42 Aero Camino, Goleta, CA 93117
Tel (805) 685-0066 | Fax (805) 685-0067
info@biopac.com | **www.biopac.com**

10.26.2020

Application Note 291: Pulse Amplitude Feedback Control Using BIOPAC Basic Scripting

This application note presents a comprehensive description of how to use the BIOPAC Basic Scripting (BBS), Acq*Knowledge*, and MP hardware to implement simple control loops. Sections include Background, Setup, Algorithm, Results, and Appendices A (channel setting for graph template) and B (code).

1.0 Background

This exercise is intended to show that one can implement feedback control to external devices using MP160/AMI100D and BIOPAC Basic Scripting (BBS). This technique is useful for applications that need/require stability in temperature, flow rate, and pressure.

2.0 Setup

Figure 1 shows the schematic for feedback control. The Flow Measurement signal is processed through CH 8 of the AMI100D then filtered via CH 40 (LPF). The signal from CH 40 is sampled in BBS during OnIdleAcquisition and adjusted as necessary to maintain the desired flowrate. The adjusted signal is routed through OUT0 then to the Flow Source control port (Figure 2a). A bleed valves are used to impact flow and allow the control loop to adjust for it (Figure 2b).

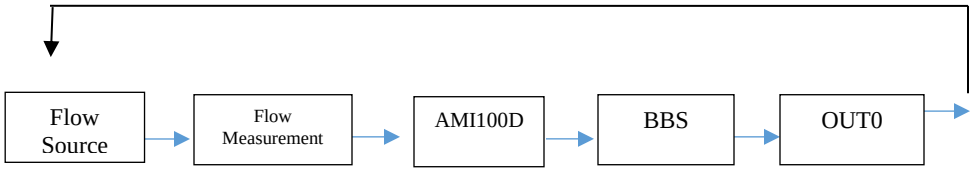

Figure 1 Schematic of Feedback Control of Flowrate

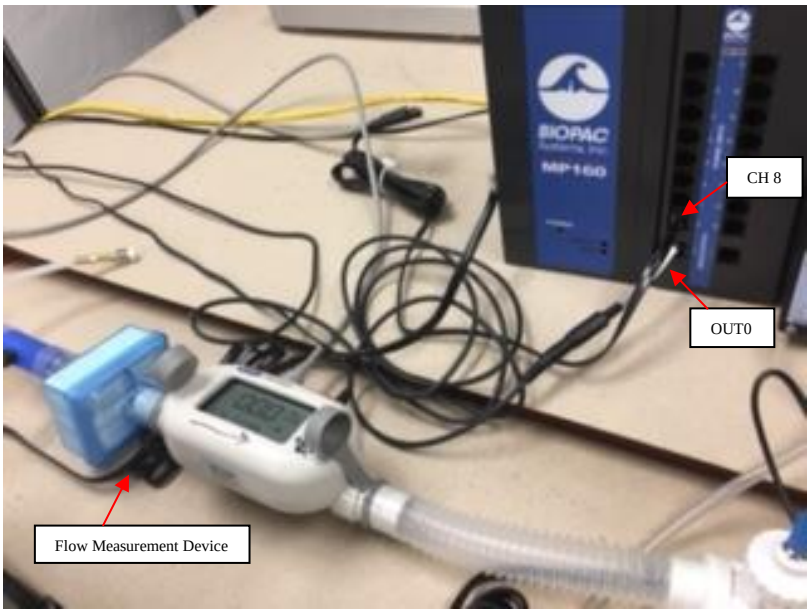

Figure 2a Flow meter signal fed to CH 8; OUT0 fed to flow source control port.

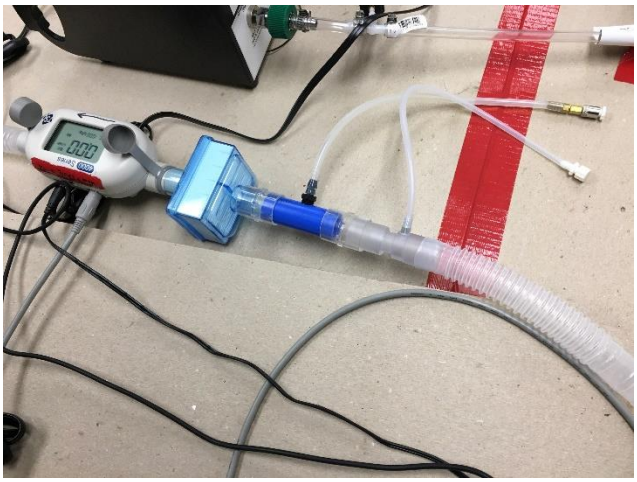

Figure 2b Bleed valves to affect flow.

To correctly applied voltage feedback to the flow source, a mapping of flow rate to voltage is needed. Figure 2 shows that mapping. Also, a regression fit to the mapping points is derived for use in the control macro code.

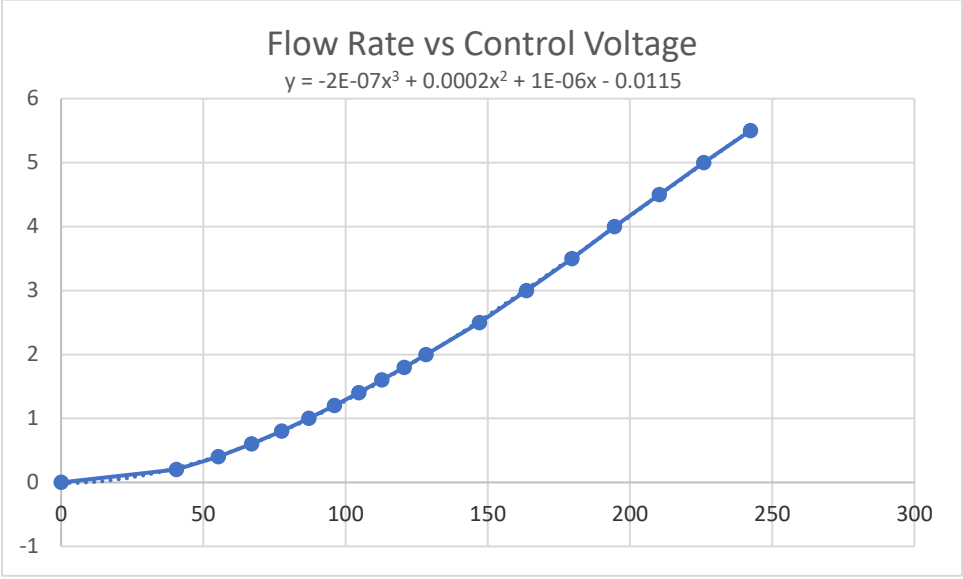

Figure 3 Mapping of Flow Rate-lpm (abscissa) vs Control Voltage-V (ordinate)

For use in the control code, the regression equation is modified as follows, *where* F = Flow, V = Voltage:

```
;relation between flowrate and voltage set to stimulator lpm --> voltage
;found through regression equation using discrete data points
V = (-0.0000002*F^3) + (0.0002*F^2) + (0.000006*F) - 0.0115
```

3.0 Algorithm

The algorithm is as follows:

The user inputs pulse width in seconds and pulse frequency in hertz which is limited by the duty cycle of 75% referenced to the pulse width entered.

The variable P# is the pulse repetition interval where it is used to sample the waveform for pulse amplitude values.

The variable H#, is that part of the time-domain waveform that is of zero amplitude.

Variable G# is the pulse width in seconds with amplitude V.

OnIdleAcquisition

- during an idle portion of the acquisition, selected the entire graph then extract the max value of the selected section of data

```
if S# = 0
    if (Ticks / 60) - T# > H#    ;upon the end of the zero value part of the waveform, set V = a value
        Select Wave 8
        Edit SelectAll
        Set HCursor HCursor2 - P#, HCursor2
        X = Max                  ;sample the waveform for amplitude value
        R = X / F                ;ratio of measured/desired flow rate
        GraphJournal Append STR$(R, 2)
        GraphJournal Append "\r"
        if R < 0.1
           R = 0.1
           V = V / R             ;adjust voltage value
           MP100 Set Out0, V ;apply voltage value to flow source
           S# = 1
           T# = Ticks / 60
        elseif R >= 0.1
          V = V / R              ;adjust voltage value
          MP100 Set Out0, V  ;apply voltage value to flow source
          S# = 1
          T# = Ticks / 60
        endif
    endif
elseif S# = 1
    if (Ticks / 60) - T# > G#   ;upon the end of the non-zero value of the waveform, set V = 0
        MP100 Set Out0, 0
        S# = 0
        T# = Ticks / 60
    endif
endif
```

4.0 Results

Figures 4a, 4b show an acquisition where the **peak pulse** flow rate, specified at 35 LPM, is maintained when leaks are introduced into the system.

After initial settling, a bleed valves are opened to allow leakage of the gas and to cause a drop in the peak pulse flow rate to 24 lpm. The control algorithm rightly corrects for this by adjusting the flow source voltage to produce MORE flow to compensate for the loss.

When the bleed valve is closed, the flow meter senses a transitory increase in peak pulse flow rate to 38 lpm to which the algorithm compensates by adjusting the flow source voltage to produce LESS flow.

Figure 4b shows a truncated view of the active channels as the full setup has 8 analog and 12 calculation channels each at 1000 S/s. The feedback technique, with the proper logic for toggling the flow source is unimpeded by these many channels and the high sample rate.

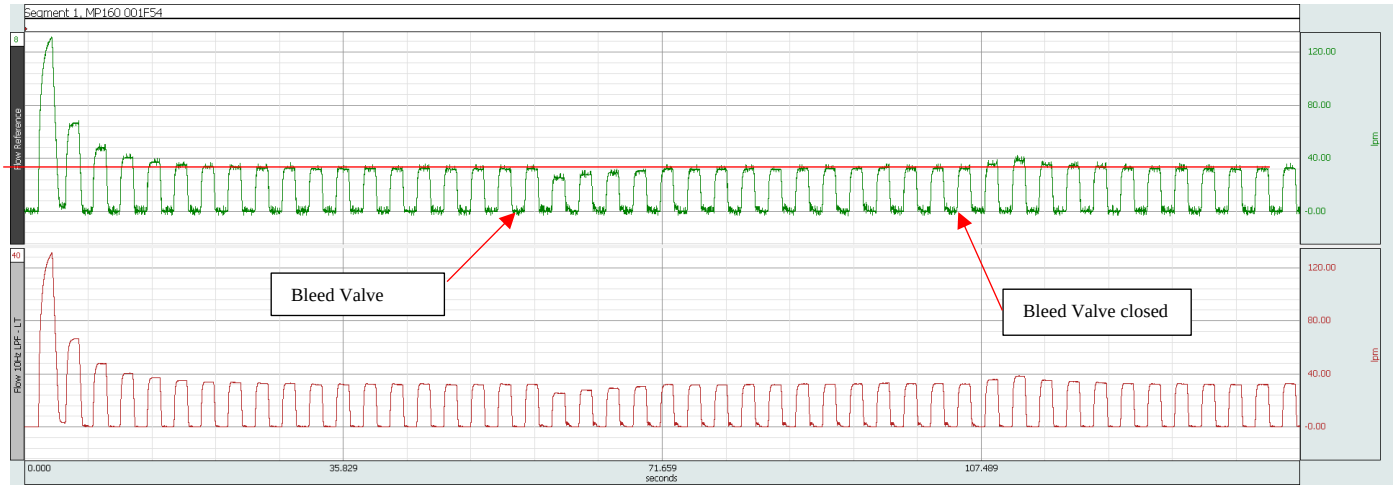

Figure 4a Pulse amplitude recovery (reference red line on CH 8) on startup, bleed valve open, bleed valve closed. Graph file with **2** channels @ 1000 S/s each. Pulse width = 1.5 sec, pulse freq = 0.3333 Hz

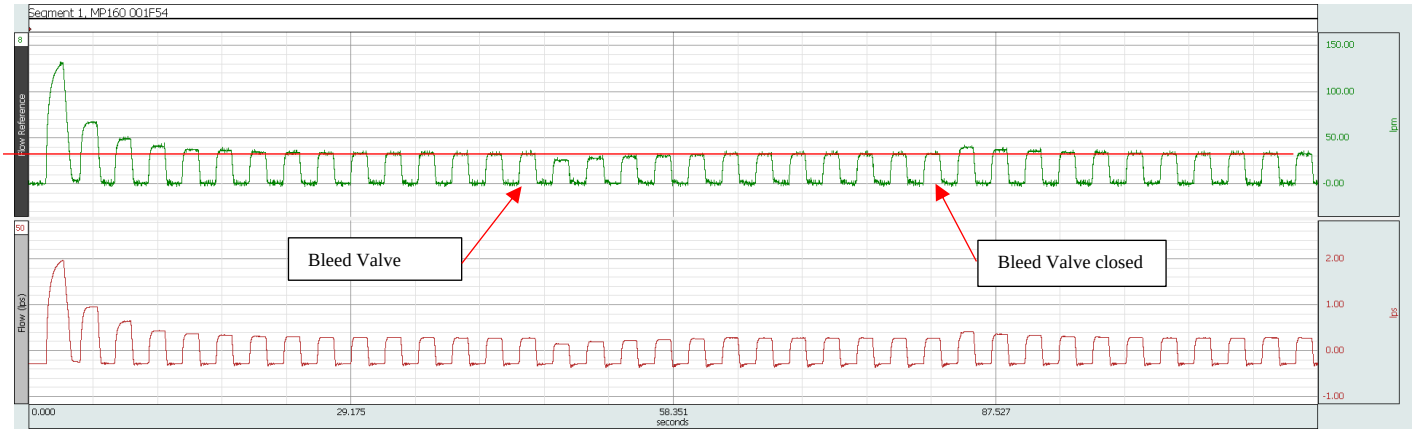

Figure 4b Pulse amplitude recovery (reference red line on CH 8) on startup, bleed valve open, bleed valve closed. Graph file with **20** channels @ 1000 S/s each. Pulse width = 1.5 sec, pulse freq = 0.3333

Appendix A | Channel Settings for Graph Template

Acquisition Settings

MP160
Serial #: MP160 001F54

Graph: Feedback Flow Ref pulse control test result.acq
Sampling rate: 1000 samples/sec
Record and replace previous data; save to memory.
Acquisition length: 2 hours (7200001 samples)

Smart Amplifier Configuration:

Acquired channels:

- Analog Channel 8
  - Label: Flow Reference
  - Scaling: 0.00030518 volts -> 0 lpm, 10.0003 volts -> 300 lpm
  - Units: lpm
- Calculation Channel 0
  - Label: Flow 10Hz LPF - LT
  - Units: lpm
  - Filter - IIR
  - Source channel: Analog Channel 8
  - Low pass at 10 Hz
  - Q: 0.707

Appendix B | Code

```
Lessons
OnOpenFile

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Ventilator feedback control.bbs;;;;;;;;;;;;;;;;;;;;
F = 25  ;lpm
T = 0
Array "PulseMetrics" Clear
; 0 = Pulse width
;1 = Pulse Repetition Interval

CloseAllGraphs

Open "", r, W$
if r <> 1
    Prompt "File could not be opened!", "OK"
else
    Select Window W$
    GraphJournal New Docked
    Call "EnterFlow"
endif
End
EnterFlow
;set the flow rate
Select Window W$
RepeatFR:
GetNumber "Enter the desired flow rate: ", "lpm", F, e, false
if e <> 0
  Prompt "Exiting the program...", "OK"
  Call "ExitProgram"
else
  if F < 0
     Prompt "Need to enter Flow Rate that is > 0.", "OK"
     goto RepeatFR
  else
    Call "PulseWidth"
```

```
  endif
endif
End
PulseWidth
ReEnterPulseWidth:
GetNumber "Enter Pulse Width", "sec", w, e, false
if e <> 0
   Prompt "User pressed 'Cancel or dialog error occurred. Exiting program.", "OK"
   Call "ExitProgram"
else
   if w < 0
      Prompt "Need to enter a pulse width that is > 0.", "OK"
      goto ReEnterPulseWidth
      Halt
   else
     W@ = STR$(w, 2)
     W@ = "Pulse Width entered (sec) = " + W@
     Prompt W@, "Accept", "Re-enter", "", r
      if r = 1
         Array "PulseMetrics" Set 0, w
         f = (2 / w)   ;max pulse repitition frequency based on 50% duty cycle of the entered PW
         Array "PulseMetrics" Set 1, f
         Call "PulseRepFreq"
     endif
     if r = 2
         goto ReEnterPulseWidth
     endif
   endif
endif
End
PulseRepFreq
ReEnterPulseRepFreq:
Array "PulseMetrics" Get 0, p
f = 1 /(1.25 * p)
F@ = STR$(f,2)
A@ = "Enter Pulse Repitition Frequency."
A@ = A@ + "Must be less than the PRF of "
```

```
A@ = A@ + F@
A@ = A@ + " based on a 75% duty cycle of the entered PW."
GetNumber A@, "Hz", g, e, false
if e <> 0
  Prompt "User pressed 'Cancel or dialog error occurred. Exiting program.", "OK"
  Call "ExitProgram"
else
  if g < 0
    Prompt "Need to enter a pulse repitition frequecy that is > 0.", "OK"
    goto ReEnterPulseRepFreq
    Halt
  else
   if g > f
     G@ = " " + STR$(g,2)
     H@ = G@ + " is greater than "
     H@ = H@ + F@
     H@ = H@ + "  Please re-enter a frequency that is lower."
     Prompt H@, "OK"
     goto ReEnterPulseRepFreq
     Halt
   else
     H@ = "Pulse repitition frequency entered (Hz) = " + G@
     Prompt H@, "Accept", "Re-enter", "", r
     if r = 1
       Array "PulseMetrics" Set 1, g
       Call "SetSTIM"
     endif
     if r = 2
       goto ReEnterPulseRepFreq
     endif
   endif
endif

End
SetSTIM
;relation between flowrate and voltage set to stimulator lpm --> voltage
;found through regression equation using discrete data points
```

```
Select Window W$

V = (-0.0000002*F^3) + (0.0002*F^2) + (0.000006*F) - 0.0115

Call "NewButtons"
End
StartMPAcq
Select Window W$
Array "PulseMetrics" Get 0, G#
Array "PulseMetrics" Get 1, g

;derive the time interval between the actual pulses
P# = (1 / g)
H# = P# - G#
RemoveAllButtons
CreateButton "Done" "Done"
MP100 StartAcq
End
OnStartAcquisition
Select Window W$

T# = Ticks / 60
S# = 0
MP100 Set Out0, 0
End
Done
Select Window W$
MP100 Set Stim 0, Off
MP100 StopAcq
Call "NewButtons"
End
NewButtons
RemoveAllButtons
CreateButton "Enter Flowrate/Pulse Parameters" "EnterFlow"
CreateButton "Start Acq" "StartMPAcq"
CreateButton "Exit Program" "ExitProgram"
End
```

```
ExitProgram
;;AppExit
Select Window W$
CloseGraph
End
OnIdleAcquisition
if S# = 0
    if (Ticks / 60) - T# > H#
        Select Wave 8
        Edit SelectAll
        Set HCursor HCursor2 - P#, HCursor2
        X = Max
        R = X / F
        GraphJournal Append STR$(R, 2)
        GraphJournal Append "\r"
        if R < 0.1
           R = 0.1
           V = V / R
           MP100 Set Out0, V
           S# = 1
           T# = Ticks / 60
        elseif R >= 0.1
          V = V / R
          MP100 Set Out0, V
          S# = 1
          T# = Ticks / 60
        endif
    endif
elseif S# = 1
    if (Ticks / 60) - T# > G#
        MP100 Set Out0, 0
        S# = 0
        T# = Ticks / 60
    endif
endif
End
```