pEchoVar: Re-publishing Variables Under a Different Name

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

`project-pavlab/appdocs/app_pechovar`

1 Overview

The pEchoVar application is a lightweight process that runs without any user interaction for ”echoing” the posting of specified variable-value pairs with a follow-on posting having different variable name. For example the posting of `FOO=5.5` could be echoed such that `BAR=5.5` immediately follows the first posting. The motivation for this tool was to convert, for example, a posting such as `GPS_X` to become `NAV_X`. The former is the output of a particular device, and the latter is a de facto standard for representing the vehicle’s longitudinal position in local coordinates.

2 Using pEchoVar

Configuring `pEchoVar` minimally involves the specification of one or more `echo` or `flip` mapping events. It may also optionally involve specifying one or more logic conditions that must be met before mapping events are posted.

2.1 Configuring Echo Mapping Events

An *echo event mapping* maps one MOOS variable to another. Each mapping requires one line using the `echo` configuration parameter of the form:

```
echo = <MOOSVar> -> <MOOSVar>
```

The source `<MOOSVar>` and target `<MOOSVar>` components are case sensitive since they are MOOS variables. A source variable can be echoed to more than one target variable. If the set of lines forms a cycle, this will be detected and pEchoVar post a configuration and run warning and cease to perform any function whatsoever. An example configuration is given in Listing 1.

Listing 2.1: An example pEchoVar configuration block.

```
//------------------------------------------------------------
// pEchoVar configuration block

ProcessConfig = pEchoVar
{
  AppTick   = 20
  CommsTick = 20

  echo = GPS_X           ->  NAV_X
  echo = GPS_Y           ->  NAV_Y
  echo = COMPASS_HEADING ->  NAV_HEADING
  echo = GPS_SPEED       ->  NAV_SPEED
}
```

2.2 Configuring Flip Mapping Events

The pEchoVar application can be used to "flip" a variable rather than doing a simple echo. A flipped variable, like an echoed variable, is one that is republished under a different name, but a flipped variable parses the contents of a string comprised of a series of `variable=value` comma-separated pairs, and republishes a portion of the series under the new variable name. For example, the following string,

```
ALPHA = "xpos=23, ypos=-49, depth=20, age=19.3, certainty=1"
```

may be flipped to publish the below new string, with the fields `xpos`, `ypos`, and `depth` replaced with `x`, `y`, `vehicle_depth` respectively.

```
BRAVO = "x=23, y=-49, vehicle_depth=20"
```

The above "flip relationship" is configured with the `flip` configuration parameter with the following form:

```
flip:<key>   = source_variable  = <variable>
flip:<key>   = dest_variable    = <variable>
flip:<key>   = source_separator = <separator>
```

```
flip:<key>   = dest_separator   = <separator>
flip:<key>   = filter           = <variable>=<value>
flip:<key>   = component        = <old-field> -> <new-field>
flip:<key>   = component        = <old-field> -> <new-field>
```

The relationship is distinguished with a `<key>`, and several components. The `source_variable` and `dest_variable` components are mandatory and must be different. The `source_separator` and `dest_separator` components are optional with default values being the string `","`. Fields in the source variable will only be included in the destination variable if they are specified in a component mapping `<old-vield> -> <new-field>`. The example configuration in Listing 2 implements the above described example flip mapping. In this case only postings that satisfy the further filter, `certainty=1`, will be posted.

Listing 2.2: An example pEchoVar configuration block with flip mappings.

```
//--------------------------------------------------------------
// pEchoVar configuration block

ProcessConfig = pEchoVar
{
  AppTick   = 10
  CommsTick = 10

  flip:1   = source_variable  = ALPHA
  flip:1   = dest_variable    = BRAVO
  flip:1   = source_separator = ,
  flip:1   = dest_separator   = ,
  flip:1   = filter           = certainty=1
  flip:1   = component        = ypos -> y
  flip:1   = component        = xpos -> x
}
```

Some caution should be noted with flip mappings - the current implementation does not check for cycles, as is done with echo mappings.

2.3 Applying Conditions to the Echo and Flip Operation

The execution of the mappings configured in pEchoVar may be configured to depend on one or more logic conditions. If conditions are specified in the configuration block, all specified logic conditions must be met or else the posting of echo and flip mappings will be suspended. The logic conditions are configured with the `condition` parameter as follows:

```
condition = <logic-expression>
```

The `<logic-expression>` syntax is described in the Appendix document on logic utilities, and may involve the simple comparison of MOOS variables to specified literal values, or the comparison of MOOS variables to one another. If a `condition` parameter is specified, pEchoVar will automatically subscribe to all MOOS variables used in the condition expressions.

2.4 Holding Outgoing Messages Until Conditions are Met

If the conditions are not met, all incoming mail messages that would otherwise result in an echo or flip posting, are held. When or if the conditions are met at some point later, those mail messages are processed in the order received and echo and flip mappings may be posted en masse. However, if several mail messages for a given MOOS variable are received and stored while conditions are unmet, only the latest received mail message for that variable will be processed. As an example, consider `pEchoVar` configured with the below two lines:

```
echo      = FOO -> BAR
condition = DEGREES <= 32
```

If the condition is not met for some period of time, and the following mail were received during this interval: `FOO="apples"`, `FOO="pears"`, `FOO="grapes"`, followed by `DEGREES=30`, then `pEchoVar` would post `BAR="grapes"` immediately on the very iteration that the `DEGREES=30` message was received. Note that `BAR="apples"` and `BAR="pears"` would never be posted. This is to help ensure that the `pEchoVar` memory doesn't grow unbounded by holding onto all mail while conditions are unmet.

The user may alternatively configure `pEchoVar` to *not* hold incoming mail messages when or if it is in a state where its logic conditions are not met. This can be done with the `hold_messages` parameter:

```
hold_messages = false   // The default is true
```

When configured this way, upon meeting the specified logic conditions, `pEchoVar` will begin processing echo and flip mappings when or if new mail messages are received relevant to the mappings. In the above example, once `DEGREES=30` is received by `pEchoVar`, nothing would be posted until new incoming mail on the variable `FOO` is received (not even `BAR="grapes"`).

2.5 Limiting the Echo Posting Frequency to the AppTick Setting

By default, when the conditions are met, an echo posting is made once for each incoming piece of mail related that echo mapping. If `FOO` is echo mapped to `BAR`, and if 40 pieces of incoming mail for `FOO` are received on one iteration, 40 postings are made to `BAR` on that iteration. Instead one may wish that, on each iteration where there is posting ready for `BAR`, that only the latest value for `BAR` be made. This may be arranged with the `echo_latest_only` parameter:

```
echo_latest_only = true    // The default is false
```

In this case, the frequency of postings to `BAR` and all other echo mappings will occur at most at a frequency equal to the `AppTick` setting.

3 Configuring for Vehicle Simulation with pEchoVar

When in simulation mode with uSimMarine, the navigation information is generated by the simulator and not the sensors such as GPS or compass as indicated in lines 9-12 in Listing 1. The simulator instead produces `USM_*` values which can be echoed as `NAV_*` values as shown in Listing 3.

Listing 3.3: An example pEchoVar configuration block during simulation.

```
//------------------------------------------------------------
// pEchoVar configuration block (for simulation mode)

ProcessConfig = pEchoVar
{
  AppTick   = 20
  CommsTick = 20

  echo = USM_X       ->  NAV_X
  echo = USM_Y       ->  NAV_Y
  echo = USM_HEADING ->  NAV_HEADING
  echo = USM_SPEED   ->  NAV_SPEED
}
```

Note in more recent versions of `uSimMarine` the simulator output may be changed to have a `NAV_` prefix by setting `prefix = NAV_`, obviating the use of `pEchoVar` configured as above.

4 Configuration Parameters for pEchoVar

The following parameters are defined for `pEchoVar`. A more detailed description is provided in other parts of this section. Parameters having default values are indicated so.

Listing 4.4: Configuration Parameters for `pEchoVar`.

`echo`:	A mapping from one MOOS variable to another constituting an echo. Section 2.1.
`echo_latest_only`:	If true, only the latest value of variable will be echoed on each iteration, even if several pieces of incoming mail have been received since the last posting. Legal values: true, false. The default is false. Section 2.1.
`condition`:	A logic condition that must be met or all echo and flip publications are held. Section 2.3.
`flip`:	A description of how components from one variable are re-posted under another MOOS variable. Section 2.2.
`hold_messages`:	If true, messages are held when conditions are not met for later processing when logic conditions are indeed met. Legal values: true, false. The default is true. Section 2.3.

5 Publications and Subscriptions for pEchoVar

The interface for `pEchoVar`, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ pEchoVar --interface or -i
```

5.1 Variables Posted by pEchoVar

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility. Section 6.
- <USER-DEFINED>: Any MOOS variable specified in either the echo or flip config parameters.

5.2 Variables Subscribed for by pEchoVar

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- <USER-DEFINED>: Any MOOS variables found in the antecedent of either the echo or flip mappings. It will also subscribe for any MOOS variable found in any of its logic conditions.

6 Terminal and AppCast Output

The pEchoVar application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 5 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. See the documentation on uMAC viewing utilities for more on appcasting and viewing appcasts. The counter on the end of line 2 in parentheses is incremented on each iteration of pEchoVar, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

Listing 6.5: Example terminal or appcast output for pEchoVar.

```
 1  ===================================================================
 2  pEchoVar alpha                                              0/0(81)
 3  ===================================================================
 4  conditions_met:   true
 5  hold_messages:    true
 6  echo_latest_only: false
 7
 8  ==================================================
 9  Echoes: (5)
10  ==================================================
11
12  Source          Dest           Hits  Posts
13  ---------  ---  -------------  ----  -----
14  NAV_X      -->  NAV_XX         324   324
15  NAV_X      -->  NAV_XPOS       324   324
16  NAV_Y      -->  NAV_YY         324   324
17  NAV_Y      -->  NAV_YPOS       324   324
18  NAV_SPEED  -->  NAV_SPEED_ALT  324   324
19
20  ==================================================
21  Flips: (1)
22  ==================================================
23
24                                        Src  Dest              Old    New
25  Key  Hits  Source             Dest    Sep  Sep   Filter      Field  Field
26  ---  ----  -----------------  ------  ---  ----  ----------  -----  -----
```

```
27  1    162   NODE_REPORT_LOCAL  FOOBAR  ,     #     type:kayak  X      xpos
28  1    162   NODE_REPORT_LOCAL  FOOBAR  ,     #     type:kayak  Y      ypos
```

Lines 4 indicates whether or not any specified logic conditions have been met. This line will also read `true` even if no logic conditions were provided. Lines 5 and 6 simply confirm the user's settings for the `hold_messages` and `echo_latest_only` parameters discussed in Sections 2.4 and 2.5 respectively.

Lines 12-18 convey the configured echo mappings, one for each line. At the end of each line, the *Hits* column shows the number of incoming mails received for that variable. The number of times it is echoed, or re-posted is shown under the *Posts* column. When `echo_latest_only` is false, these numbers should match. Lines 20-28 convey the configure flips. A single flip configuration, identified by its key, may have several lines, as in this example. Here the only difference between lines 27 and 28 are the flip components. One maps `X` to `xpos`, and the other maps `Y` to `ypos`.

The terminal or appcast output shown in Listing 5 above may be seen first hand by running the Alpha example mission. The below configuration block in Listing 6 corresponds to the above appcast output. The user just needs to add `pEchoVar` to the Antler launch list.

Listing 6.6: Example pEchoVar configuration from the Alpha example mission.

```
 1  ProcessConfig = pEchoVar
 2  {
 3    AppTick = 1
 4    CommsTick = 1
 5
 6    echo = NAV_X     -> NAV_XX
 7    echo = NAV_X     -> NAV_XPOS
 8    echo = NAV_Y     -> NAV_YY
 9    echo = NAV_Y     -> NAV_YPOS
10    //echo = NAV_YY    -> FOOBAR
11    //echo = FOOBAR    -> NAV_Y
12    echo = NAV_SPEED -> NAV_SPEED_ALT
13
14    FLIP:1    = source_variable  = NODE_REPORT_LOCAL
15    FLIP:1    = dest_variable    = FOOBAR
16    FLIP:1    = source_separator = ,
17    FLIP:1    = dest_separator   = #
18    FLIP:1    = filter = type == kayak
19    FLIP:1    = component = X -> xpos
20    FLIP:1    = component = Y -> ypos
21  }
```

The two echo mappings in lines 10 and 11 may be commented out to demonstrate the detection of echo mapping cycles. The pairs of mappings in Lines 6-7 and 8-9 demonstrate that a single incoming variable may be mapped to multiple destinations.