Scheduling Communications — pScheduler

Paul Newman

March 17, 2009

Abstract

This document will give you a description of pScheduler — a process useful for generating and marshalling communication packets within a MOOS community.

1 Introduction

pScheduler is a simple tool for generating and responding to messages sent to the MOOSDB by processes in a MOOS community. It supports three competencies, which will now be described. Example configuration blocks will also be given.

SEQUENCES A looping sequence of messages can be created and published by pScheduler . Each element of a sequence is specified in the configuration block with a line: SEQUENCE = PUBLISH_NAME @ VALUE : TIME_OFFSET . For example, the configuration in Figure 1 would write the variable LIGHT_CONTROL to the DB every 2 seconds with its String value alternating between "ON" and "OFF". The total sequence period is given by the maximum TIME_OFFSET parameter. Figure 3 shows a more complicated configuration block, in which a sequence is constructed from several different variables.

```
ProcessConfig  = pScheduler
{
    SEQUENCE   = LIGHT_CONTROL @ ON  :  2
    SEQUENCE   = LIGHT_CONTROL @ OFF  :  4
}
```

Figure 1: Configuring a sequence with pScheduler

TIMERS A "timer" allows a variable to be written to the database repetitively. A timer can be started and stopped by publication (by some other

application) of user-specified variables. The scheduler can also be told to derive the value of the periodic variable from another MOOS variable, which, if it arrives in the scheduler's mail box, overrides the initial value. This sounds complicated but isn't. An example is useful. The general syntax is as follows

`TIMER = PUBLISH_NAME @ TIME , START_VARIABLE, STOP_VARIABLE,VALUE_VARIABLE -> VALUE`

Figure 2 shows a typical configuration block. In this case the variable

```
ProcessConfig = pScheduler
{
    TIMER = CAMERA_CONTROL @ 4.0 , MISSION_START,
MISSION_END,DESIRED_CAMERA_ANGLE -> 0.0
    TIMER = CAMERA_GRAB @ 4.0, MISSION_START -> GRAB
    TIMER = CAMERA_GRAB @ 4.0, -> GRAB
}
```

Figure 2: Configuring a timer with `pScheduler`

`CAMERA_CONTROL` will be published every 4 seconds after some third-party application writes (publishes) the `MISSION_START` variable. If, while the timer is active, the variable `CAMERA_ANGLE` is published, the value of `CAMERA_CONTROL` will be copied from that message. If `CAMERA_ANGLE` is never published it will always have its default value of 0.0. The timer turns off if any process publishes `MISSION_END` . The second example is simpler and writes `CAMERA_GRAB` with value "GRAB" every 4 seconds as soon as `MISSION_START` is written. The third version starts immediately (note the comma is required...). Figure 3 shows a few more examples of the configuration of timers in `pScheduler` .

RESPONSES The last competency is one of responding to the publication of one variable with the publication of one or more different variables. The syntax is obvious: `RESPONSE = STIMULUS_VARIABLE : RESPONSE_VARIABLE @ VALUE,RESPONSE_VARIABLE2 @ VALUE,....` . Here `STIMULUS_VARIABLE` is the name of the variable we wish `pScheduler` to respond to, and after the colon comes a command-separated list of response variables with the values they should contain. Figure 3 shows some clear example configurations of "responses" in `pScheduler`

```
ProcessConfig = pScheduler
{
    //generate a sequence 6 seconds long ....
    // VAR1 will fire after 1 second
    // VAR2 fire after 3 seconds
    // VAR3 fire after 6 seconds
    // one second later VAR1 will fire again..repeat...
    SEQUENCE = VAR1 : RED @ 1 SEQUENCE = VAR2 : ORANGE @ 3 SEQUENCE =
    VAR3 : GREEN @ 6

    //generate a timer that writes "VAR_T1" with value "TimerData"
    TIMER = VAR_T1 @ 3.0, -> TimerData

    // generate a timer that writes "VAR_T2" with a string version
    // of the current value of DB_TIME (published by the DB) TIMER =
    VAR_T2 @ 2.0  ,,,DB_TIME -> TimerData

    // generate a timer that writes "VAR_T3" with the current
    // value of DB_TIME (published by the DB)
    // which only starts when "GO_T3" is published and stops
    // when "STOP_T3" is published
    TIMER = VAR_T3 @ 4.0,GO_T3,STOP_T3,DB_TIME -> TimerData

    //generate a response to "SURPRISE_ME". The variable "BOO"
    // takes on string value "HOO" and variable R9 has value
    //"get_a_grip"
    RESPONSE = SURPRISE_ME : BOO @ HOO, R9 @ get_a_grip

    //generate a response to "DB_TIME".
    RESPONSE = DB_TIME : ACKNOWLEDGMENT @ I_GOT_THE_TIME

}
```

Figure 3: An extended `pScheduler` configuration block