

2.680
UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 12: Augmenting the Helm with Third Party Behaviors

April 7th, 2026



Web: <http://oceanai.mit.edu/2.680>
Email:
Mike Benjamin, mikerb@mit.edu

MIT 2.680 Spring 2026 – Marine Autonomy – Lecture: "Writing Behaviors" Photo by Arjan Vermeij GLINT '09

1


Why Build Your Own Behavior?

- Sometimes a new desired capability can be achieved by writing a MOOS app and streaming *updates* to an existing behavior.
- Sometimes the new desired result can be achieved by stitching together existing behaviors
- Sometimes is best to write a new behavior.


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

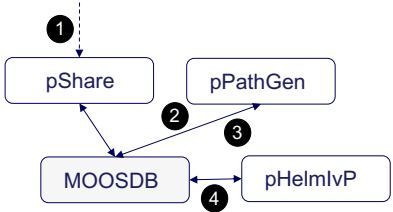
2



Recall the TSP Lab



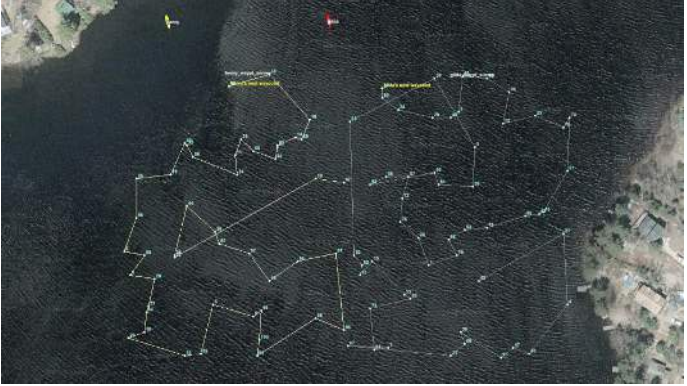
- Each vehicle received a set of visit points.
- A path planner generated a sequence, i.e., a path plan
- The path was updated to an existing waypoint behavior via the updates interface.



```

graph TD
    1((1)) --> pShare
    pShare -- 2 --> pPathGen
    pPathGen -- 3 --> MOOSDB
    MOOSDB -- 4 --> pHelmIvP
  
```

- 1 Visit points received from shoreside
- 2 pGenPath determines an ordering
- 3 pGenPath publishes the path via an updates posting
- 4 The helm's waypoint behavior is updated and executed



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

3



Why Build Your Own Behavior?



- Sometimes a new desired capability can be achieved by writing a MOOS app and streaming *updates* to an existing behavior.
- ➔ • Sometimes the new desired result can be achieved by stitching together existing behaviors
- Sometimes is best to write a new behavior.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

4



Recall the Bravo UUV Surface Mission (Lab 07)



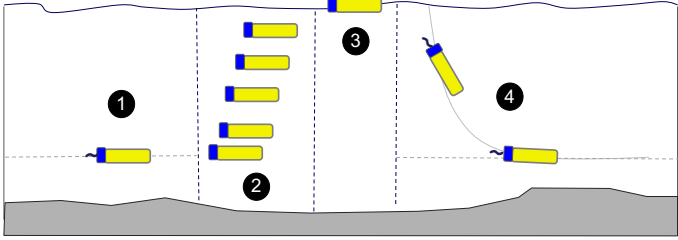
- 1 The UUV operates at a prescribed depth
- 2 Periodically it stops, floats to the surface
- 3 After arriving at the surface, it waits N seconds (presumably for a GPS fix)
- 4 After finishing at the surface, it re-dives to its normal operational depth

- 1 Constant Depth Loiter Timer

- 2

- 3 Timer

- 4 Constant Depth Loiter Timer



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

5



Why Build Your Own Behavior?



- Sometimes a new desired capability can be achieved by writing a MOOS app and streaming *updates* to an existing behavior.
- Sometimes the new desired result can be achieved by stitching together existing behaviors
- ➔ • Sometimes is best to write a new behavior.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

6



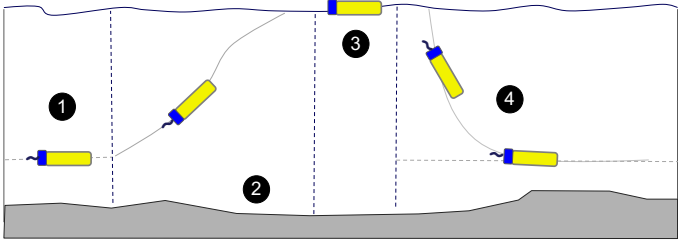
The BHV_PeriodicSurface Behavior



- 1 The UUV operates at a prescribed depth for a time, specified by `period`.
- 2 It then drives to the surface at speed `ascent_speed`, gradually reducing its speed to zero at depth `zero_speed_depth`.
- 3 At the surface it waits for `mark_variable` to be posted. Or until `max_time_at_surface` has elapsed. While at the surface it publishes status to `atsurface_status_var` MOOS var.
- 4 After finishing at the surface, it re-dives to its normal operational depth, resetting its period clock.

```

period          = 200
mark_variable   = GPS_RECEIVED
ascent_speed    = 2.0
ascent_grade    = linear
zero_speed_depth = 4
max_time_at_surface = 600
atsurface_status_var = FIX_STATUS
            
```



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

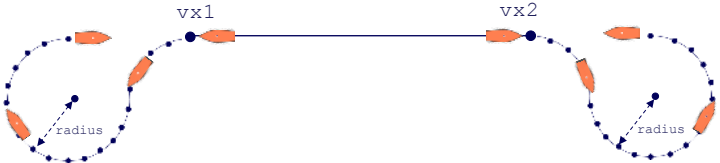
7



The Leg Run Behavior

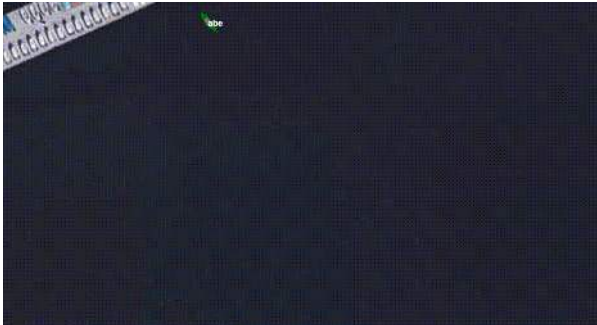


The **LegRun** behavior is originally designed to facilitate the testing of a marine robotic platform by enabling the vehicle traverse back and forth over a legrun track line prescribed by two vertices. After reaching the end of the track line, the vehicle switches into a Williamson turn mode to ensure the vehicle aligns with the track line when starting in the other direction.



Configuration Parameters:

vx1	Location of vertex 1
vx2	Location of vertex 2
speed	Traversal speed, m/s
turn1_rad	Turn radius at vx1
turn2_rad	Turn radius at vx2
turn_rad	Turn radius at both ends



Motivation Overview

Waypoint Behavior

Dynamic Updates

Loiter Behavior


Min/Max Depth

OpRegion Behavior


StationKeep Behavior

Michael Benjamin ©2024 MIT Dept of Mechanical Engineering

8




The Leg Run Behavior (with Synchronization)



Coordination: Introducing inter-vehicle messaging, the LegRun behavior can be configured to coordinate timing between vehicles.

vx1	Location of vertex 1
vx2	Location of vertex 1
speed	Traversal speed
coord	true/false
Coord_on_leg	true/false



Motivation Overview

Waypoint Behavior

Dynamic Updates

Loiter Behavior


Min/Max Depth

OpRegion Behavior


StationKeep Behavior

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

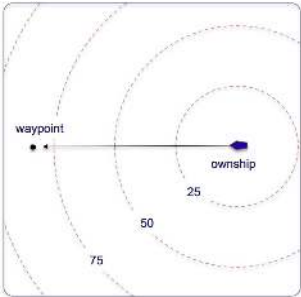
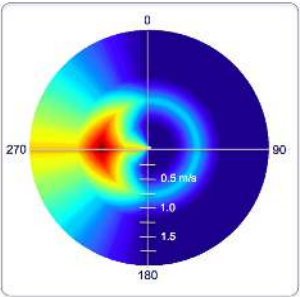
9



When a New Behavior is Needed



- For *transiting* type problems – feeding the waypoint behavior may be ok. Traversing visit points, generating waypoints based on currents etc.
- The waypoint behavior has a convenient unimodal objective function:

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

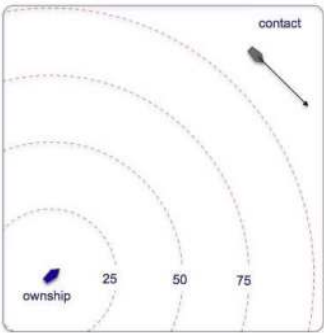
10

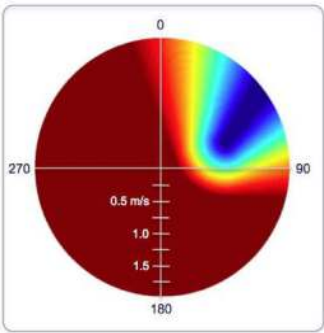


When a New Behavior is Needed



- For other types of behaviors, e.g., collision avoidance, the updates parameter is less amenable to calculation by external MOOS apps.
- The collision avoidance behavior has a multi-modal (plateaus) objective function:





- There is no way to “feed” this behavior into a waypoint behavior. (Which point on the plateau would you choose?)

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

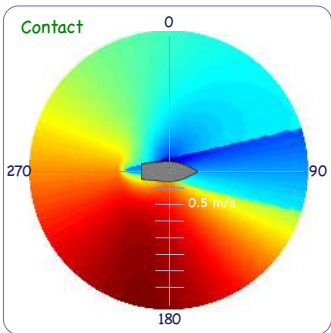
11

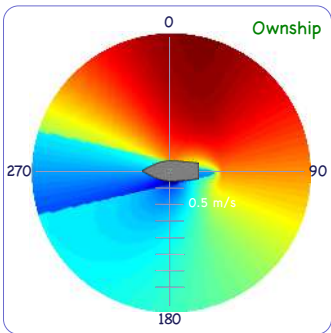


When a New Behavior is Needed



- For other types of behaviors, e.g., collision avoidance, the updates parameter is less amenable to calculation by external MOOS apps.
- The collision avoidance behavior has a multi-modal (plateaus) objective function:





- The Rule 14 COLREGS behavior with a bias to early starboard turns.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer



Posting Messages

A Simple Example

Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

12

Adding a New Behavior

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

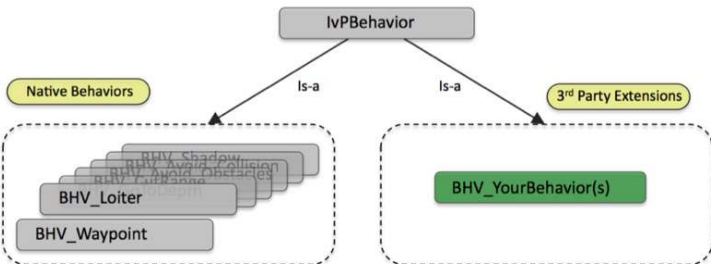
Michael Benjamin ©2026

13

Adding a Behavior

- Your behavior will also be a subclass of the IVPBehavior superclass
- It will inherit functionality of the IVPBehavior superclass
- It will overload virtual functions of the IVPBehavior superclass.



```

classDiagram
    class IVPBehavior
    class NativeBehaviors
    class BHV_Shadow
    class BHV_AvoidObstacles
    class BHV_Loiter
    class BHV_Waypoint
    class ThirdPartyExtensions
    class BHV_YourBehavior_s

    IVPBehavior <|-- NativeBehaviors
    IVPBehavior <|-- ThirdPartyExtensions
    NativeBehaviors <|-- BHV_Shadow
    NativeBehaviors <|-- BHV_AvoidObstacles
    NativeBehaviors <|-- BHV_Loiter
    NativeBehaviors <|-- BHV_Waypoint
    ThirdPartyExtensions <|-- BHV_YourBehavior_s
    
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

14



Generating the Files for a New Behavior



- A new behavior may be created by copying a similar behavior and proceed by editing
- Or you can generate a new behavior from scratch with the `GenBehavior` script:

```

$ cd lib_behaviors
$ GenBehavior BHV_FooBar "Bob T. Builder"
BHV_FooBar generated. Don't forget to to update your CMakeLists.txt file


$ ls
BHV_BHV_FooBar.cpp  BHV_BHV_FooBar.h

```


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

15



Adding the New Behavior to the Build System



You already have an example of a third-party behavior in your moos-ivp-extend tree. It should be built along with all your other code.

```

$ cd moos-ivp-extend
$ cd src/lib_behaviors-test
$ ls
AOF_SimpleWaypoint.cpp  BHV_SimpleWaypoint.cpp  CMakeLists.txt
AOF_SimpleWaypoint.h   BHV_SimpleWaypoint.h   README

```

Look at the [README](#) file and follow the instructions.

```

$ more README
To use the dynamic loading of behaviors, you need to set the
following environment variable (in your .bashrc file):


IVP_BEHAVIOR_DIRS='/home/bob/moos-ivp-extend/lib'

```


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

16



Adding the New Behavior to the Build System



- Regardless of whether the behavior was generated from scratch, or copied, you will need to update your `CMakeLists.txt` file:

```
#-----
#
#                               BHV_SimpleWaypoint
#-----
ADD_LIBRARY(BHV_SimpleWaypoint SHARED
  BHV_SimpleWaypoint.cpp AOF_SimpleWaypoint.cpp)
TARGET_LINK_LIBRARIES(BHV_SimpleWaypoint
  helmivp
  behaviors
  ivpbuild
  logic
  ivpcore
  bhvutil
  mbutil
  geometry
  ${SYSTEM_LIBS} )
```

- Make a copy of the following block. Edited to reflect your behavior and any other C++ support code.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

17



Creating Your Own Behavior Library



- You may decide to create your own behavior library rather than adding a new behavior to the `lib_behaviors-test` library.
- In this case you can copy the whole `lib_behaviors-test` library using it as a template.

- If you make your own library folder, just remember to add the folder to the `CMakeLists.txt` file in `moos-ivp-extend/src`

```
$ cd moos-ivp-extend/src/
$ more CMakeLists.txt

#=====
# List the subdirectories to build...
#=====
ADD_SUBDIRECTORY(lib_behaviors-test)
ADD_SUBDIRECTORY(lib_behaviors-jane)
ADD_SUBDIRECTORY(pXRelayTest)
ADD_SUBDIRECTORY(pExampleApp)
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

18





Behavior Functions

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

19

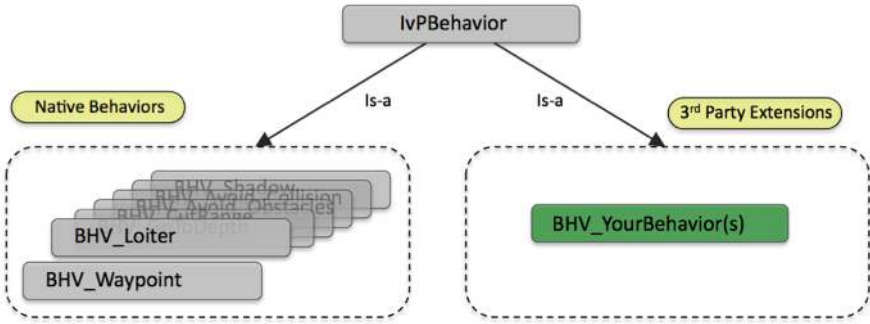


Behavior Functions

Inherited from IvPBehavior Superclass



- From a behavior **user** perspective, behaviors have common configuration parameters.
- From a behavior **author** perspective, they have common overloadable functions and callable runtime functions and libraries.
- These common behavior features are derived by inheritance from the **IvPBehavior** superclass.



```


classDiagram
    class IvPBehavior
    class NativeBehaviors
    class ThirdPartyExtensions
    class BHV_Loiter
    class BHV_Waypoint
    class BHV_Shadow
    class BHV_AvoidObstacles
    class BHV_AvoidCollision
    class BHV_Pathfinder
    class BHV_YourBehavior_s

    IvPBehavior --> NativeBehaviors : is-a
    IvPBehavior --> ThirdPartyExtensions : is-a
    
```


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

20



Behavior Functions are Related to Behavior State



Recall that Behaviors may be in one of four states:

Idle

Running

Active

Completed

The **idle** state: a behavior has not met its run conditions, as defined by the **condition** parameter.

The **running** state: a behavior has met its run conditions, as defined by the **condition** parameter.

The **active** state: a behavior is in the running state, and it is producing an objective function over the helm's configured decision space.

The **completed** state: a behavior has completed. Completion is defined by the behavior author or may be due to a **duration** timeout defined generally for all behaviors.

}

}

Solely Based on conditions

Based on author implementation

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

MIT Dept of Mechanical Engineering

21



Helm-Invoked Behavior Functions



- MOOS Apps have several key overloadable functions – this is where the uniqueness of the app is implemented.
- IvP behaviors also have several overloadable functions - invoked by the helm under certain specific situations

MOOSApp Overloadable functions

```

OnStartup ()
{
  ...
}
OnNewMail ()
{
  ...
}
Iterate ()
{
  ...
}
                
```

IvPBehavior Overloadable functions
(based on *state*)

```

onRunState ()
{ }
onIdleState ()
{ }
onCompleteState ()
{ }
onIdleToRunState ()
{ }
onRunToIdleState ()
{ }
onInactiveState ()
{ }
                
```

```

setParam ()
{ }
postConfigStatus ()
{ }
onSetParamComplete ()
{ }
onHelmStart ()
{ }
onRunStatePrior ()
{ }
                
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

MIT Dept of Mechanical Engineering

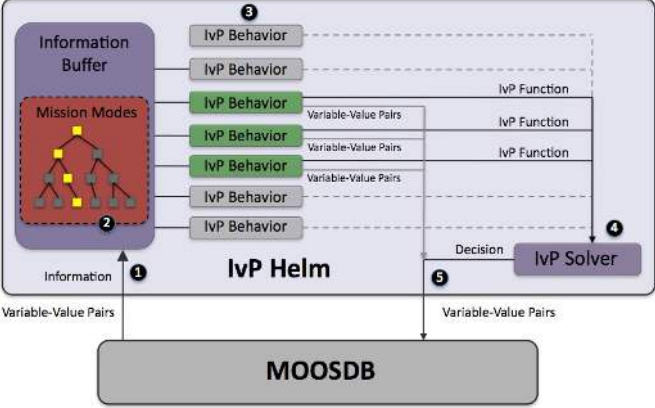
22



Interval Programming and the IvP Helm




- 1 Mail is read in the MOOS OnNewMail() function and applied to a local buffer.
- 2 The helm mode is determined and set of running behaviors determined.
- 3 Behaviors do their thing – posting MOOS variables and an IvP function.
- 4 Competing behaviors are resolved with the IvP solver.
- 5 The Helm decision and any behavior postings are published to the MOOSDB.



Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

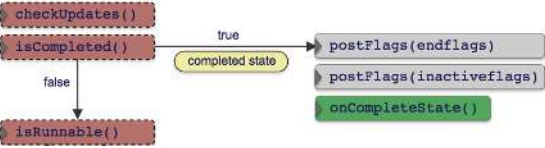
23



Helm-Invoked Behavior Functions

Chain of Contingency Events





```

onRunState ()
{}
OnIdleState ()
{}
onCompleteState ()
{}
onIdleToRunState ()
{}
onRunToIdleState ()
{}
onInactiveState ()
{}
    
```


Key to Helm-Invoked Functions

- Immutable
- Overloaded
- Arguments from mission configuration

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

24



Helm-Invoked Behavior Functions

Chain of Contingency Events



```

graph TD
    checkUpdates[checkUpdates()] --> isCompleted[isCompleted()]
    isCompleted -- true --> completed_state[completed state]
    isCompleted -- false --> isRunnable[isRunnable()]
    isRunnable -- true --> running_state[Running state]
    isRunnable -- false --> idle_state[Idle state]
    
    completed_state --> postFlags_end[postFlags(endflags)]
    completed_state --> postFlags_inactive[postFlags(inactiveflags)]
    completed_state --> onCompleteState[onCompleteState()]
    
    idle_state --> postFlags_idle[postFlags(idleflags)]
    idle_state --> postFlags_inactive2[postFlags(inactiveflags)]
    idle_state --> onIdleState[onIdleState()]
    idle_state --> onRunToIdleState[onRunToIdleState()]
    idle_state --> onInactiveState[onInactiveState()]
    
    running_state --> postFlags_run[postFlags(runflags)]
    running_state --> onRunState[onRunState()]
    running_state --> onIdleToRunState[onIdleToRunState()]
    
    onRunState -- No function produced --> postFlags_inactive3[postFlags(inactiveflags)]
    onRunState -- IVP Function produced --> postFlags_active[postFlags(activeflags)]
    onRunState -- IVP Function produced --> ActiveState[Active state]
    
    onInactiveState --> onInactiveState2[onInactiveState()]
    
    style checkUpdates stroke-dasharray: 5 5
    style isCompleted stroke-dasharray: 5 5
    style isRunnable stroke-dasharray: 5 5
    style postFlags_end fill:#ccc
    style postFlags_inactive fill:#ccc
    style onCompleteState fill:#008000
    style postFlags_idle fill:#ccc
    style postFlags_inactive2 fill:#ccc
    style onIdleState fill:#008000
    style onRunToIdleState fill:#008000
    style onInactiveState fill:#008000
    style postFlags_run fill:#ccc
    style onRunState fill:#008000
    style onIdleToRunState fill:#008000
    style postFlags_inactive3 fill:#ccc
    style postFlags_active fill:#ccc
    style onInactiveState2 fill:#008000
    style ActiveState fill:#ffff00
    
```

Key to Helm-Invoked Functions


- Immutable
- Overloaded
- Arguments from mission configuration

onRunState()	{ }
OnIdleState()	{ }
onCompleteState()	{ }
onIdleToRunState()	{ }
onRunToIdleState()	{ }
onInactiveState()	{ }

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

25



Helm-Invoked Behavior Functions

Chain of Contingency Events



```

graph TD
    checkUpdates[checkUpdates()] --> isCompleted[isCompleted()]
    isCompleted -- true --> completed_state[completed state]
    isCompleted -- false --> isRunnable[isRunnable()]
    isRunnable -- true --> running_state[Running state]
    isRunnable -- false --> idle_state[Idle state]
    
    completed_state --> postFlags_end[postFlags(endflags)]
    completed_state --> postFlags_inactive[postFlags(inactiveflags)]
    completed_state --> onCompleteState[onCompleteState()]
    
    idle_state --> postFlags_idle[postFlags(idleflags)]
    idle_state --> postFlags_inactive2[postFlags(inactiveflags)]
    idle_state --> onIdleState[onIdleState()]
    idle_state --> onRunToIdleState[onRunToIdleState()]
    idle_state --> onInactiveState[onInactiveState()]
    
    running_state --> postFlags_run[postFlags(runflags)]
    running_state --> onRunState[onRunState()]
    running_state --> onIdleToRunState[onIdleToRunState()]
    
    onRunState -- No function produced --> postFlags_inactive3[postFlags(inactiveflags)]
    onRunState -- IVP Function produced --> postFlags_active[postFlags(activeflags)]
    onRunState -- IVP Function produced --> ActiveState[Active state]
    
    onInactiveState --> onInactiveState2[onInactiveState()]
    
    style checkUpdates stroke-dasharray: 5 5
    style isCompleted stroke-dasharray: 5 5
    style isRunnable stroke-dasharray: 5 5
    style postFlags_end fill:#ccc
    style postFlags_inactive fill:#ccc
    style onCompleteState fill:#008000
    style postFlags_idle fill:#ccc
    style postFlags_inactive2 fill:#ccc
    style onIdleState fill:#008000
    style onRunToIdleState fill:#008000
    style onInactiveState fill:#008000
    style postFlags_run fill:#ccc
    style onRunState fill:#008000
    style onIdleToRunState fill:#008000
    style postFlags_inactive3 fill:#ccc
    style postFlags_active fill:#ccc
    style onInactiveState2 fill:#008000
    style ActiveState fill:#ffff00
    
```


Key to Helm-Invoked Functions

- Immutable
- Overloaded
- Arguments from mission configuration


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

26



Helm-Invoked **Overloadable** Behavior Functions



Helm Invoked: Because the helm calls them automatically when appropriate.
Overloadable: Because by default they are No-Op functions. Behavior authors get to write an alternative version as they see fit.

- `onRunState()` The primary guts of the behavior. This is where an objective function may be generated to influence the vehicle trajectory.
- `onIdleState()` Executed when the behavior has NOT met its run conditions. It may continue to update local behavior state, and perhaps make postings to the MOOSDB.
- `onCompleteState()` Executed when the behavior completes. It may augment any postings made with **end flags**.
- `onRunToIdleState()` Executed once upon entering the idle state from the run state
- `onIdleToRunState()` Executed once upon entering the run state from the idle state
- `onInactiveState()` Executed whenever the behavior *does not* produce an objective function.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

27



Helm-Invoked Overloadable Behavior Functions (not State-Based)



Helm Invoked: Because the helm calls them automatically when appropriate.
Overloadable: Because by default they are No-Op functions. Behavior authors get to write an alternative version as they see fit.

- `setParam()` The function where parameter handling is implemented. Invoked at helm startup, and when any dynamic configurations via the **updates** variable have been received.
- `onSetParamComplete()` Called once after handling parameter initialization or updates. Allows for any initialization work to be done while ensuring all parameters have been set.
- `onHelmStart()` Invoked at the helm start, even if the behavior is a template with no initial instance.
- `postConfigStatus()` Optionally post information to the MOOSDB summarizing how the behavior is configured.
- `onRunStatePrior()` A capability available new in release 17.7. It allows the behavior author to determine if `onRunState()` needs to be invoked, perhaps because the objective function it would produce is expected to be nearly identical to the previously returned objective function. This function returns a Boolean. If it returns false, then the helm re-uses the objective function from the previous iteration.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering


Michael Benjamin ©2026

28



Alpha Mission Example

In-Mission Reverse with `updates`

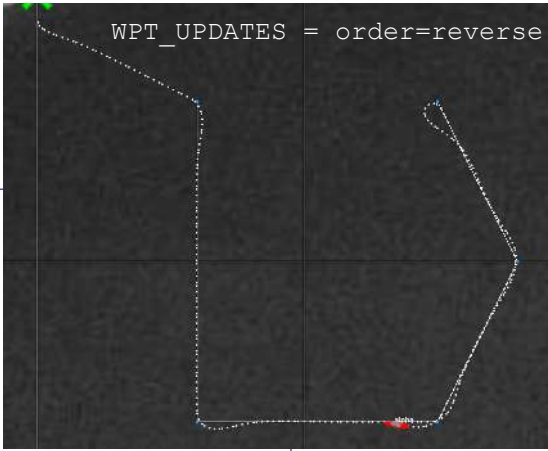


- After traversing the waypoints once, the `cycleflag` is published
- The `cycleflag` publishes to the `updates` variable, reversing the pattern direction for the second cycle.

```

name      = waypoint_survey
priority  = 100
condition = RETURN=false
condition = DEPLOY=true
endflag   = RETURN=true
speed     = 4.0
cycleflag = WPT_UPDATES=order=reverse
updates   = WPT_UPDATES
polygon   = 60,-40 : 60,-160 : 150,-160 : 180,100 : 150,-40

```



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

31



User Invoked Behavior Functions



User Invoked Behavior Functions: Utility functions invoked by the behavior authors for:

Accessing the Info Buffer

- `getBufferDoubleVal(string moos_var)`
- `getBufferStringVal(string moos_var)`
- `getBufferCurrTime()`
- `getBufferLocalTime()`
- `getBufferMsgTimeVal(string moos_var)`
- `getBufferDoubleVector(string moos_var)`
- `getBufferStringVector(string moos_var)`

Other Functions

- `addInfoVar(string moos_var)`
- `setComplete()`

Posting to the MOOSDB

- `postMessage(string moosvar, double val, string key="")`
- `postMessage(string moosvar, string val, string key="")`
- `postMessage(string moosvar, bool val, string key="")`
- `postMessage(string moosvar, int val, string key="")`
- `postRepeatableMessage(string moos_var, double val)`
- `postRepeatableMessage(string moos_var, double val)`
- `postEMessage(string val)`
- `postWMessage(string val)`

For further help/descriptions:
http://oceanai.mit.edu/ivpman/help/ivp_bhv_utils

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

32






Accessing the Info Buffer

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

33

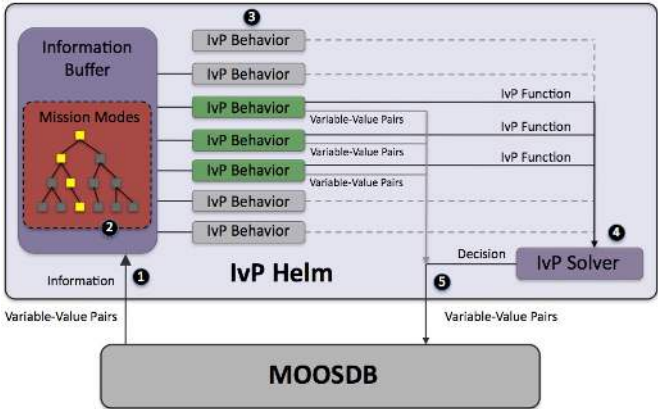



Interval Programming and the IvP Helm

- 1 Mail is read in the MOOS OnNewMail() function and applied to a local buffer.
- 2 The helm mode is determined and set of running behaviors determined.
- 3 Behaviors do their thing – posting MOOS variables and an IvP function.
- 4 Competing behaviors are resolved with the IvP solver.
- 5 The Helm decision and any behavior postings are published to the MOOSDB.

The Information Buffer:


- Updated automatically by the helm from latest MOOSDB postings.
- Behaviors then update locally from the information buffer.




Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

34



The Helm Information Buffer



Available Behavior Functions

Two primary functions for getting the latest variable value:

```
double getBufferDoubleVal(string varname, bool& result)
```

```
string getBufferStringVal(string varname, bool& result)
```

↓

Name of the MOOS Variable

↓


Returns **false** if nothing known about the given MOOS variable.

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

35



Behavior Functions for Accessing the Info Buffer



The primary function for getting the latest variable value of type **double**:

```
double getBufferDoubleVal(string moos_var)
```

↓

Name of the MOOS Variable

Note: if nothing is known about a numerical value, it returns zero.

The primary function for getting the latest variable value of type **string**:

```
string getBufferStringVal(string moos_var)
```

↓

Name of the MOOS Variable


Note: if nothing is known about a numerical value, it returns the empty string "".

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

36



Behavior Functions for Accessing the Info Buffer



Another function for getting the latest variable value of type **double**:

```
double getBufferDoubleVal(string moos_var, bool&)
```

↓
Name of the MOOS Variable

The Boolean will be set to false if nothing is known about the MOOS variable.

Another function for getting the latest variable value of type **string**:

```
string getBufferStringVal(string moos_var, bool&)
```

↓
Name of the MOOS Variable


The Boolean will be set to false if nothing is known about the MOOS variable.

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

37



The Helm Information Buffer



Example from Simple Waypoint Behavior

Example from the Simple Waypoint Behavior

```
double getBufferDoubleVal(string moos_var, bool&)
```

```

01 IvPFunction *BHV_SimpleWaypoint::onRunState()
02 {
03     bool ok1, ok2;
04     m_osx = getBufferDoubleVal("NAV_X", ok1);
05     m_osy = getBufferDoubleVal("NAV_Y", ok2);
06     if(!ok1 || !ok2) {
07         postWMessage("No ownship X/Y info in buffer.");
08         return(0);
09     }

```

Standard interface

Get X/Y info from the info buffer


Post warnings if nothing known about X/Y

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

38



Requesting the Helm to Populate the Information Buffer



A behavior makes a request to the helm to include a given variable in the helm `info_buffer`:

```
void addInfoVars(string varnames)
```

Example from the SimpleWaypoint behavior:

```
01 // Procedure: Constructor
02 BHV_SimpleWaypoint::BHV_SimpleWaypoint(IvPDomain domain)
03     :IvPBehavior(domain)
04 {
05     addInfoVars("NAV_X, NAV_Y");
06 }
```

Behavior constructor

Requesting NAV_X and NAV_Y position for this behavior

- Another instantiated behavior may have also registered for these variables, in which case the above is unnecessary.
- However, each behavior author should always request the components of the `info_buffer` needed by this behavior.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

39



The Helm Information Buffer



Requesting info to be buffered

When a behavior registers for a MOOS variable, it may optionally specify that it is ok if the variable is unknown to the info buffer when queried:

```
void addInfoVars(string varnames, string warning_hint)
```

For example:

```
addInfoVars("WPT_INDEX", "no_warning");
```

Without the `"no_warning"` argument, if `WPT_INDEX` has never been written to the MOOSDB when a `getBufferDoubleVal("WPT_INDEX")` is invoked, the helm will automatically post a run warning in the appcasting output, and post to the MOOSDB the variable `BHV_WARNING`, indicating:

```
WPT_INDEX info not found in helm info_buffer
```

For some variables, that's perfectly normal, so we'd like to avoid having a distracting warning.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering


Michael Benjamin ©2026

40



The Helm Information Buffer

Recent History of changes



Two functions for getting the latest history variable values:

```
vector<double> getBufferDoubleVector(string varname, bool& result)
```

```
Vector<string> getBufferStringVector(string varname, bool& result)
```

- Each returns a vector of values posted to the information buffer since the last helm iteration.
- The information buffer does not otherwise keep a history of variable changes, just the set of changes occurring on a single iteration.
- The Boolean `result` will be false only if the info buffer has *never* had any posts to the given varname. It's quite possible that the result is true while returning an empty vector.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example

Lab Preview


Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

41



The Helm Information Buffer

Time Information



The information buffer also stores information about time. The following two functions are defined by the `IvPBehavior` superclass and may be called:

```
double IvPBehavior::getBufferCurrTime() (UTC time, number of seconds since January 1, 1970)
```

```
double IvPBehavior::getBufferLocalTime() (Time since the helm was launched)
```

The user can also query the info buffer to determine how long it has been since a variable has last been updated (by the helm via received mail):

```
double IvPBehavior::getBufferTimeVal(string moos_var)
```

The returned value should be exactly zero if this variable was updated by new mail received by the helm on the current iteration. If the variable name is not found in the buffer, the return value is -1.

(This function may be exploited by the behavior to opt NOT to perform an action if certain sensor information has not changed.)

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer



Posting Messages

A Simple Example

Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

42






Posting Messages

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

43

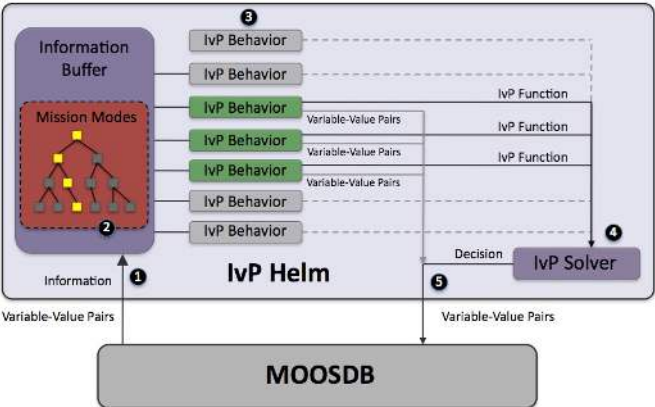



The IvP Helm Iterate Loop Posting Messages

- 1 Mail is read in the MOOS OnNewMail() function and applied to a local buffer.
- 2 The helm mode is determined and set of running behaviors determined.
- 3 Behaviors do their thing – posting MOOS variables and an IvP function.
- 4 Competing behaviors are resolved with the IvP solver.
- 5 The Helm decision and any behavior postings are published to the MOOSDB.

The Information Buffer:

- Updated automatically by the helm from latest MOOSDB postings.
- Behaviors then update locally from the information buffer.




The diagram illustrates the IvP Helm Iterate Loop. At the bottom is the MOOSDB. An arrow labeled '1' points from MOOSDB to the Information Buffer. Inside the Information Buffer is a tree structure labeled 'Mission Modes' with an arrow labeled '2' pointing to it. Above the Information Buffer are several 'IvP Behavior' boxes. Arrows labeled '3' point from these behaviors to 'IvP Function' boxes. Arrows labeled '4' point from the IvP Functions to the 'IvP Solver'. The IvP Solver outputs a 'Decision' (arrow labeled '4') and 'Variable-Value Pairs' (arrow labeled '5') back to the MOOSDB. The Information Buffer also outputs 'Variable-Value Pairs' (arrow labeled '5') to the MOOSDB.


Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

44



Posting Messages from a Behavior



Behaviors produce two forms of output:

1. IvP objective functions
2. Postings to the MOOSDB in the form of variable-value pairs.

- The helm collects all postings produced by behaviors, and publishes them on their behalf at the end of each helm iteration.
- The helm fills in the behavior name, into the “auxilliary source” field, into each posted message.
- The two key functions are:

```
void postMessage(string varname, string value)
```

```
void postMessage(string varname, double value)
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

45



The Helm Duplication Filter



Recall the Helm utilizes a *duplication filter*, by default blocking successive messages with the same numerical or string value.

Behavior 1:	<code>postMessage("TEMPERATURE", 32);</code>	}	Results in two postings
Behavior 2:	<code>postMessage("TEMPERATURE", 99);</code>		
Behavior 1:	<code>postMessage("TEMPERATURE", 32);</code>	}	Results in one posting
Behavior 2:	<code>postMessage("TEMPERATURE", 32);</code>		

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

46



Overriding the Helm Duplication Filter



The filter may be overridden by posting with a key:

```
void postMessage(string varname, string value, string key="")
```

```
void postMessage(string varname, double value, string key="")
```


Behavior 1:	<code>postMessage("TEMPERATURE", 32);</code>	}	Results in one postings
Behavior 2:	<code>postMessage("TEMPERATURE", 32);</code>		
Behavior 1:	<code>postMessage("TEMPERATURE", 32, "one");</code>	}	Results in two postings
Behavior 2:	<code>postMessage("TEMPERATURE", 32, "two");</code>		

↑
Unique keys

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

47



Overriding the Helm Duplication Filter



The filter may be overridden by explicitly indicating that all posts are to be made

```
void postRepeatableMessage(string varname, string value)
```

```
void postRepeatableMessage(string varname, double value)
```

↑
Note the "Repeatable"

Behavior 1:	<code>postMessage("TEMPERATURE", 32);</code>	}	Results in one postings
Behavior 2:	<code>postMessage("TEMPERATURE", 32);</code>		
Behavior 1:	<code>postRepeatableMessage("TEMPERATURE", 32);</code>	}	Results in two postings
Behavior 2:	<code>postRepeatableMessage("TEMPERATURE", 32);</code>		

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer


Posting Messages

A Simple Example


Lab Preview

Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

48



Further Convenience Methods for Posting to the MOOSDB



As a convenience functions, numerical postings may be posted, rounded to the closest integer:

```
void postIntMessage(string varname, double value, string key="")
```

The Idea is to have more successive equivalent posts to allow the duplication filter to be more effective.

```
postIntMessage("TEMPERATURE", 32.1);
postIntMessage("TEMPERATURE", 32.2);
postIntMessage("TEMPERATURE", 32.4);
postIntMessage("TEMPERATURE", 32.6);
postIntMessage("TEMPERATURE", 32.7);
postIntMessage("TEMPERATURE", 32.9);
postIntMessage("TEMPERATURE", 33.1);
postIntMessage("TEMPERATURE", 33.4);
postIntMessage("TEMPERATURE", 33.7);
postIntMessage("TEMPERATURE", 34.1);
```

Results in one posting with value 32.00000

Results in one posting with value 33.00000

Results in one posting with value 34.00000

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

49



Further Convenience Methods for Posting to the MOOSDB



Three additional methods are available as convenience functions:

- The first will take a Boolean value and post the appropriate string to the given MOOS variable. Like other posting methods, it also allows for specifying an optional key.

```
void postBoolMessage(string moos_var, bool value, string key="")
```

- To post a message to the special MOOS variable `BHV_WARNING`, the following method is used. *It will not be subjected to the helm duplication filter.*

```
void postWMessage(string message)
```

- To post a message to the special MOOS variable `BHV_ERROR`, the following method is used. *It will not be subjected to the helm duplication filter.*

```
void postEMessage(string message)
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

50






A Simple Example

Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

51

The BHV_SimpleWaypoint Behavior

- The `BHV_SimpleWaypoint` behavior is in your `moos-ivp-extend` tree.
- It is a very simple version of the `BHV_Waypoint` behavior in the `moos-ivp` tree.
- It is configured with single waypoint, capture radius and speed:

```

Behavior = BHV_SimpleWaypoint
{
  pwt = 100
  condition = RETURN=true
  endflag   = RETURN=false


  speed = 2.0 // meters per second
  radius = 8.0
  ptx = 60
  pty = -40
}

```


Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

52




The BHV_SimpleWaypoint Behavior



- The `BHV_SimpleWaypoint` behavior is in your `moos-ivp-extend` tree.
- It is a very simple version of the `BHV_Waypoint` behavior in the `moos-ivp` tree.
- It is configured with single waypoint, capture radius and speed:

An example mission is in `moos-ivp-extend/missions/alder:`



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

53



Behavior Parameter Setting



The `setParam()` function

Each behavior may overload a function for setting parameters:

```
bool setParam(string param, string value);
```

Note: if this function returns false for any one parameter, the helm will launch into a “MALCONFIG” mode. (try it!)

```
Behavior = BHV_SimpleWaypoint
{
  pwt = 100
  condition = RETURN=true
  endflag = RETURN=false
  speed = 2.0 // meters per second
  radius = 8.0
  ptx = 60
  pty = -40
}
```

Note: The parameter is first sent to the `IvPBehavior` superclass for handling. If it is unknown to the superclass, it is sent to the subclass handler.

SO: You may not override parameters handled at the superclass level.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

54



The BHV_SimpleWaypoint Behavior



The Constructor

The Constructor:

```

01 // Procedure: Constructor
02 BHV_SimpleWaypoint::BHV_SimpleWaypoint(IvPDomain domain) :
03     IvPBehavior(domain)
04 {
05     addInfoVars("NAV_X, NAV_Y");
06 }
```

Behavior constructor

Requesting NAV_X and NAV_Y position for this behavior

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

55



The BHV_SimpleWaypoint Behavior



The setParam() Function

The setParam() function:

```

bool BHV_SimpleWaypoint::setParam(string param, string val)
{
    double double_val = atof(val.c_str());
    if(param == "ptx")
        m_nextpt.set_vx(double_val);
    else if(param == "pty")
        m_nextpt.set_vy(double_val);
    else if(param == "speed")
        m_desired_speed = double_val;
    else if(param == "radius")
        m_arrival_radius = double_val;
    else
        return(false);
    return(true);
}
```

```

Behavior = BHV_SimpleWaypoint
{
    pwt = 100
    condition = RETURN=true
    endflag = RETURN=false

    speed = 2.0
    radius = 8.0
    ptx = 60
    pty = -40
}
```

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering


Michael Benjamin ©2026

56



The BHV_SimpleWaypoint Behavior

The `onIdleState()` function



```

01 void BHV_SimpleWaypoint::onIdleState ()
02 {
03     postViewPoint(false);
04 }

```

When idle, this behavior will “erase” its waypoint

```

01 void BHV_SimpleWaypoint::postViewPoint(bool viewable)
02 {
03     m_nextpt.set_label(m_us_name + "'s next waypoint");
04     m_nextpt.set_type("waypoint");
05     m_nextpt.set_source(m_descriptor);
06
07     string point_spec;
08     if(viewable)
09         point_spec = m_nextpt.get_spec("active=true");
10     else
11         point_spec = m_nextpt.get_spec("active=false");
12     postMessage("VIEW_POINT", point_spec);
13 }

```

This one utility function will either generate a message to draw the waypoint, or generate a message to erase the waypoint.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering


Michael Benjamin ©2026

57



The BHV_SimpleWaypoint Behavior

The `onRunState()` function



The `onRunState()` function

```

01 IvPFunction *BHV_SimpleWaypoint::onRunState ()
02 {
03     bool ok1, ok2;
04     m_osx = getBufferDoubleVal("NAV_X", ok1);
05     m_osy = getBufferDoubleVal("NAV_Y", ok2);
06     if(!ok1 || !ok2) {
07         postWMessage("No ownship X/Y info in info_buffer.");
08         return(0);
09     }
10
11     double dist = hypot((m_nextpt.x()-m_osx), (m_nextpt.y()-m_osy));
12     if(dist <= m_arrival_radius) {
13         setComplete();
14         postViewPoint(false);
15         return(0);
16     }
17
18     // Part 3: Build an objective function: next class!!!
19     return(ipf);
20 }

```

Get needed info from the info_buffer

Determine if we have reached our waypoint. If so, we're done

Otherwise generate an objective function to reach our waypoint.

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

58



The BHV_SimpleWaypoint Behavior



The `onRunState()` function

The `onRunState()` function

```

01 IvPFunction *BHV_SimpleWaypoint::onRunState ()
02 {
03     bool ok1, ok2;
04     m_osx = getBufferDoubleVal("NAV_X", ok1);
05     m_osy = getBufferDoubleVal("NAV_Y", ok2);
06     if(!ok1 || !ok2) {
07         postWMessage("No ownship X/Y info in info_buffer.");
08         return(0);
09     }
10
11     double dist = hypot((m_nextpt.x()-m_osx), (m_nextpt.y()-m_osy));
12     if(dist <= m_arrival_radius) {
13         setComplete(); ←
14         postViewPoint(false);
15         return(0);
16     }
17
18     // Part 3: Build an objective function: next class!!!
19     return(ipf);
20 }

```

Get needed info from the info_buffer

Determine if we have reached our waypoint. If so, we're done


Otherwise generate an objective function to reach our waypoint.

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

59



The setComplete() Function



setComplete ()



- The `setComplete()` function may be invoked by from within any behavior to indicate that the job of the behavior is done.
- It is highly specific to a particular behavior, e.g, a waypoint behavior hitting all its waypoints.
- It will cause any specified *end flags* to be posted.
- Unless configured with `perpetual=true`, the behavior will be destroyed on the next iteration.

Motivation Overview
Adding a New Behavior
Behavior Functions
Accessing the Info Buffer
Posting Messages
A Simple Example
Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

60

Lab Preview

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

61

Lab 10 Introduction to Behavior Writing

- Generate of a new behavior class
- Add a behavior to the build system
- Obtain information from the information buffer into your behavior
- Overload general behavior functions to suit your needs
- Handle the setting of behavior parameters set by the user

<https://vimeo.com/91971985>

Alpha Range Pulse

MIT 2.680

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages


A Simple Example

Lab Preview


MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

62




Lab 10 Introduction to Behavior Writing



- Modify / Extend the first behavior to produce a zig-zag leg each time it arrives at a waypoint.
- Learn how to use basic functionality of the IvPBuild Toolbox.

<https://vimeo.com/91975349>



Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages



A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

63

THE END

Motivation Overview

Adding a New Behavior

Behavior Functions

Accessing the Info Buffer

Posting Messages

A Simple Example

Lab Preview

MIT Dept of Mechanical Engineering

Michael Benjamin ©2026

64