pBasicContactMgr: Managing Platform Contacts

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
`project-pavlab/appdocs/app_pcmanager`

1 Overview

The `pBasicContactMgr` application deals with information about other known vehicles in its vicinity. It is not a sensor application, but rather handles incoming "contact reports" which may represent information received by the vehicle over a communications link, or may be the result of on-board sensor processing. By default the `pBasicContactMgr` posts to the MOOSDB summary reports about known contacts, but it also may be configured to post alerts, i.e., MOOS variables, with select content about one or more of the contacts.

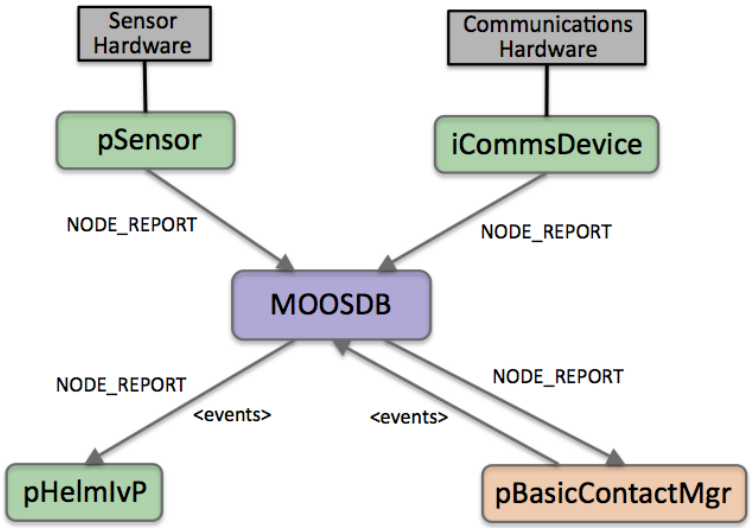

Figure 1: **The pBasicContactMgr Application:** The `pBasicContactMgr` utility receives `NODE_REPORT` information from other MOOS applications and manages a list of unique contact records. It may post additional user-configurable alerts to the MOOSDB based on the contact information and user-configurable conditions. The source of contact information may be external (via communications) or internal (via on-board sensor processing). The pSensor and iCommsDevice modules shown here are fictional applications meant to convey these two sources of information abstractly.

The `pBasicContactMgr` application is partly designed with simultaneous usage of the IvP Helm in mind. The alerts posted by `pBasicContactMgr` may be configured to trigger the dynamic spawning of behaviors in the helm, such as collision-avoidance behaviors. The `pBasicContactMgr` application does not perform sensor fusion, and does not reason about or post information regarding the confidence it has in the reported contact position relative to ground truth. These may be features added in the future, or perhaps may be features of an alternative contact manager application developed by a third party source.

2 Using pBasicContactMgr

The operation of `pBasicContactMgr` consists of posting user-configured alerts, and the posting of several MOOS variables, the `CONTACTS_*` variables, indicating the status of the contact manager.

2.1 Contact Alert Configuration

Alert messages are used to alert other MOOS applications when a contact has been detected within a certain range of ownship. Multiple alert types may be configured, each keyed on the *alert id*. A single alert type may be defined over several lines, where each line contains the alert id of the alert type being configured. Alerts are configured in the mission file with the `alert` parameter as follows:

```
alert = id=<alert-id>, var=<MOOSVar>,              pattern=<string>
alert = id=<alert-id>, alert_range=<distance>,     cpa_range=<distance>
alert = id=<alert-id>, alert_range_color=<color>, cpa_range_color=<color>
```

The var=<MOOSVar> component indicates the MOOS variable posted for the given alert. The pattern=<string> component may be any string with any, none, or all of the following macros available for expansion:

$[VNAME]:	The name of the contact.
$[X]:	The position of the contact in local x coordinates.
$[Y]:	The position of the contact in local y coordinates.
$[LAT]:	The latitude position of the contact in earth coordinates.
$[LON]:	The longitude position of the contact in earth coordinates.
$[HDG]:	The reported heading of the contact.
$[SPD]:	The reported speed of the contact.
$[DEP]:	The reported depth of the contact.
$[VTYPE]:	The reported vessel type of the contact.
$[UTIME]:	The UTC time of the last report for the contact.

If the right-hand side of the pattern=<string> component contains its own parsing separators, it is recommended that the entire <alert-pattern> string is put within double quotes to ensure proper parsing, as in Line 12 in Listing 5.

The alert_range=<distance> component represents a threshold range to a contact, in meters. When a contact moves within this range, an alert will be generated. If this distance is left unspecified, a default value will be used. The default value for all alert types is 1000 meters. This fallback default alert range may be changed with the configuration parameter default_alert_range.

The cpa_range=<distance> component also represents a threshold range, in meters. Typically this cpa_range is greater than the alert_range parameter. When a contact is noted to be within the cpa_range, the contact and ownship trajectories are considered, to calculate the closest point of approach (CPA). If the CPA range is determined to be within the alert_range, an alert is generated, even if the present range between ownship and contact is outside the alert_range.

The alert_range_color=<color> component indicates a desired color for rendering the alert_range circle. Rendering is done by pBasicContactMgr by posting to the variable VIEW_CIRCLE, typically handled by pMarineViewer. The default value is "gray70". The cpa_range_color=<color> component similarly indicates a desired color for rendering the cpa_range circle. The default value is "gray30". See the Colors Appendix for more on colors.

The posting of rendering circles by pBasicContactMgr may be disabled in one of three ways. First, they may be disabled outright for all alerts all the time by setting display_radii=false in the MOOS configuration block. Second, they may be turned off by posting BCM_DISPLAY_RADII=false to the MOOSDB at any time. This hook may be configured into a button or action-pull-down menu item in pMarineViewer for example. This will override any static setting in the MOOS configuration block. The third method for disabling the circles is to specify the color "invisible" for any one of the alert ranges. This value is interpreted at least in pMarineViewer as in indication that it is not to be drawn. The latter method provides a finer-grained control of rendering some circles but not others.

See lines 10-13 in Listing 5 for an example alert configuration.

2.2 Dynamic Contact Alert Configuration

Alert configuration may also be handled dynamically via an incoming MOOS variable, BCM_ALERT_REQUEST. This has the same format used in the alert configuration lines described above. For example, the pBasicContactMgr configuration lines:

Listing 2.1: Configuration snippet for pBasicContactMgr.

```
1  //---------------------------------------------------------
2  // pBasicContactMgr Configuration Block3
4
5  ProcessConfig = pBasicContactMgr
6  {
7    . . .
8    alert = id=avd, var=CONTACT_INFO, val="name=$[VNAME] # contact=$[VNAME]"
9    alert = id=avd, alert_range=80, alert_range_color=green
10   . . .
11 }
```

could be achieved with the following posting to the MOOSDB:

```
BCM_ALERT_REQUEST = id=avd, var=CONTACT_INFO, val="name=$[VNAME] # contact=$[VNAME]",
                    alert_range=80, alert_range_color=green
```

Who makes this posting and why is this preferable? The answer to the first question is the client, or the intended consumer of the alert. The reason why this is preferable is a bit more subtle. The short answer is that the client knows best. Limiting the alert configuration to the client can simplify mission configuration and reduce unintended mis-configurations.

Consider the case, for example, with the CollisionAvoidance behavior. This behavior is typically configured to complete, and die, when the behavior contact exceeds a certain range. This range is set by the user in the behavior parameter completed_dist. A situation to avoid would be the case where the behavior completes but then is immediately spawned again due to a subsequent immediate alert from the contact manager. This would occur, for example, if the behavior set its completed_dist to 100, but the contact manager was configured as it is in Listing 1 above, with the alert_range set to 80.

Without dynamic alert configurations in pBasicContactMgr, the configuration of the AvoidCollision behavior would need to be explicitly coordinated in two places, in the behavior configuration file and the mission configuration file. For example, one would need to make sure that line 8 in Listing 2 below reflected a completed distance *less than* the alert distance specified in line 9 in Listing 1 above.

Listing 2.2: Configuration snippet for AvoidCollision behavior.

```
1  //------------------------------------------
2  Behavior = BHV_AvoidCollision
3  {
```

```
  4    . . .
  5    updates        = CONTACT_INFO
  6    endflag        = CONTACT_RESOLVED = $[CONTACT]
  7    templating     = spawn
  8    completed_dist = 100
  9    . . .
 10  }
```

From the perspective of the contact manager, it simply supports both styles of alert configurations, and the issue of how or whether a client chooses to request alerts is of no concern to the contact manager, or those who configure it. The issue of how alert requests are generated for the AvoidCollision behavior is discussed separately in the section on this behavior.

2.3 Contact Alert Triggers

Alerts are triggered for all contacts based on range between ownship and the reported contact position. It is assumed that each incoming contact report minimally contains the contact's name and present position. An alert will be triggered if the current range to the contact falls within the distance given by `alert_range`, as in Contact-A in Figure 2.

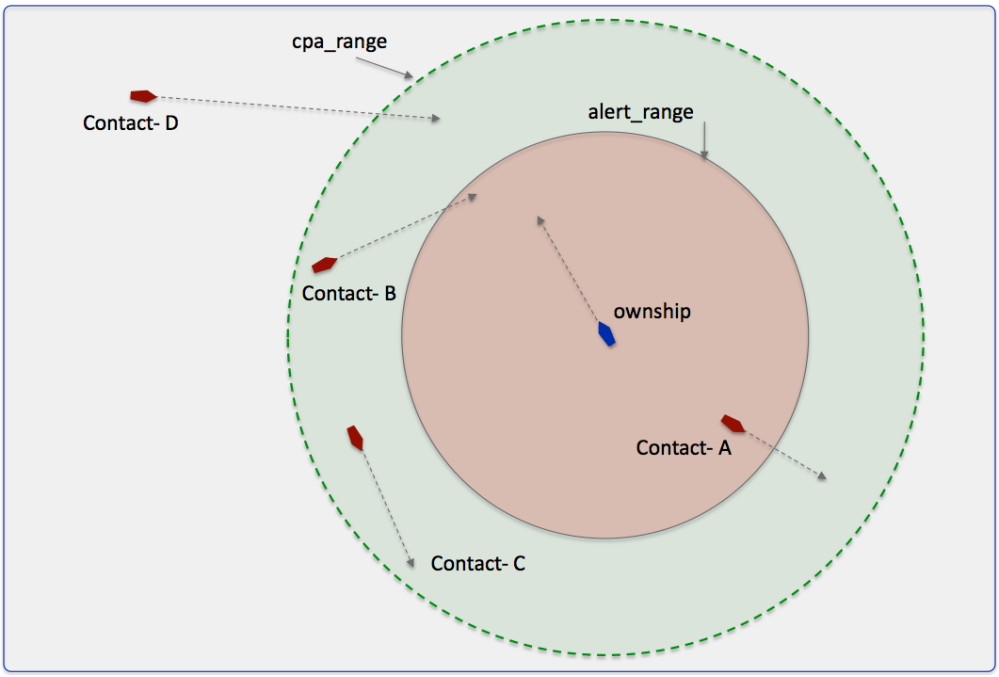

Figure 2: **Alert Triggers in pBasicContactMgr:** An alert may be triggered by pBasicContactMgr if the contact is within the `alert_range`, as with Contact-A. It may also be triggered if the contact is within the `cpa_range`, and the contact's CPA distance is within the `alert_range`, as with Contact-B. Contact-C shown here would not trigger an alert since its CPA distance is its current range and is not within the `alert_range`. Contact-D also would not trigger an alert despite the fact that its CPA with ownship is apparently small, since its current absolute range is greater than `cpa_range`.

The contact manager may also be configured with a second trigger criteria consisting of another range to contact. The cpa_range may be set individually for a given user defined alert, but also has a default value which may be set in the configuration file:

```
alert_cpa_range = <distance>
```

The cpa_range is typically larger than the alert_range. (Its influence is effectively disabled when or if it is set to be equal to or less than the alert_range.) When a contact is outside the alert_range, but within the cpa_range, as with Contact-B in Figure 2, the closest point of approach (CPA) between the contact and ownship is calculated given their presently-known position and trajectories. If the CPA distance falls below the alert_range value, an alert is triggered.

2.4 Contact Alert Record Keeping

The contact manager keeps a record of all known contacts for which it has received a report. This list is posted in the MOOS variable CONTACTS_LIST, in a comma-separated string such as:

```
CONTACTS_LIST = "delta,gus,charlie,henry"
```

Once an alert is generated for a contact it is put on the *alerted* list and this subset of all contacts is posted in the MOOS variable CONTACTS_ALERTED. Each entry in the list names a vehicle and alert id separated by a comma, such as:

```
CONTACTS_ALERTED = "(delta,avd)(charlie,avd)"
```

Likewise, those contacts for which no alert has been generated are in the *unalerted* list and this is reflected in the MOOS variable CONTACTS_UNALERTED. Again, each entry is comprised of both a vehicle name and alert id separated by a comma.

```
CONTACTS_UNALERTED = "(gus,avd)(henry,avd)"
```

Contact records are not maintained indefinitely and eventually are *retired* from the records after some period of time during which no new reports are received for that contact. The period of time is given by the contact_max_age configuration parameter. The list of retired contacts is posted in the MOOS variable CONTACTS_RETIRED:

```
CONTACTS_RETIRED = "bravo,foxtrot,kilroy"
```

A contact recap of all non-retired contacts is also posted in the MOOS variable CONTACTS_RECAP:

```
CONTACTS_RECAP = "name=ike,age=11.3,range=193.1 # name=gus,age=0.7,range=48.2 # \
                  name=charlie,age=1.9,range=73.1 # name=henry,age=4.0,range=18.2"
```

Note: Each of these five MOOS variables is published only when its contents differ from its previous posting.

2.5 Guarding Against Unbounded Memory Growth

In some systems it may be possible for the sensors or sensor system to be generating many contacts, with unique IDs, over time. This problem may arise simply from long duration (days+) missions in

high contact density situations. It may also be partly a result of a sensor that easily loses track on a contact, generating several IDs for the same contact as it approaches and ultimately departs the ownship vicinity.

The contact manager needs to truly forget about a contact after some period of time. When a contact manager has stopped receiving reports on a contact after a time interval of contact_max_age seconds, the contact is moved onto the retired list, reported by the variable CONTACTS_RETIRED. However, the contact manager has not yet forgotten about this contact. After an additional period of contact_max_age_history, the contact manager will completely discard all memory of this contact, and it will no longer appear even on the CONTACTS_RETIRED list. The default for this parameter is 60 seconds. It may be set to zero seconds, essentially forgetting the contact as soon as it is retired. And it may be set no higher than 600 seconds. The hard-coded upper limit is ensure a user does not unintentionlly enable possible unbounded growth.

2.6 Contact Resolution

An alert is generated by the contact manager for a given contact *once*, when the alert trigger criteria is first met. In the iteration when the criteria is met, the contact is moved from the *unalerted* list to the *alerted* list, the alert is posted to the MOOSDB, and no further alerts are posted despite any future calculations of the trigger criteria. One exception to this is when the pBasicContactMgr receives notice that a contact has been resolved, through the MOOS variable CONTACT_RESOLVED. When a contact is resolved, it is moved from the alerted list back on to the un-alerted list.

3 Deferring to Earth Coordinates over Local Coordinates

Incoming node reports contain the position information of the contact and may be specified in either local x-y coordinates, or earth latitude longitude coordinates or both. By default pBasicContactMgr uses the local coordinates for calculations and the earth coordinates are merely redundant. It may instead be configured, with the contact_local_coords parameter, to have its local coordinates set from the earth coordinates if the local coordinates are missing:

```
contact_local_coords = lazy_lat_lon
```

It may also be configured to always use the earth coordinates, even if the local coordinates are set:

```
contact_local_coords = force_lat_lon
```

The default setting is `verbatim`, meaning no action is taken to convert coordinates. If either of the other two above settings are used, the latitude and longitude coordinates of the local datum, or `(0,0)` point must be specified in the MOOS mission file, with LatOrigin and LongOrigin configuration parameters. (They are typically present in all mission files anyway.)

4 Usage of the pBasicContactMgr with the IvP Helm

The IvP helm may used in conjunction with the contact manager to coordinate the dynamic spawning of certain helm behaviors where the instance of the behavior is dedicated to a helm

objective associated with a particular contact. For example, a collision avoidance behavior, or a behavior for maintaining a relative position to a contact for achieving sensing objectives, would be examples of such behaviors. One may want to arrange for a new behavior to be spawned as the contact becomes known. The helm needs a cue in the form of a MOOS variable posting to trigger a new behavior spawning, and this is easily arranged with the alerts in the pBasicContactMgr.

On the flip-side of a new behavior spawning, a behavior may eventually declare itself completed and remove itself from the helm. The conditions leading to completion are defined within the behavior implementation and configuration. No cues external to the helm are required to make that happen. However, once an alert has been generated by the contact manager for a particular contact, it is not generated again, unless it receives a message that the contact has been resolved. Therefore, if the helm wishes to received future alerts related to a contact for which it has received an alert in the past, it must declare the contact *resolved* to the contact manager as discussed in Section 2.6. This would be important, for example, in the following scenario: (a) a collision avoidance behavior is spawned for a new contact that has come within range, (b) the behavior completes and is removed from the helm, presumably because the contact has slipped safely out of range, (c) the contact or ownship turns such that a collision avoidance behavior is once again needed for the same contact.

An example mission is available for showing the use of the contact manager and its coordination with the helm to spawn behaviors for collision avoidance. This mission is `m2_berta` and is described in the IvP Helm documentation. In this mission two vehicles are configured to repeatedly go in and out of collision avoidance range, and the contact manager repeatedly posts alerts that result in the spawning of a collision avoidance behavior in the helm. Each time the vehicle goes out of range, the behavior completes and dies off from the helm and is declared to the contact manager to be resolved.

5 Terminal and AppCast Output

The status of the contact manager may be monitored from from an open console window where pBasicContactMgr is launched. Example output is shown below in Listing 3.

Listing 5.3: Example pBasicContactMgr terminal and appcast output.

```
1  ============================================================================
2  pBasicContactMgr gilda                                          0/0 (379)
3  ============================================================================
4  Alert Configurations (1):
5  ---------------------
6  Alert ID = avd
7    VARNAME   = CONTACT_INFO
8    PATTERN   = name=$[VNAME]#contact=$[VNAME]
9    RANGE     = 40, green
10   CPA_RANGE = 45, invisible
11
12 Alert Status Summary:
13 ----------------------
14         List: henry
15     Alerted:
16   UnAlerted: (henry,avd)
17     Retired:
```

```
18        Recap: vname=henry,range=136.55,age=2.05
19
20  Contact Status Summary:
21  ----------------------
22  Contact     Range     Alerts     Alerts     Alerts
23                        Total      Active     Resolved
24  -------     -----     ------     ------     --------
25  henry       136.6     1          0          1
26
27
28  Recent Events (3):
29  [159.35]: Resolved: (henry,all_alerts)
30  [159.35]: TryResolve: (henry,all_alerts)
31  [104.53]: CONTACT_INFO=name=henry#contact=henry
```

On line 2, the "0/0" indicates there were no configuration warnings and no run-time warnings (thus far). The "(379)" represents the iteration counter of pBasicContactMgr. In lines 4-10, the alerts configured by the user in the MOOS configuration block are shown. If multiple alerts types are configured, they would each be listed here separated by their alert id.

In lines 12-18, the record-keeping status of the contact manager is output. These five lines are equivalent to the content of the CONTACTS_* variables described in Section 2.4. In lines 20-25, the status and alert history for each known contact is shown. Finally, in lines 28, a limited list of recent events is shown. Typically an event is either an alert generated or an alert resolved. The alert resolution is split into two events, the alert resolution attempt, and the actual resolution. This may help draw the user's attention if an alert is attempted but failed.

6 Configuration Parameters for pBasicContactMgr

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

Listing 6.4: Configuration Parameters for pBasicContactMgr.

alert:	A description of a single alert. Section 2.1.
contact_local_coords:	Determines if the local coordinates of incoming node reports are filled by translated latitude longitude coordinates. Legal values: verbatim, lazy_lat_lon, force_lat_lon. The default is verbatim, meaning no translation action is taken.
default_alert_range:	The range to a contact, in meters, within which an alert is posted. Legal values: any positive number. The default is 1000. Section 2.1.
default_cpa_range:	The range to a contact, in meters, within which an alert is posted if the closest point of approach (CPA) falls within this range. Legal values: any positive number. The default is 1000. Section 2.1.
default_alert_range_color:	The default color for rendering the alert range radius. Legal values: any color in Colors Appendix . The default is *gray70*. Section 2.1.

default_cpa_range_color: The default color for rendering the cpa range radius. Legal values: any color in the Colors Appendix. The default is *gray30*. Section 2.1.

contact_max_age: Seconds between reports before a contact is dropped from the list. Legal values: any non-negative number. The default is 600. Section 2.4.

contact_max_age_history: Seconds after a contact is retired before it is removed entirely from the contact manager memory, and thus even the posted retired lists. This is guard against unbounded memory growth. The default is 60. Section 2.5.

display_radii: If true, the two alert ranges are posted as viewable circles. Legal values: true, false. The default is false.

post_closest_range: If true, the range to the closest contact is posted, rounded to the nearest meter. The default is false (In the release after 17.7).

6.1 An Example MOOS Configuration Block

To see an example MOOS configuration block, enter the following from the command-line:

```
$ pBasicContactMgr --example or -e
```

This will show the output shown in Listing 5 below.

Listing 6.5: Example configuration of the pBasicContactMgr application.

```
1  ================================================================
2  pBasicContactMgr Example MOOS Configuration
3  ================================================================
4
5  ProcessConfig = pBasicContactMgr
6  {
7    AppTick   = 4
8    CommsTick = 4
9
10   // Alert configurations (one or more, keyed by id)
11   alert = id=avd, var=CONTACT_INFO
12   alert = id=avd, val="name=avd_$[VNAME] # contact=$[VNAME]"
13   alert = id=avd, range=80, alert_range_color=white
14   alert = id=avd, cpa_range=95, cpa_range_color=gray50
15
16   // Properties for all alerts
17   default_alert_range       = 1000   // the default in meters
18   default_cpa_range         = 1000   // the default in meters
19   default_alert_range_color = color  // the default is gray65
20   default_cpa_range_color   = color  // the default is gray35
21
22   // Policy for retaining potential stale contacts
23   contact_max_age  = 3600            //  the default in secs.
24
```

```
25    // Configuring other output
26    display_radii    = false  // or {true}
27    alert_verbose    = false  // If true, ALERT_VERBOSE published.
28
29    // Policy for linear extrapolation of stale contacts
30    decay = 30,60                      // the default in secs
31
32    contacts_recap_interval = 5        // the default in secs
33
34    contact_local_coords    = verbatim  // the default
35  }
```

7 Publications and Subscriptions for pBasicContactMgr

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

```
$ pBasicContactMgr --interface or -i
```

7.1 Variables Published by pBasicContactMgr

The primary output of pBasicContactMgr to the MOOSDB is the set of user-configured alerts. Other variables are published on each iteration where a change is detected on its value:

- 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 5.
- CONTACTS_LIST: A comma-separated list of contacts.
- CONTACTS_RECAP: A comma-separated list of contact summaries.
- CONTACTS_ALERTED: A list of contacts for which alerts have been posted.
- CONTACTS_UNALERTED: A list of contacts for which alerts are pending, based on the range criteria.
- CONTACTS_RETIRED: A list of contacts removed due to the information staleness.
- CONTACT_MGR_WARNING: A warning message indicating possible mishandling of or missing data.
- VIEW_CIRCLE: A rendering of the alert ranges.

Some examples:

```
CONTACTS_LIST      = gus,joe,ken,kay
CONTACTS_ALERTED   = gus,kay
CONTACTS_UNALERTED = ken,joe
CONTACTS_RETIRED   = bravo,foxtrot,kilroy
CONTACTS_RECAP     = name=gus,age=7.3,range=13.1 # name=ken,age=0.7,range=48.1 # \
                     name=joe,age=1.9,range=73.1 # name=kay,age=4.0,range=18.2
```

7.2 Variables Subscribed for by pBasicContactMgr

The pBasicContactMgr application will subscribe for the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- BCM_DISPLAY_RADII: If false, no postings will be made for rendering the alert and cpa range circles.
- BCM_ALERT_REQUEST: If false, no postings will be made for rendering the alert and cpa range circles.
- CONTACT_RESOLVED: A name of a contact that has been declared resolved, possibly with a particular alert specified.
- NAV_HEADING: Present ownship heading in degrees.
- NAV_SPEED: Present ownship speed in meters per second.
- NAV_X: Present position of ownship in local x coordinates.
- NAV_Y: Present position of ownship in local y coordinates.
- NODE_REPORT: A report about a known contact.

7.3 Command Line Usage of pBasicContactMgr

The pBasicContactMgr application is typically launched as a part of a batch of processes by pAntler, but may also be launched from the command line by the user. To see command-line options enter the following from the command-line:

```
$ pBasicContactMgr --help or -h
```

This will show the output shown in Listing 6 below.

Listing 7.6: Command line usage for the pBasicContactMgr application.

```
1  ============================================================
2  Usage: pBasicContactMgr file.moos [OPTIONS]
3  ============================================================
4
5  SYNOPSIS:
6  ------------------------------------
7    The contact manager deals with other known vehicles in its
8    vicinity. It handles incoming reports perhaps received via a
9    sensor application or over a communications link. Minimally
10   it posts summary reports to the MOOSDB, but may also be
11   configured to post alerts with user-configured content about
12   one or more of the contacts.
13
14 Options:
15   --alias=<ProcessName>
16       Launch pBasicContactMgr with the given process
17       name rather than pBasicContactMgr.
18   --example, -e
19       Display example MOOS configuration block.
20   --help, -h
21       Display this help message.
22   --interface, -i
23       Display MOOS publications and subscriptions.
```

```
  --version,-v
      Display the release version of pBasicContactMgr.

Note: If argv[2] does not otherwise match a known option,
      then it will be interpreted as a run alias. This is
      to support pAntler launching conventions.
```