions.

Writing Custom Algorithms

Like any C language program, you write your algorithm as a program source (ASCII text). Your algorithm source code is then translated into executable code by the HP E1415A's driver when you execute the command:

ALGorithm:DEFine '<alg_name>','<your algorithm source string here>'

You execute one ALG:DEFINE command for each algorithm you load into the HP E1415A.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Overall Programming Sequence

In addition to defining algorithms, SCPI commands are provided to configure input and output channels, configure the trigger system, and to return any data that algorithms generate in the FIFO and CVT. The generalized programming sequence for the HP E1415A is:

1. Define measurement functions (volts, ohms, temperature etc.) for each input channel
2. Configure the Trigger System (typically using the internal Trigger Timer)
3. Define the algorithms (either built-in PIDs or your custom C algorithms)
4. Initiate the trigger system to start running algorithms (reads inputs, executes all algorithms, and updates the outputs once per trigger).
5. Retrieve data from FIFO or CVT, if necessary.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Simple Examples

The best way to show the power and simplicity of the E1415A is to give you a few examples. This section will explore two simple controllable systems and how we used the E1415A to control them. These examples are intended to show a simple implementation of a control algorithm and are not necessarily the best implementations. Each example shows the setup, the programming, and the function of the control loop.

A Simulated Temperature Bath Example

In this example, we will look at controlling a simulated temperature bath. The example implements the bath using a plastic bottle, a light bulb, a fan, and a T-type Thermocouple. Figure 7 shows a pictorial diagram of the example hardware.

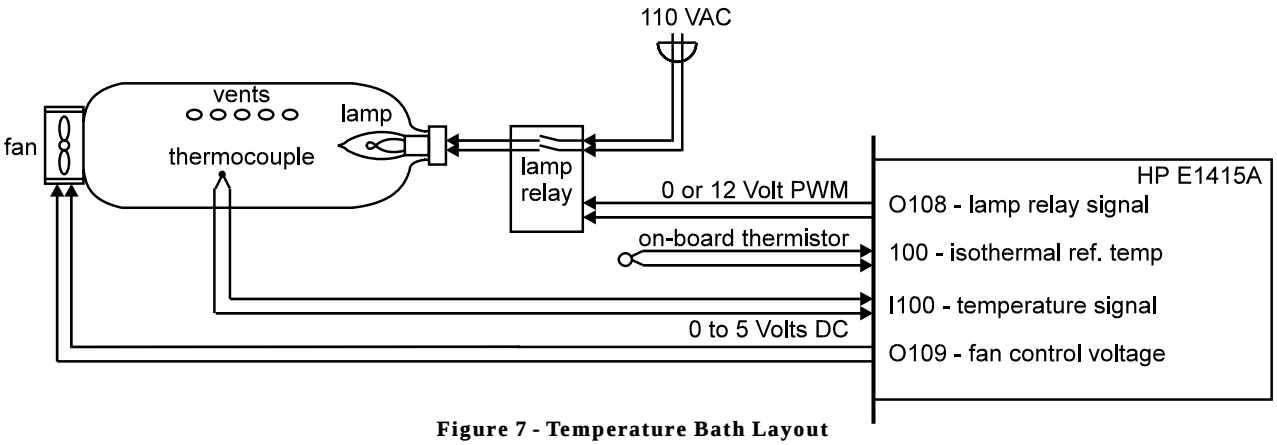


Figure 7 - Temperature Bath Layout

Operation of the Example

In this example, the fan (cooler), and the light (heater) are controlling the temperature of the air inside the bottle. The algorithm makes a temperature measurement from the thermocouple. If the air temperature is less than the heater setpoint, the algorithm increases the duty cycle (on-time) of the lamp relay drive voltage . If the temperature is greater than the cooler setpoint, the algorithm supplies increased drive voltage to the fan motor. If the calculated fan voltage is great enough for the fan to run, the algorithm supplies an initial pulse to the fan to get reliable starting even at low drive levels.

The algorithm implements an internal counter to derive the pulse width modulated signal for the light relay. This counter also sets the relay update rate to 1Hz (.01 sec trigger rate times 100). The fan starting pulse is 10 counts long, or .1 seconds.

The setup time for this experiment was very small. For this example, we started with a “template” example program supplied with the HP E1415A. We then edited the template’s SCPI command sequence (where necessary), wired to the connector module, and were controlling temperature in less than two hours.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following is a sequence of SCPI commands and Algorithm Language with I/O library syntax removed. This is to simplify understanding the example function. The complete example listing can be found on page 14.

```
  /* preset HP E1415A to known state
*RST

  /*  Set Up analog input channel functions  */
SENS:REF THER, 5000, (@101)      /* Ref temp input */
SENS:FUNC:TEMP TC, T, (@100)     /* TC t type input */
SENS:REF:CHAN (@101),(@100)      /* set up 101 as ref for 100 */

  /* Configure Trigger Subsystem and Data Format */
TRIG:SOUR TIMER;:TRIG:TIMER .01          /* This sets the trigger to 10ms */
SAMP:TIMER 10e-6                         /* default for analog input scanning */
FORM REAL,32                             /* sets the data format to real  */

/********************************** begin Algorithm Code **********************************/
    static float time=0,ltime=50;\n                                  /*set up and initialize */
    static float temp=1,fansetpoint=35,lightsetpoint=35;\n           /*algorithm variables  */
    static float fan=0,light,tc;\n
    time = time + 1;\n                                               /* set up algorithm timer */
    if (time > 99) {time = 0;}\n
    if (time == 10) {0109 = fan;}\n     /* set fan voltage after initial kick */
    if (ltime < time) {0108 = 0;} else {0108 = 12;}\n          /* Turn light ON for ltime/100 duty cycle */
    if (time == 99) {                        /* do calculations for control of fan and light */
      fan = 0;\n
      ltime=0;\n
      tc = I100;\n                           /* read thermocouple (returns degrees C */
    /* if colder than lightsetpoint then calculate light-ON percentage and turn ON the light */
      if (tc < lightsetpoint) {ltime = ((lightsetpoint-tc)*25);0108 = 12;}\n

    /* if hotter than the fansetpoint then calculate the fan ON voltage */
      if (tc > fansetpoint) {fan = ((tc-fansetpoint)*3);}\n
      if (fan > 5) {fan = 5.001;}\n                                  /* clip the fan output to 5.001 */
    /* if the fan output is enough to run the fan (>2) then kick the fan ON with full output initially */
      if (fan > 2) {0109 = 5;} else {0109 = 0;}\n

      writecvt(I100,20);writecvt(fan,21);writecvt(ltime,22);\n  /* record data to view from computer */
    }\n;
    /********************************** End Algorithm Code **********************************/

ALG:DEF 'ALG1',algorithm  /* down-load algorithm source string in "algorithm"  to HP E1415A */

INIT   /* Initiate Trigger System - start scanning and running algorithms  */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Balance Beam

The balance beam example was created to show how easy it is to program a solution to a more difficult control problem. Being a mechanical system, it is also intriguing to watch.

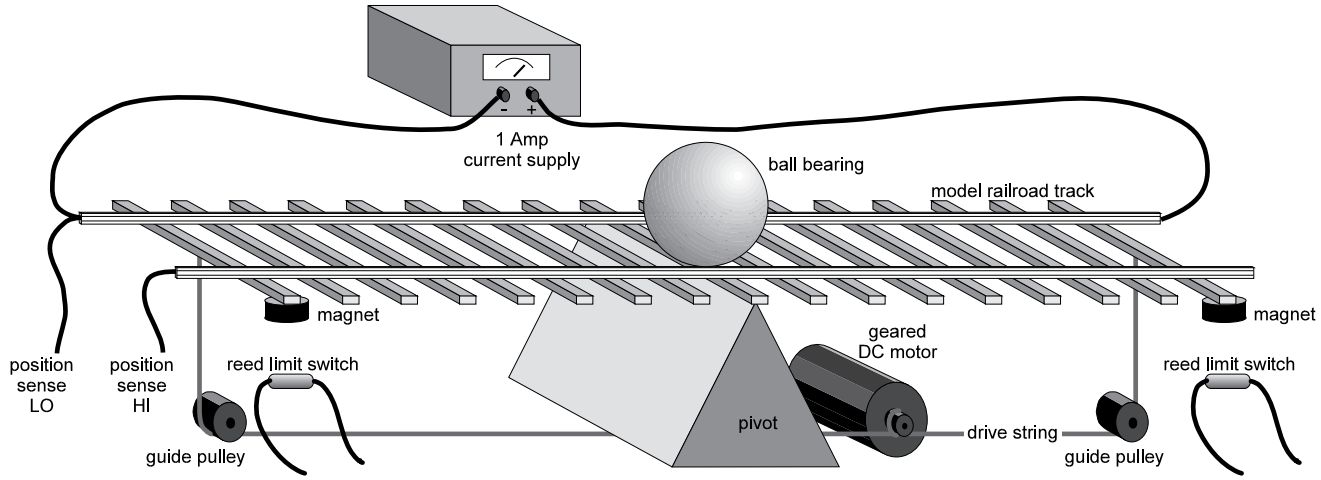


Figure 8 - Balance Beam Mechanical Layout

Operational Overview

Figure 8 shows the mechanical layout of the balance beam while Figure 9 shows the electrical diagram. A photograph can be found on page 13. The intent is to control the position of a ball bearing on a section of model railroad track. The example hardware allows the algorithm to sense the position of the ball, and then drive the motor to tilt the track. What makes this control application difficult is that the only information to the algorithm is the position of the ball. The tilt of the track is not directly measured by the algorithm.

We refer to the algorithm we have developed for this example as a "PDA". A proportional term tries to zero ball position error. A derivative term tries to zero ball velocity. The "A" term tries to zero the ball acceleration (equivalent to leveling the track).

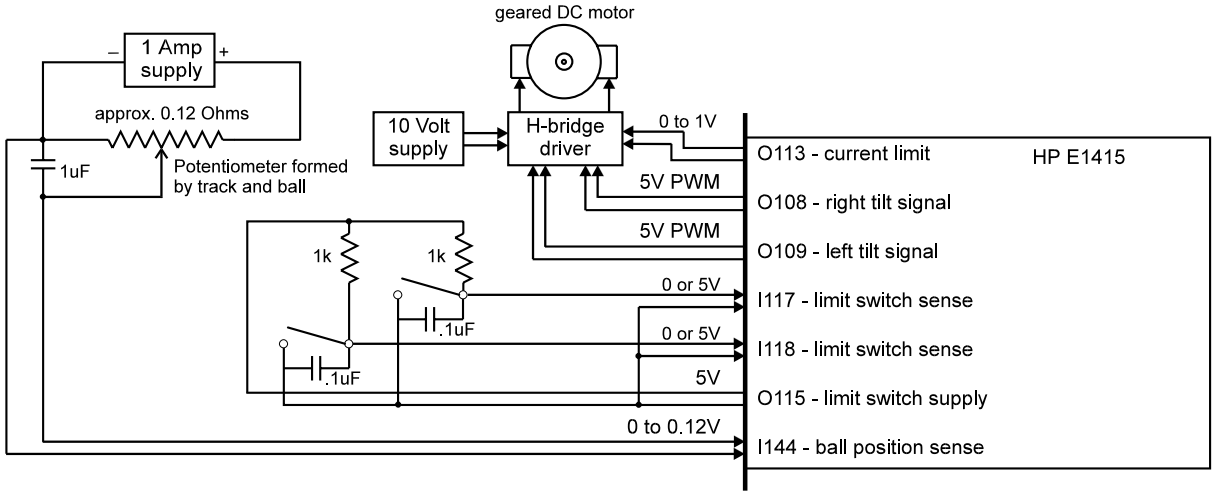


Figure 9 - Balance Beam Schematic

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Detailed Operation

This section will help you understand the example program that follows.

Initialization

Initialize algorithm counter, provide 5 volts from O115 for limit switch pull-ups, and pre-set the current limit FET drive (O113).

Algorithm Counter

The algorithm implements an internal counter to derive the pulse width modulated signal for the motor drive. The counter also allows dividing the complete algorithm execution into several sections. This allows a portion of the algorithm to execute at each trigger event, reducing the required time between triggers.

Read Ball Position

Current flowing in one rail develops about 0.12Volts over its length. The ball bearing and second rail form the wiper of a potentiometer. The sense voltage is zero with the ball at the left end and about 0.12 volts at the right end. Up to 50 readings from input channel I144 are summed to form the average ball position calculated in the next step.

Calculate Terms

When *alg_cntr* reaches 2, the following algorithm terms are evaluated.

- readings averaged to determine ball position: $position = \frac{positionsampls}{samplcnt}$
- the velocity term: *(position - positionold)* x *veloc_factor* (the derivative of the position term)
- the acceleration term: *(veloc_term - veloc_old)* x *accel_factor* (the derivative of the velocity term)
- the position term: *(position - setpoint)* x *position_factor*

Sum the Terms

When *alg_cntr* reaches 4, the output value is calculated from the sum of the individual terms and the current limit FET drive is output on channel O113.

- the output value: output = accel_term + veloc_term + position_term

Extract Direction & Set PWM On-Time

When *alg_cntr* reaches 5, the direction is determined from the sign of the output variable. The output magnitude becomes the count for *on_time. On_time/50* is the duty cycle for the PWM motor signal.

- the direction: *dir = sign(output); output = abs(output)*
- the motor pulse width on-time: *on_time = output*

Output PWM Motor Signal

Every trigger cycle, this section checks the limit swithces (I117 & I118), and the *on_time* variable to decide whether to continue to the drive motor. The direction variable (*dir*) determines which output to drive (O108=left tilt, and O109=right tilt).

Send Variables to CVT

Makes variable values available to the application program for display.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following is a sequence of SCPI commands and Algorithm Language with I/O library syntax removed. This is to simplify understanding the example function. The complete example listing will be found on page 16.

```
*RST;*CLS   /* Start from a known instrument state */

 /*  Setup SCP functions  */
INP:FILT:FREQ 10,(@144)                 /* filter the track voltage */
INP:FILT:STAT ON,(@144)                 /* turn on the filter      */
 /* Configure Trigger Subsystem and Data Format */
TRIG:SOUR TIMER;:TRIG:TIMER .001                /* Set E1415A trig to 1ms   */
SAMP:TIMER 10e-6                        /* default                 */
FORM REAL,32                            /* set data format to real */


/*************************** begin Algorithm Code ******************************/
/* variable declaration and variable initialization */
    static float alg_cntr=0,on_time=25,setpoint=.058,error=0;\n
    static float position=1,positionsampls=0,samplcnt=0;\n
    static float position_term=0,positionold=1,position_factor = 500;\n
    static float veloc_term=0,veloc_old=0,veloc_factor = 21000;\n
    static float accel_term = 0,accel_factor = 22;\n
    static float output=0,outpfloor=19,dir=0,currentlimit=450;\n

/* INITIALIZATION */
    if (alg_cntr == 0) {O112 = 5;O113 = currentlimit/100;O115 = 5;}\n

/* SET ALGORITHM COUNTER */
    alg_cntr = alg_cntr + 1;\n
    if (alg_cntr > 50) {alg_cntr = 1;}\n

/* READ BALL POSITION */
/* sum for average and throw out bad data */
    if ((I144>.001)&&(I144<.130)) {positionsampls = positionsampls + I144;samplcnt = samplcnt + 1;}\n

/* CALCULATE TERMS */
    if (alg_cntr==2) {\n
      /* read track voltage for position information */
      /* then calculate position, velocity, and acceleration terms */
         positionold = position;position =positionsampls/samplcnt;positionsampls=0;samplcnt=0;\n
         veloc_old=veloc_term;veloc_term = (position-positionold)*veloc_factor;\n
         accel_term = (veloc_term-veloc_old)*accel_factor;\n
         position_term = ( position - setpoint)*position_factor;\n
       }\n

     /* CLIP THE TERMS */
   if (alg_cntr==3) {\n
        accel_term = max(-500, min(500, accel_term));\n
        position_term = max(-40, min(40, position_term));\n
        veloc_term = max(-50, min(50, veloc_term));\n
      }\n

    /* SUM THE TERMS */
   if (alg_cntr==4) {\n
        O113 = currentlimit/100;\n   /* O113 controls power FET gate to limit motor current */
        output = accel_term+veloc_term+position_term;\n   /* sum terms for magnatude of control drive */
      }\n
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  /* EXTRACT DIRECTION & SET PWM ON-TIME */
if (alg_cntr==5) {\n
 if (output < 0) {output = abs(output);dir = -1;}\n
       else {dir = 1;}\n
output = max(outpfloor, output);\n   /* use at least minimum output value */
on_time = output;\n
}\n

/* OUTPUT PWM MOTOR SIGNAL */
/* based on direction variable, limit switch readings and on_time variable */
if ((I118 > 2)&&(dir > 0)&&(on_time > alg_cntr)) {O108=5;O109=0;} else \
if ((I117 > 2)&&(dir < 0)&&(on_time > alg_cntr)) {O109=5;O108=0;} \n"
else {O108 = 0;O109=0;}\n

/* SEND VARIABLES TO CVT */
writecvt(output,21);writecvt(dir,22);writecvt(veloc_term,23);writecvt(accel_term,24);writecvt(position,25);\n \n;

/****************************** end Algorithm Code ******************************/
ALG:DEF 'ALG1','algorithm'  /* Download algorithm source string in "algorithm" to HP E1415A */

INIT          /* Initiate Trigger System - start scanning and running algorithm */
```

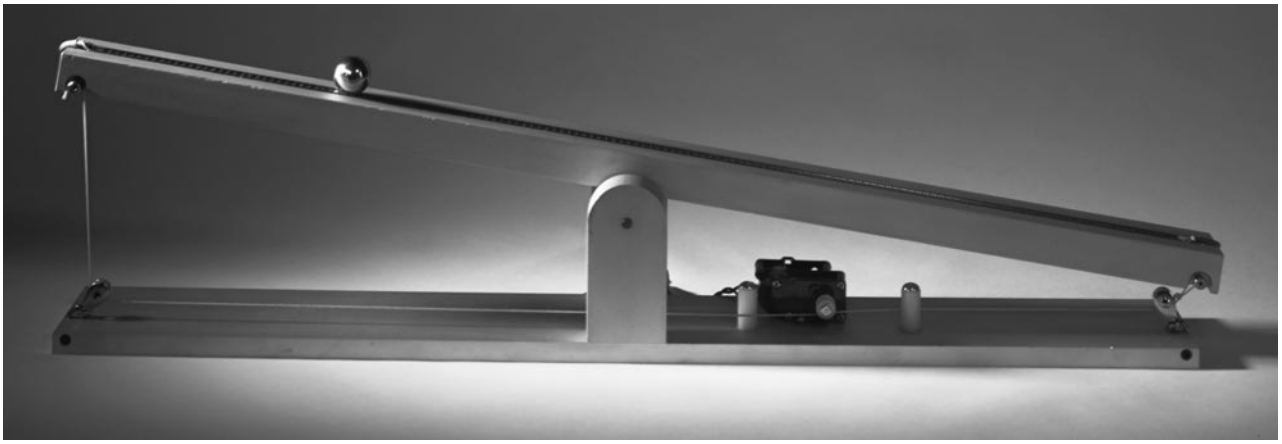


Figure 10

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Complete Listing for Examples

Temperature Bath Example

```
/* C-SCPI Example program for the E1415A Algorithmic Closed Loop Controller
 * file name "cligpid1.cs"
 *
 * This program example shows the use of a custom control algorithm
 * to control a fan and a light relay in order to regulate the temperature
 * inside a bottle.
 */

/* Standard include files */
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <math.h>

/* Instrument control include files */
#include <cscpi.h>      /* C-SCPI include file */

/* Declare constants  */
#define E1415_ADDR    "vxi,24"      /* The C-SCPI address of your E1415A */
INST_DECL(e1415, "E1415A", REGISTER);       /* E1415 */

/* Main program */
void main()
{
   /* Main program local variable declarations */
   char              *algorithm;                    /* Algorithm string */
   int                        alg_num;              /* Algorithm number being loaded */
   char              string[333];                   /* Holds error information */
   int32             error;                         /* Holds error number */

   INST_STARTUP();                                  /* Initialize the C-SCPI routines */

   /* Open the E1415 device session with error checking */
   INST_OPEN(e1415, E1415_ADDR);                    /* Open the E1415A */
   if (! e1415) {                                   /* Did it open? */
            (void) fprintf(stderr, "Failed to open the E1415A at address %s\n",
                        E1415_ADDR);
            (void) fprintf(stderr, "C-SCPI open error was %d\n", cscpi_open_error);
            exit(1);
   }
   /* Check for startup errors */
   INST_QUERY(e1415,"syst:err?\n", "%d,%S", &error, string);
   if (error) {
      (void) printf("syst:err %d,%s\n", error, string);
      exit(1);
   }

   /* Start from a known instrument state */
   INST_CLEAR(e1415);                       /* Selected device clear */
   INST_SEND(e1415, "*RST;*CLS\n");

   /*  Set Up SCP functions  */
   INST_SEND(e1415, "sens:ref ther, 5000, (@101)\n");          /* Ref temp input        */
   INST_SEND(e1415, "sens:func:temp tc, t, (@100)\n");         /* TC t type input       */
   INST_SEND(e1415, "sens:ref:chan (@101),(@100)\n");          /* set up 101 as ref for 100 */

   /* Configure Trigger Subsystem and Data Format */

   INST_SEND(e1415, "trig:sour timer;:trig:timer .01\n");   /* This sets the trigger to 10ms */
   INST_SEND(e1415, "samp:timer 10e-6\n");                  /* default                       */
   INST_SEND(e1415, "form real,32\n");                      /* sets the data format to real */

   /* Download algorithm with in-line code */
      algorithm = " \n"

   "static float time=0,ltime=50;\n"                        /*set up and initialize */
   "static float temp=1,fansetpoint=35,lightsetpoint=35;\n" /*algorithm variables  */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
"static float fan=0,light,tc;\n"

"time = time + 1;\n"                                    /* set up algorithm timer */
"if (time > 99) {time = 0;}\n"

"if (time == 10) {O109 = fan;}\n"                       /* switch to control fan voltage after */
                                                        /* initial kick if 'fan' was greater   */
                                                        /* than 2. See 'if (time == 99)'       */

"if (ltime < time) {O108 = 0;} else {O108 = 12;}\n"     /* Turn light ON for    */
                                                        /* ltime/100 duty cycle */

"if (time == 99) {"                                     /* do calculations for control of fan and light */
  "fan = 0;\n"
  "ltime=0;\n"
  "tc = I100;\n"                                        /* read thermocouple with automatic calculation */
                                                        /* to return temperature in degrees C */

  "if (tc < lightsetpoint) {ltime = ((lightsetpoint-tc)*25);O108 = 12;}\n"
                                                        /* if colder than lightsetpoint then calculate light-ON */
                                                        /* percentage and turn ON the light */

  "if (tc > fansetpoint) {fan = ((tc-fansetpoint)*3);}\n"
                                                        /* if hotter than the fansetpoint then calculate the fan*/
                                                        /* ON voltage*/

  "if (fan > 5) {fan = 5.001;}\n"                       /* clip the fan output to 5.001      */
  "if (fan > 2) {O109 = 5;} else {O109 = 0;}\n"         /* if the fan output is enough to run */
                                                        /* the fan (>2) then kick the fan ON  */
                                                        /* with full output initially        */

   "writecvt(I100,20);writecvt(fan,21);writecvt(ltime,22);\n"  /* record data to view from computer */
 "}\n";

INST_SEND(e1415, "alg:def 'ALG1',%*B\n", strlen(algorithm) + 1, algorithm);

/* Initiate Trigger System - start scanning and running algorithms  */
INST_SEND(e1415,"init\n");

/* Alter run-time variables and Retrieve Data */
while( 1 ) {
        float32 setpoint = 0, process_info[4];
        int i;

        /* type in < 0 to exit */
  printf("Enter desired light setpoint(0:100, neg to quit): ");
  scanf( "%f",&setpoint );
  if ( setpoint < 0.00 ) break;
  INST_SEND(e1415,"alg:scal 'alg1','lightsetpoint',%f\n", setpoint ); /* write new value */
        printf("Enter desired fan setpoint(0:100, neg to quit): ");
        scanf( "%f",&setpoint );
        if ( setpoint < 0.00 ) break;
        INST_SEND(e1415,"alg:scal 'alg1','fansetpoint',%f\n", setpoint );  /* write new value */

        INST_SEND(e1415,"alg:upd\n");                   /* Update the card with the new values */


        for ( i = 0; i < 10 ; i++ )                     /* read CVT 10 times          */
    {                                                   /* ALG1 has elments 20-22 in CVT */
        INST_QUERY( e1415, "data:cvt? (@20:22)","%f",&process_info ); /* read data */
        printf("Temp, Fan voltage (0-5), Light duty (0-100%): %f, %f, %f\n",process_info[0],
        process_info[1],process_info[2]);               /*display algorithm data to screen */
    }
 }
/* Normal end of program */
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Balance Beam Example

```
/* C-SCPI Example program for the E1415A Algorithmic Closed Loop Controller
 * file name "balan1.cs"
 *
 *  This show a custom algorithm to control the balancing of a
 *  ball on a track
 *
 */

/* Standard include files */
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <math.h>

/* Instrument control include files */
#include <cscpi.h>                              /* C-SCPI include file */

/* Declare constants  */
#define E1415_ADDR    "vxi,24"      /* The C-SCPI address of your E1415A */
INST_DECL(e1415, "E1415A", REGISTER);        /* E1415A */

/* Main program */
void main()
{
   /* Main program local variable declarations */@CPROG =     char                    *algorithm; /*
Algorithm string */
   int                  alg_num;    /* Algorithm number being loaded */
   char                 string[333]; /* Holds error information */
   int32       error;                  /* Holds error number */

   INST_STARTUP();     /* Initialize the C-SCPI routines */

   /* Open the E1415A device session with error checking */
   INST_OPEN(e1415, E1415_ADDR);              /* Open the E1415A */
   if (! e1415) {                                                          /* Did it open? */
            (void) fprintf(stderr, "Failed to open the E1415A at address %s\n", E1415_ADDR);
            (void) fprintf(stderr, "C-SCPI open error was %d\n", cscpi_open_error);
            exit(1);
   }
   /* Check for startup errors */
   INST_QUERY(e1415,"syst:err?\n", "%d,%S", &error, string);
   if (error) {
      (void) printf("syst:err %d,%s\n", error, string);
      exit(1);
   }

   /* Start from a known instrument state */
   INST_CLEAR(e1415);             /* Selected device clear */
   INST_SEND(e1415, "*RST;*CLS\n");

   /*  Setup SCP functions  */
   INST_SEND(e1415, "inp:filt:freq 10,(@144)\n");                  /* filter the track voltage */
   INST_SEND(e1415, "inp:filt:stat on,(@144)\n");                  /* turn on the filter     */

   /* Configure Trigger Subsystem and Data Format */

   INST_SEND(e1415, "trig:sour timer;:trig:timer .001\n");         /* Set E1415A trig to 1ms   */
   INST_SEND(e1415, "samp:timer 10e-6\n");                         /* default                  */
   INST_SEND(e1415, "form real,32\n");                             /* set data format to real */

   /* Download algorithm with in-line code */
      algorithm = " \n"
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* set up variable and defaults */
    "static float alg_cntr=0,on_time=25,setpoint=.058,error=0;\n"
    "static float position=1,positionsampls=0,samplcnt=0;\n"
    "static float position_term=0,positionold=1,position_factor = 500;\n"
    "static float veloc_term=0,veloc_old=0,veloc_factor = 21000;\n"
    "static float accel_term = 0,accel_factor = 22;\n"
    "static float output=0,outpfloor=19,dir=0,currentlimit=450;\n"

/* initialization */
    "if (alg_cntr == 0) {O112 = 5;O113 = currentlimit/100;O115 = 5;}\n"

/* set up alg timer */
    "alg_cntr = alg_cntr + 1;\n"
    "if (alg_cntr > 50) {alg_cntr = 1;}\n"

/* read data from track to average and throw out bad data */
    "if ((I144>.001)&&(I144<.130)) {positionsampls = positionsampls + I144;samplcnt = samplcnt + 1;}\n"

/* start calculations; spread out over several time periods to maintain algorithm speed */

    "if (alg_cntr==2) {\n"

      /* read track voltage for position information        */
      /* then calculate proportional, derivative, */
      /* and acceleration terms                           */

        "positionold = position;position = positionsampls/samplcnt;positionsampls=0;samplcnt=0;\n"
        "veloc_old=veloc_term;veloc_term = (position-positionold)*veloc_factor;\n"
        "accel_term = (veloc_term-veloc_old)*accel_factor;\n"
        "position_term = (position - setpoint)*position_factor;\n"
      "}\n"

      /* Clip accel, position, veloc terms */

    "if (alg_cntr==3) {\n"
        "accel_term = max(-500, min(500, accel_term));\n"
        "position_term = max(-40, min(40, position_term));\n"
        "veloc_term = max(-50, min(50, veloc_term));\n"
      "}\n"

      /* Set current control fet (O113) and calculate output */

    "if (alg_cntr==4) {\n"
        "O113 = currentlimit/100;\n"
        "output = accel_term+veloc_term+position_term;\n"
      "}\n"

      /* calculate output direction and magnitude then set duty-cycle term (on_time)*/
    "if (alg_cntr==5) {\n"
        "if (output < 0) {output = abs(output);dir = -1;}\n"
            "else {dir = 1;}\n"
        "output = max(outpfloor, output);\n"  /* use at least minimum output value */
        "on_time = output;\n"@CPROG =          "}\n"

    /* turn on/off outputs and write information to the CVT */
    "if ((I118 > 2)&&(dir > 0)&&(on_time > alg_cntr)) {O108=5;O109=0;} else \n"
     "if ((I117 > 2)&&(dir < 0)&&(on_time > alg_cntr)) {O109=5;O108=0;} \n"
     "else {O108 = 0;O109=0;}\n"

"writecvt(output,21);writecvt(dir,22);writecvt(veloc_term,23);writecvt(accel_term,24);writecvt(position,25);\n"
    "\n";

  INST_SEND(e1415, "alg:def 'ALG1',%*B\n", strlen(algorithm) + 1, algorithm);   /* Define Algorithm */

       INST_QUERY(e1415,"syst:err?\n", "%d,%S", &error, string);
       if (error) {
          (void) printf("While loading alg1, syst:err %d,%s\n",error, string);
          exit(1);}
   /* Initiate Trigger System - start scanning and running algorithm  */
  INST_SEND(e1415,"init\n");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        INST_QUERY(e1415,"syst:err?\n", "%d,%S", &error, string);
        if (error) {
           (void) printf("While loading alg1, syst:err %d,%s\n",error, string);
           exit(1);}


   /* Alter run-time variables and retrieve data */
   while( 1 ) {
              float32 setpoint = 0, process_info[4];
              int i;

   /* type in < 0 to exit */
      printf("Enter desired motor current limit(450 - default, neg to quit): ");
      scanf( "%f",&setpoint );
      if ( setpoint < 0.00 ) break;
      INST_SEND(e1415,"alg:scal 'alg1','currentlimit',%f\n", setpoint );

      printf("Enter desired ball position (the setpoint)(0.06 - default, neg to quit): ");
      scanf( "%f",&setpoint );
      if ( setpoint < 0.00 ) break;
      INST_SEND(e1415,"alg:scal 'alg1','setpoint',%f\n", setpoint );

      printf("Enter desired output drive floor(19 - default): ");
      scanf( "%f",&setpoint );
      INST_SEND(e1415,"alg:scal 'alg1','outpfloor',%f\n", setpoint );

      printf("Enter desired position factor(500 - default, neg to quit): ");
      scanf( "%f",&setpoint );
      if ( setpoint < 0.00 ) break;
      INST_SEND(e1415,"alg:scal 'alg1','position_factor',%f\n", setpoint );

      printf("Enter desired velocity factor(21000 - default): ");
      scanf( "%f",&setpoint );
      INST_SEND(e1415,"alg:scal 'alg1','veloc_factor',%f\n", setpoint );

      printf("Enter desired acceleration factor(22 - default): ");
      scanf( "%f",&setpoint );
      INST_SEND(e1415,"alg:scal 'alg1','accel_factor',%f\n", setpoint );

      INST_SEND(e1415,"alg:upd\n");                    /* cause new values to take effect */

      for ( i = 0; i < 1000 ; i++ )
      {                                                /* read CVT 1000 times */

                  /* ALG1 has elments 21-25 in CVT */
                  INST_QUERY( e1415, "data:cvt? (@21:25)","%f",&process_info );
                  printf("out,dir,dif,acc,position: %f, %f, %f, %f, %f\n",process_info[0],

process_info[1],process_info[2],process_info[3],process_info[4]);
              }
   }
   /* Normal end of program */
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Built-in PID Algorithms

The following information shows the function block diagrams for the built-in PIDs “PIDA” and “PIDB”.

PIDA Operation

The PIDA algorithm was written to provide maximum loop update rate. Figure 11 shows the functional block diagram of PIDA.

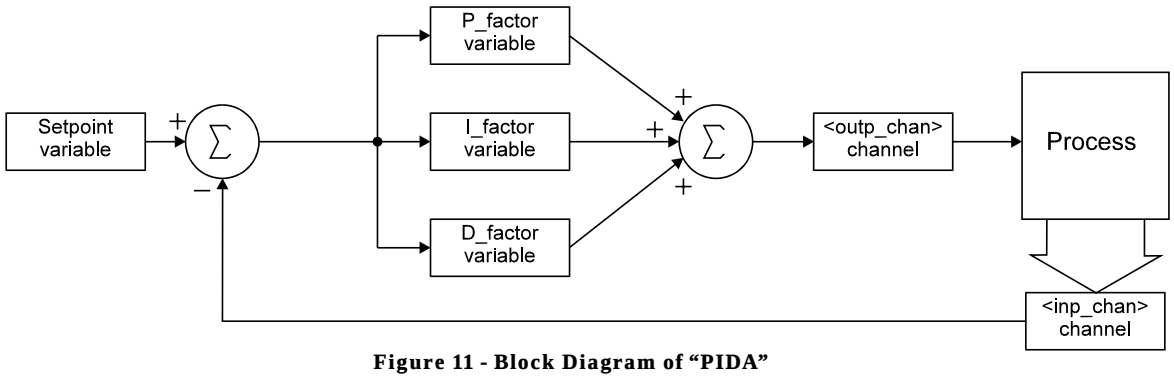


Figure 11 - Block Diagram of “PIDA”

PIDA Listing

The following is the Algorithm Language listing for PIDA.

```
/*****************************************************************************/
/*  I/O Channels                                                             */
/*  Must be defined by the user                                              */
/*                                                                           */
/* inchan - Input channel name                                               */
/* outchan - Output channel name                                             */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/*****************************************************************************/
/* PID algorithm for E1415A controller module.  This algorithm is called    */
/* once per scan trigger by main().  It performs Proportional, Integral      */
/* and Derivative control.                                                   */
/*                                                                           */
/*                                                                           */
/* The output is derived from the following equations:                       */
/*                                                                           */
/* PID_out = P_out + I_out + D_out                                           */
/* P_out = Error * P_factor                                                  */
/* I_out = I_out + (Error * I_factor)                                        */
/* D_out = ((Error - Error_old) * D_factor)                                  */
/* Error = Setpoint - PV                                                     */
/*                                                                           */
/* where:                                                                    */
/*   Setpoint is the desired value of the process variable (user supplied)   */
/*   PV is the process variable measured on the input channel                */
/*   PID_out is the algorithm result sent to the output channel              */
/*   P_factor, I_factor, and D_factor are the PID constants                  */
/*  (user supplied)                                                          */
/*                                                                           */
/*                                                                           */
/* At startup, the output will abruptly change to P_factor * Error.          */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/* User determined control parameters                                    */
   static float Setpoint = 0; /* The setpoint                            */
   static float P_factor = 1; /* Proportional control constant           */
   static float I_factor = 0; /* Integral control constant               */
   static float D_factor = 0; /* Derivative control constant             */
/*                                                                       */
/*  Other Variables                                                      */
   static float I_out; /* Integral term
*/
   static float Error;
/* Error term                                                            */
   static float Error_old;
/* Last Error - for derivative                                           */
/*                                                                       */
/*PID algorithm code:*/
/* Begin PID calculations */
/* First, find the Process Variable "error" */
/* This calculation has gain of minus one (-1) */
   Error = Setpoint - inchan;
/* On the first trigger after INIT, initialize the I and D terms */
   if (First_loop)
   {
/* Zero the I term and start integrating */
      I_out = Error * I_factor;
/* Zero the derivative term */
            Error_old = Error;
   }
/* On subsequent triggers, continue integrating */
   else /* not First trigger */
   {
            I_out = Error * I_factor + I_out;
   }
/* Sum PID terms */
   outchan = Error * P_factor + I_out + D_factor * (Error - Error_old);
/* Save values for next pass */
            Error_old = Error;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PIDB Operation

The PIDB algorithm includes clip limits and alarm limits on several parameters, bump-less manual-to-auto control switching, and parameter values written to the CVT and optionally to the FIFO. Figure 12 shows the functional block diagram of PIDB.

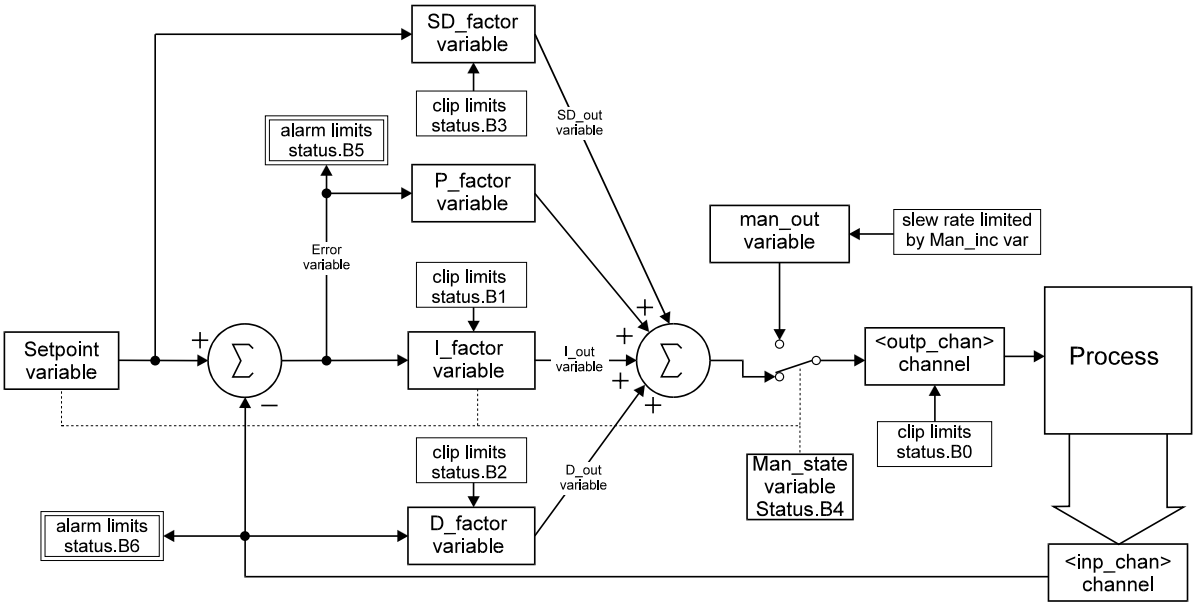


Figure 12 - Block Diagram of "PIDB"

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For more information on Hewlett-Packard Test & Measurement products, applications or services please call your local Hewlett-Packard sales offices. A current listing is available via Web through AccessHP at http://www.hp.com.
If you do not have access to the internet please contact one of the HP centers listed below and they will direct you to your nearest HP representative.

United States:
Hewlett-Packard Company
Test and Measurement Organization
5301 Stevens Creek Blvd.
Bldg. 51L-SC
Santa Clara, CA 95052-8059
1 800 452 4844

Canada:
Hewlett-Packard Canada Ltd.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
(905) 206 4725

Europe:
Hewlett-Packard
European Marketing Centre
P.O. Box 999
1180 AZ Amstelveen
The Netherlands

Japan:
Hewlett-Packard Japan Ltd.
Measurement Assistance Center
9-1, Takakura-Cho, Hachioji-Shi,
Tokyo 192, Japan
Tel: (81-426) 48-0722
Fax: (81-426) 48-1073

Latin America:
Hewlett-Packard
Latin American Region Headquarters
5200 Blue Lagoon Drive
9th Floor
Miami, Florida 33126
U.S.A.
(305) 267 4245/4220

Australia/New Zealand:
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
Australia
1 800 629 485

Asia Pacific:
Hewlett-Packard Asia Pacific Ltd
17-21/F Shell Tower, Times Square,
1 Matheson Street, Causeway Bay,
Hong Kong
Fax: (852) 2506 9285

All registered and unregistered trademarks mentioned are the sole property of their respective companies.

The information contained in this document is subject to change without notice.

©1996 Hewlett-Packard Company
Printed in the USA 9/96
5965-3311E

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP