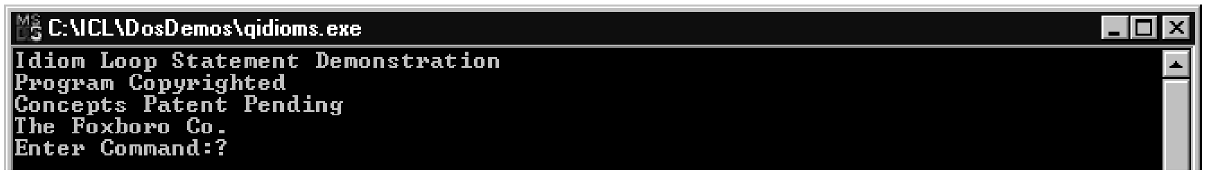
Simple Single Loop Idiom and Simulation Demo

Edgar H. Bristol — Fellow Emeritus — 4/1/01
Control Concepts Originated — The Foxboro Co.
28 Union St
Foxboro MA, 02035
USA
Tel. (508) 543-8829
email: ebristol@mediaone.net

Contents

INTRODUCTION

This is a quick guide/walk through to a very limited (and probably buggy; it was written in three evenings and then evolved) demo program for the Idiom Loop Statement. It includes a basic set of Idioms, which can be strung in a single statement up to the point that the implementing GBASIC program runs out of declared array memory. It includes a simple simulation statement capability just adequate to simulate the intended kind of feed forward, cascaded, constraint control demo. The program is run by any DOS/WINDOWS activation of the qidioms.exe program, which will cause the opening display:



```
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:?
```

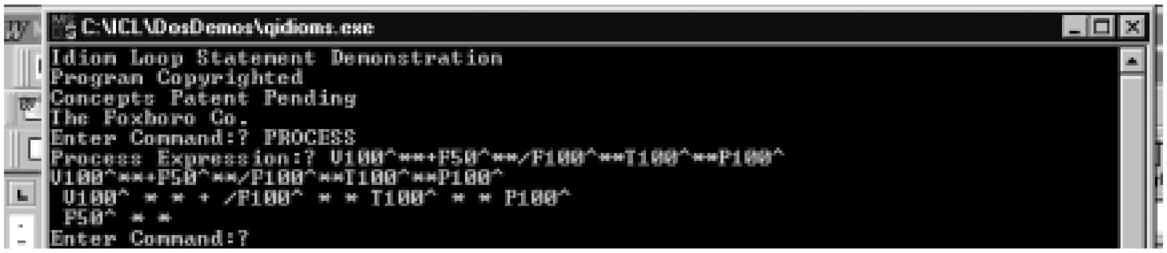
The above message not withstanding, use the program any way you want; the half dozen or so patents have already issued, but are not being used. The patents apply to real control applications not demos. All of the demo commands expect all Cap letters, so it would be good to select the Caps Lock key.

The normal progression though a simulation would call for entering a process model simulation, and then entering a succession of different Loop Statements, and Tuning, Listing and Simulation commands to demonstrate the behavior of the Loops. As an initial reminder, which will be elaborated later, entry of a process model must be preceded by the PROCESS command, and entry of a Loop Statement must be preceded by an IDIOMS command, each command carriage return terminated. An easy error results when entering the statement without the preceding command; in this case nothing happens. The process modeling and Idiom Statement syntax checks are limited to recognition of the point in the statement where a syntax error has been made. The statement then has to be recopied from the beginning. Sorry not to be more helpful. Of course a full Idiom tool or language would be more user friendly (and take longer than three evenings at a control conference to implement).

Commands

PROCESS Command

This (carriage return terminated) command introduces a single-valve, process model, modeling statement, as illustrated and elaborated later. The particular statement shown (in the line initially printing "Process Expression:?") declares a valve variable V100, a disturbance variable F50, and three measured



```
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:? PROCESS
Process Expression:? V100^**+F50^**/F100^**T100^**P100^
V100^**+F50^**/F100^**T100^**P100^
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:?
```

variables F100, T100, and P100. It will plot each variable. More specifically, the V100, with lag dynamics, is added to the disturbance F50, with lag dynamics, to generate a combined flow F100, which, with dynamics, affects the temperature T100, which, with dynamics, affects the pressure P100. No claim is made for the attractiveness of this statement form. Only Tags of four (or less) characters are supported.

The text feedback to this command has two parts: The statement is echoed on the next line, and then an attempted visualization of the parallel signal paths is shown. The statement syntax consists of a progression of phrases each starting with:

- a Tag,
- an optional '^',
- one or more optional '*'s,
- a '+', if a following simulated quantity is to be added to the result of the current phrase, and
- a '/' if several simulated quantities are to be combined to generate the following Tag value.

Each phrase (except those terminated by a '+') generates the result reflected in the initial Tag in the following phrase. In more detail, the individual expressions making up each phrase have the following meanings:

Tags. These represent the different simulated process variables. Each phrase applies one simulated first order lag to its initial Tag variable.

*****. Each asterisk following a Tag represents one additional first order lag applied sequentially to the simulated process variable. Since the overall simulation purpose is to demonstrate the Idioms, no attempt has been made to provide a user set-able time constant for the individual lags.

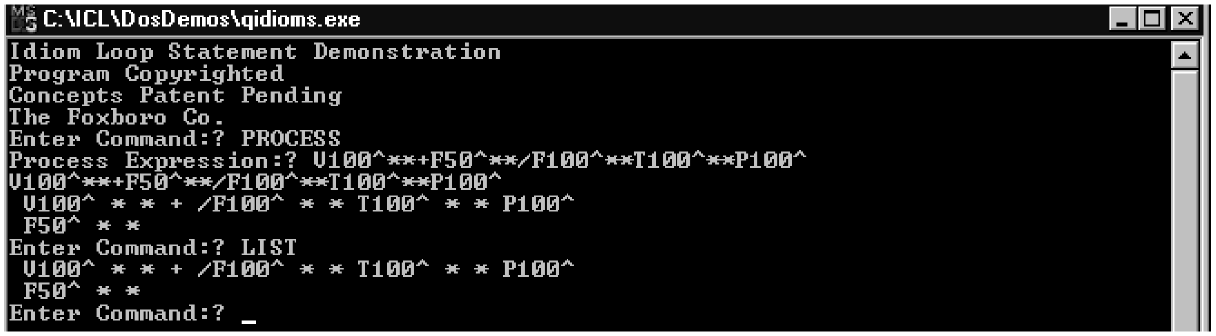
^. The caret causes the preceding Tag to be plotted on the screen as part of the response to the SIMULATE (or RESIM) command.

+. The Plus causes the result of the preceding phrase simulation to be added to the result of the following phrase simulation, to create some later combined simulated variable.

/. The slash indicates that the following Tag represents a variable which is the combined result of preceding Tagged variables.

LIST (See **LISTGR** below) **Command**

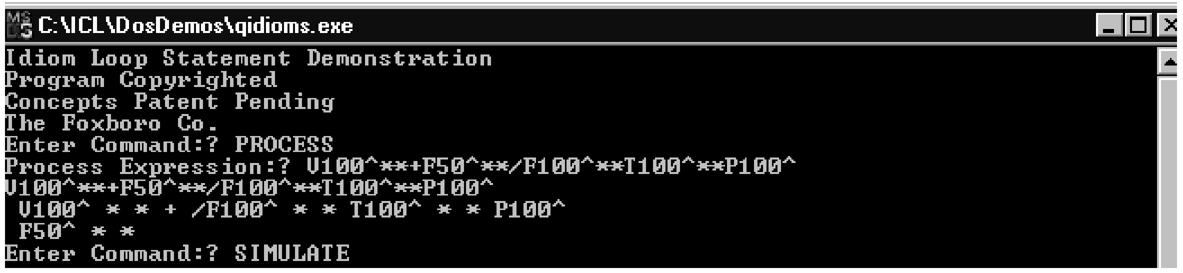
This command causes the most recently entered simulation modeling and/or Idiom statement to be



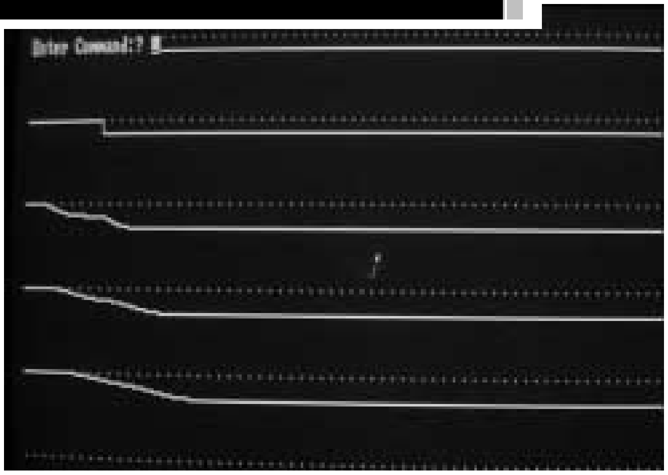
```
C:\ICL\DosDemos\qidioms.exe
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:? PROCESS
Process Expression:? V100^**+F50^**/F100^**T100^**P100^
V100^**+F50^**/F100^**T100^**P100^
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:? LIST
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:? _
```

listed in text form.

Simulation Commands



```
C:\ICL\DosDemos\qidioms.exe
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:? PROCESS
Process Expression:? V100^**+F50^**/F100^**T100^**P100^
V100^**+F50^**/F100^**T100^**P100^
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:? SIMULATE
```



```
Enter Command:? █
```

SIMULATE Command

SIMULATE causes the existing structure to be simulated, in three different modes:

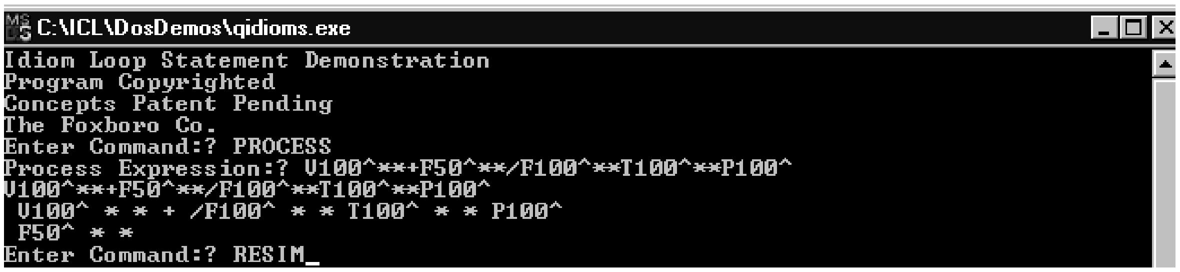
- When the process is first simulated and no controls have been defined (with the Idiom Loop Statement), the simulation proceeds with steps automatically applied to each of the independent valve or disturbance variables in succession. The figure shows the result for the above model:

In order of definition in the simulated model above, from top to bottom, the trended variables are: V100, F50, F100, T100, P100. The first step disturbance is applied to V100, the second to F50. The combined disturbances, with progressively greater dynamics, generate F100, T100, and P100.

- After the first IDIOMS Command, the following SIMULATE command causes the closed loop simulation with the valves controlled, and disturbances applied successively as above, under the setpoint defined for the controlled variable (The default setpoint is .5 (plotted half scale).). Note that cascaded setpoints (except the primary one) will be determined by the controller driving them, whatever their initial state.
- When a later SIMULATE or IDIOMS command is entered the resulting simulation picks up at the end of the previous time without restarting the simulation. Any changes in control structure are picked up bumplessly from the state previously generated by any previous control structure. The purpose of this is to allow the demonstration of bumpless dynamic changes in control structure, within the larger Idiomatic Control Language[1-3]. This facility allows sequenced and conditional execution of Idiom Loop Statements in appropriately extended syntax.

RESIM Command

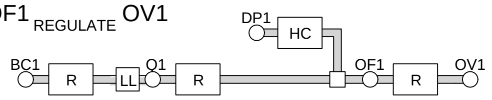
RESIM applies only if an Idiom Loop Statement is defined. It is used to reinitialize the Simulation activity and has the affect of running the simulation from the initial setpoint and disturbance state as in the first Simulation command above (illustrated below, but without the proper preceding Idiom statement).



```
C:\ICL\DosDemos\qidioms.exe
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:? PROCESS
Process Expression:? V100^**+F50^**/F100^**T100^**P100^
V100^**+F50^**/F100^**T100^**P100^
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:? RESIM_
```

Idiom Loop Statement

$BC1_{REGULATE\ LOLIMIT}\ Q1_{REGULATE}\ DP1_{HICONSTR}\ OF1_{REGULATE}\ OV1$

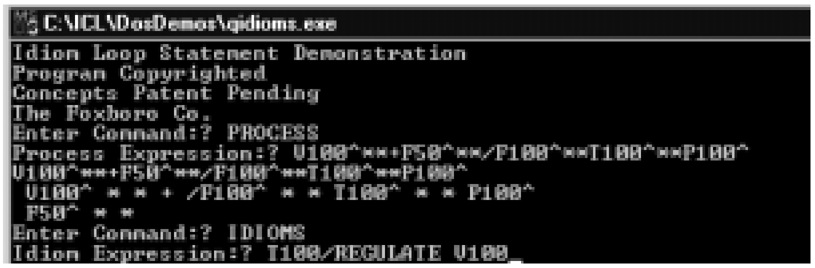


The Idiom Loop Statement is defined elsewhere, particularly in the "Idiom Based Documentation of Continuous (Feedback) Process Control" document.

This statement includes a sequence of Tags each sequence broadly representing one cascaded, single degree of freedom, path. After each Tag in the sequence is a list of subscripted Idioms, with an initial Primary Idiom defining a basic Regulate or Constraint Idiom Intent, and following secondary feedforward or limiting Idioms.

IDIOMS Command

Each demonstrated Idiom Loop Statement is initiated by a carriage returned IDIOMS command. The statement is then entered with Tags entered directly, and Idiom names entered preceded by a '/', as below.



```
C:\ICL\DosDemos\qidioms.exe
Idiom Loop Statement Demonstration
Program Copyrighted
Concepts Patent Pending
The Foxboro Co.
Enter Command:? PROCESS
Process Expression:? V100^**+F50^**/F100^**T100^**P100^
V100^**+F50^**/F100^**T100^**P100^
 V100^ * * + /F100^ * * T100^ * * P100^
 F50^ * *
Enter Command:? IDIOMS
Idiom Expression:? T100/REGULATE V100_
```

Syntax errors are recognized either in terms of Tags not matching the process simulation Tags, or Idioms in error. The error feedback takes the form of a restatement of the input text up to the point of the error, which must be then corrected by reentry of the statement. Completion of the statement with carriage return causes a graphic and text listing of the resulting statements as well as the entry for following simulations.

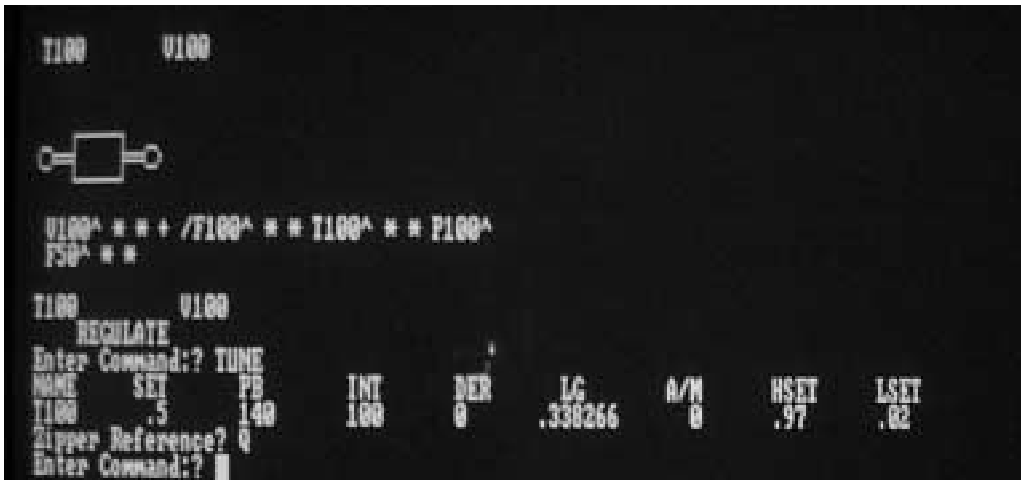
Idioms

The following Idioms are supported/demonstrated.

- **REGULATE** (also abbreviated **R**). This represents the normal PID regulation action (the Derivative action is inactive in the simulation).
- **HICONSTR** (also abbreviated **HC**). This represents the PID (the Derivative action is inactive in the simulation) acting as a high constraint override through a Selector.
- **LOCONSTR** (also abbreviated **LC**). This represents the PID (the Derivative action is inactive in the simulation) acting as a low constraint override through a Selector.
- **FEEDFWD** (also abbreviated **FF**). This represents the normal properly compensated feed forward function, applicable as a secondary support to a REGULATE Idiom. The /FF is followed by the Tag for the measured disturbance variable (as in /FF F50 in the example).
- **HILIMIT** (also abbreviated **HL**). This represents the normal application of a high limit on the output of a regulation Idiom.
- **LOLIMIT** (also abbreviated **LL**).). This represents the normal application of a low limit on the output of a regulation Idiom.

LISTGR Command

This command causes the most recently entered Idiom to be listed with its graphic form (and Process).



```
T100      V100



V100^ * * + /T100^ * * T100^ * * P100^
F50^ * *

T100       V100
   REGULATE
Enter Command:? TUNE
NAME    SET     PB      INT     DER     LG        A/M    HSET    LSET
T100     .5     140     100     0       .338266    0      .97     .02
Zipper Reference? Q
Enter Command:?
```

TUNE Command

This command (see above) lists the parameters for each Idiom Loop Statement Tag, allowing their change. The “Zipper Reference?” (from the language SuperVariable concept[2, 3]) line calls for the entry of a parameter reference (e.g. T100.SET) to allow a parameter change (later figure) or Q (for Quit). Note that the Idiom (controller) setpoint (T100.SET) is defaulted to .5 (half scale) as indicated earlier. The listed parameters are:

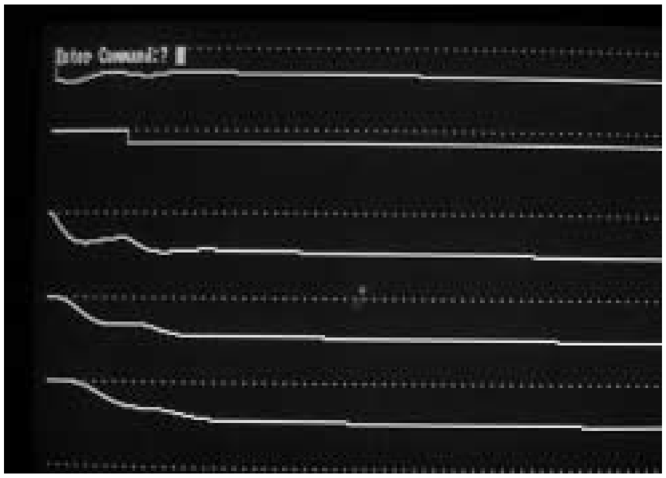
• (**NAME**,	The Tag for the variable associated with the Idioms.)
• **SET**,	The Setpoint associated with the variable and Idiom.
• **PB**,	The Proportional Band associated with the Idiom.
• **INT**,	The Integral Time associated with the Idiom.
• **DER**,	The Derivative Time associated with the Idiom.
• **LG**,	The internal (lag) Integral calculated value associated with the Idiom.
• **A/M**,	The Auto/Manual state associated with the Idiom.
• **HSET**,	The Setpoint for any associated HiLimit Idiom.
• **LSET**,	The Setpoint for any associated LoLimit Idiom.

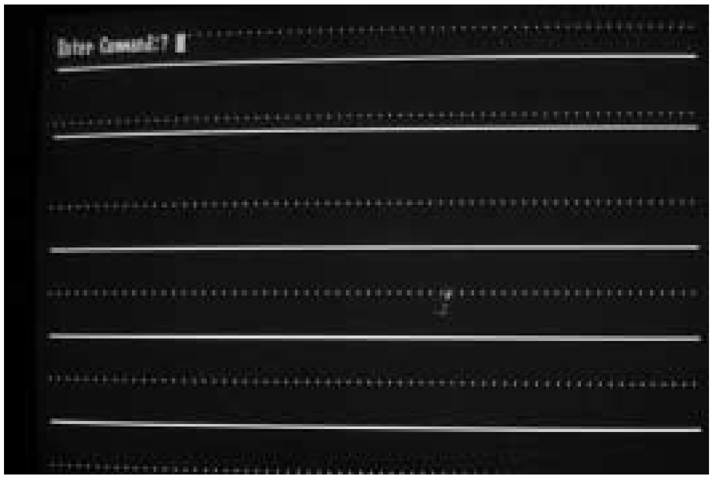
Z Command

This Command allows the entry of changes in parameter more directly, similar to TUNE but without the display of the table of parameters for the Loop.

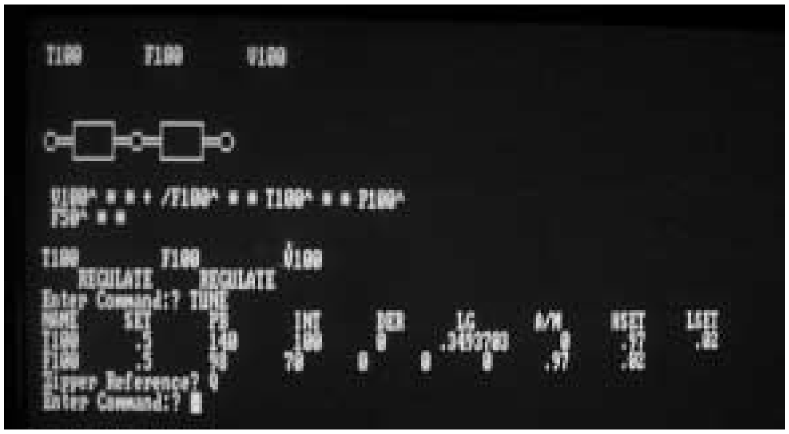
DEMONSTRATION WALKTHROUGH

The figures above constitute the start of a demo run through: the process model specification and simulation, the single Regulate Idiom Statement, the Tuning, etc. Continuing in this vein we can Simulate the result of the initial single Idiom Idiom Loop Statement:





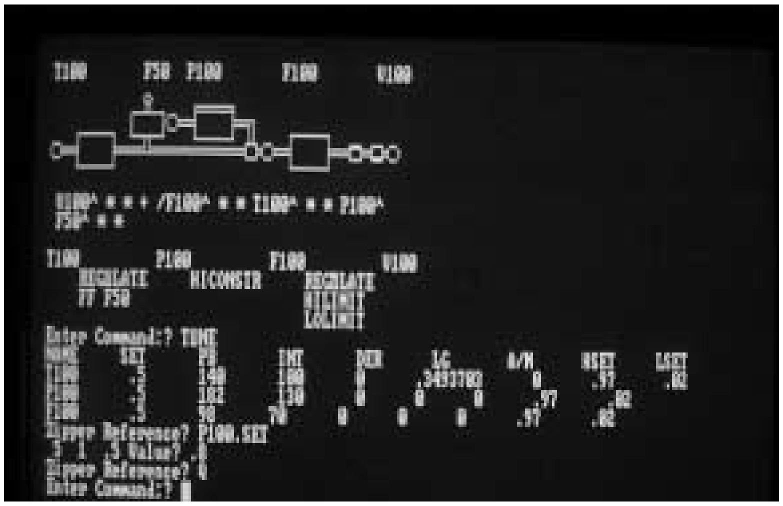
As before the variables are plotted in defined order: V100, F50, F100, T100, P100. The initial V100 is determined by the controller output. After the setpoint change and load disturbance, the T100 settles to the .5 (half scale) setpoint value. A repeated SIMULATE command causes the continuation from the point in time left off. No further changes follow.



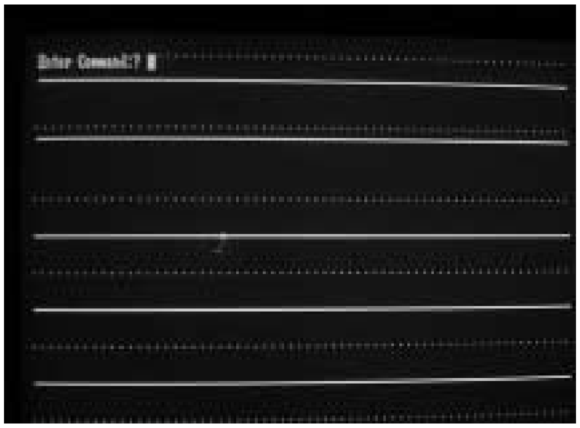
```
T100        F100        V100

V100^ * * + /F100^ * * T100^ * * P100^
F50^ * *
T100          F100          V100
    REGULATE      REGULATE
Enter Command:? TUNE
NAME    SET    PB     INT    DER    LG         A/M    HSET   LSET
T100    .5     140    100    0      .3497983   0      .97    .02
F100    .5     70     70     0      0          0      .97    .02
Zipper Reference? 0
Enter Command:? █
```

A change in control structure, to a cascaded loop can now be entered (to the right), with an IDIOMS command and the entry of T100/REGULATE F100/REGULATE V100 or T100/R F100/R V100). When completed, the listed result is as shown. The TUNE command shows the defaulted setpoints; the F100 setpoint will be taken over by the primary Idiom controller. A SIMULATE command would show that the changed structure is imposed with no process bump, showing the flat continuation of before.

A more complicated structure can now be entered (e.g. T100/R/FF F50 P100/HC F100/R/HL/LL V100):



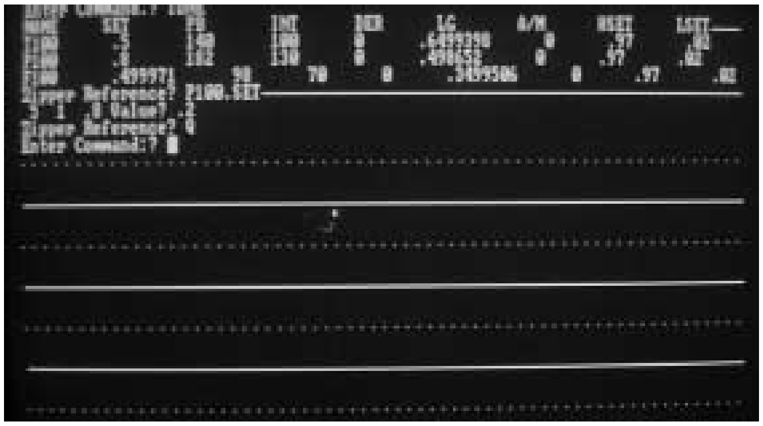
```
T100      F50 P100       F100      V100

V100^ * * + /F100^ * * T100^ * * P100^
F50^ * *

T100         P100           F100        V100
   REGULATE       HICONSTR     REGULATE
   FF F50                      HILIMIT
                               LOLIMIT
Enter Command:? TUNE
NAME     SET       PB     INT     DER      LG         A/M      HSET     LSET
T100      .3       140    100      0     .3493780      0        .97      .02
P100      .3       102    130      0       0      0         .97      .02
F100      .3       98   70     0       0       0      .97      .02
Zipper Reference? P100.SET
 3  1   .3 Value? .8
Zipper Reference? 0
Enter Command:? █
```



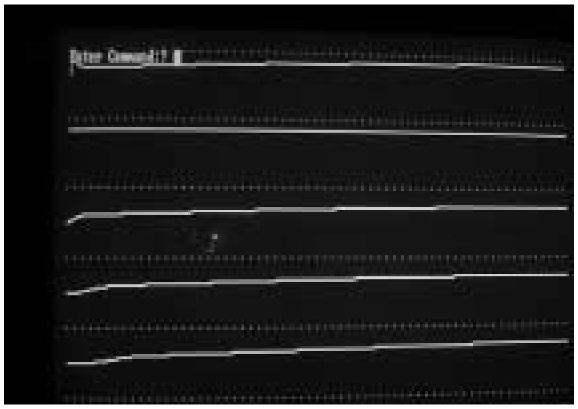
```
Enter Command:? █
```

This includes the cascading of the previous structure, a feed forward, a constraint control, and high and low limits on the output. In the TUNE command above the constraint setpoint (P100.SET) has been set to .8, outside of the range of any effect. A SIMULATE command (above) shows that the change was again made bumplessly. A RESIM command would show the re-simulation from the beginning with disturbances and the anticipative corrective action of the feed forward

The TUNE command can now be used to change the constraint setpoint to a value (from .8 to .2), which overrides:



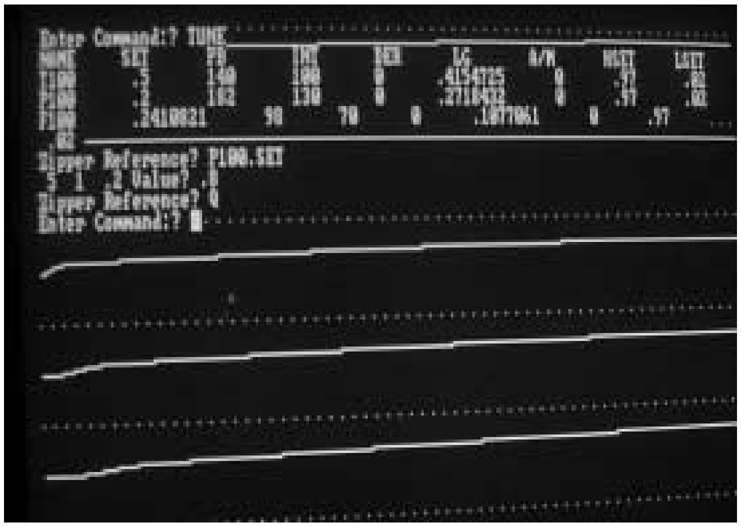
```
Enter Command:? TUNE
NAME     SET       PB     INT     DER      LG         A/M      HSET     LSET
T100      .3       140    100      0     .6489390      0        .97      .02
P100      .3       102    130      0     .498652       0        .97      .02
F100     .499971      98     70      0      .3499506      0      .97      .02
Zipper Reference? P100.SET
 3  1   .8 Value? .2
Zipper Reference? 0
Enter Command:? █
```



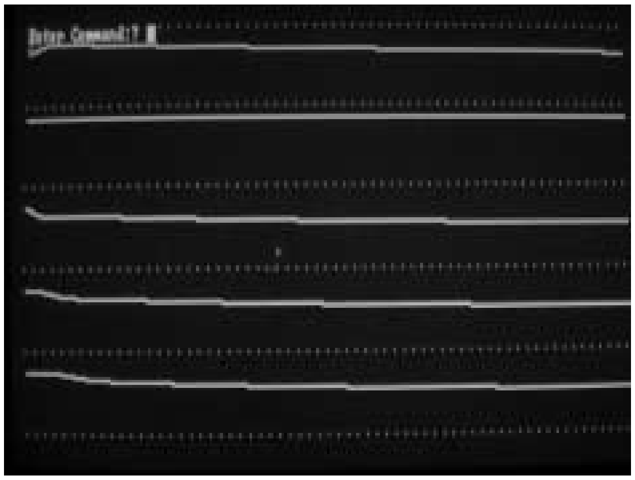
```
Enter Command:? █
```

The SIMULATE command now shows the take over of the Constraint Idiom Controller, driving the P100 to the intended constraint setpoint (.2, also driving the simulated T100 to that same value, as necessary to accomplish the constraint control):

A return of the constraint setpoint to the original value restores the original control of T100:



```
Enter Command:? TUNE
NAME     SET       PB    INT   DER   LG         A/M   HILT   LOLT
T100     .5        140   100   0     .4154725   0     .97    .02
F100     .2        102   130   0     .2710432   0     .97    .02
P100     .2410821      90    70    0     .1077061   0     .97
 .02
Zipper Reference? P100.SET
 5  1  .2 Value? .8
Zipper Reference? Q
Enter Command:? 
```



```
Enter Command:? 
```

HOW TO STOP THE PROGRAM

The only way to stop the program is to exit the DOS shell by typing Control-C. The shell will then display ***Break*** followed by **Hit any key to return to system**. Then if you hit any key the shell will disappear.

IMPLEMENTATION

The demo is implemented in compiled GBASIC, circa 1986, following the p-code implementation strategy defined for the field device version of the Idiomatic Control Language[3] and the more general specification outlined in the associated documentation tool document[1]. Unlike the conventional control block implementations (or Object implementations modeled after them), the p-code implementation is a natural reflection of an operator statement representation, as described here. It supports compact integration of continuous control with other sequencing and conditional programming execution, and the bumpless transitioning essential to such an integration. The p-coded implementation is less obviously related to the current Object programming interest than control blocks, but, in fact, each p-code is an Object, whatever the implementation language, with methods, which uniformly support the different control compilation, documentation, and operational requirements. More important, the p-code implementation is much more compatible to a higher level expression and management of Idioms as Patterns (in the Object Oriented meaning of that word).

Bibliography

[1] E.H. Bristol, “Idiom Based Documentation of Continuous (Feedback) Process Control”, privately distributed, email available, user guide.

[2] E.H. Bristol, “SuperVariable Process Data Definition”, ISA SP50.4 Working Paper, Oct. 24, ’90.

[3] E.H. Bristol, “A Field Device Language in a Process Control Language Family”, ISA2000, New Orleans, Oct. 5-9, ’00.