**MACE Regression Test (Doc: RT\_01)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case #:** | RT\_01 | **Test Case Name:** | MACE Regression Test |
| **System(s):** | MACE API  MACE Core | **Subsystem(s):** | MACE External Link Module  MACE Vehicle Module  MACE GCS Module  MACE GUI |
| **Designed by:** | Patrick Nolan | **Design Date:** | 7/10/2018 |
| **Executed by:** | *<Tester(s)>* | **Execution Date:** | *<Test Date>* |
| **Short Description:** | Basic regression test for MACE communications and high level C2 of a vehicle. Instructions for both local and external link connections are provided. | | |

|  |  |
| --- | --- |
| **Requirement ID** | **Requirement Description** |
| ###-### | The MACE ground instance connects to an airborne MACE instance via External Link |
| ###-### | The MACE ground instance connects to a simulated vehicle via a local link. |
| ###-### | The MACE GUI displays the correct vehicle HUD. |
| ###-### | The MACE GUI receives vehicle updates (heartbeat, pose, battery level, etc.). |
| ###-### | The MACE GUI can issue commands to the vehicle (arm/disarm, takeoff, initiate auto mission, etc.). |
| ###-### | The MACE GUI can sync all data for all connected vehicles (vehicle missions, vehicle modes, home positions, etc.). |

|  |  |
| --- | --- |
| **Reference Document** | **Document Description** |
| R\_01 | Requirements Specification |
| IG\_01 | MACE Installation and Compilation Guide (<https://github.com/heronsystems/MACE/wiki/install-and-run>) |

|  |  |
| --- | --- |
| **Hardware Item** | **Hardware Description** |
| Computer (2) | At least two computers are required: One to run the ground instance of MACE, one to run the airborne instance. |
| Networking Equipment | XBee radios (Digimesh) |

|  |
| --- |
| **Pre-conditions:**  Prior to executing this test case, MACE and supporting libraries must be installed and compiled in accordance with “IG\_01: MACE Intallation and Compilation Guide.” Make sure to also install/configure the Ardupilot SITL simulator on the machine that will be running the Vehicle Comms module. If testing the External Link portion of this regression test, two computers are needed each with an XBee serial radio connected.  **Dependencies:**  There are no dependencies for this test case. |

|  |
| --- |
| **Input Test Data:**  General Test Configuration (*Local link*):   * One MACE instance: * Ground MACE Instance—Ground station module, Vehicle Comms Module * MACE GUI   General Test Configuration (*External link*):   * Two MACE instances: * Ground MACE Instance—Ground station module, External Link Module * Airborne MACE instance—External Link Module, Vehicle Comms Module * MACE GUI   Example configuration files can be found in the directory of the MACE repository.  Safety Constraints:   * None |

**Option 1: Local link setup and Execution**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected System Response** | **Pass/Fail** | **Comment(s)** |
| *Setup—Ground Instance* | | | | |
|  | *(WINDOWS)* Open a Cygwin terminal window and navigate to the ardupilot/ArduCopter directory.  *(LINUX)* Open a terminal window and navigate to the ardupilot/ArduCopter directory | The working directory is changed to the ArduCopter directory. |  |  |
|  | Launch a simulated ArduCopter vehicle by running the following command (all one line):  $ sim\_vehicle.py -j4 -I 4 --console  --out=udp:127.0.0.1:14558 --custom-location=37.890425,-76.811910,0,240 | A MAVProxy window and accompanying console window will open. The simulated vehicle will go through its initial startup procedure. |  |  |
|  | In the SITL terminal window, load a mission with the following command (all one line):  $ wp load ..\Tools\autotest\copter\_mission\_simple.txt | The SITL terminal will indicate the waypoints were loaded successfully and display the loaded waypoints. |  |  |
|  | Open a command prompt (Windows) or two (2) terminal windows (Linux) and navigate to the MACE GUI directory:  $ cd <Path>/<to>/<MACE>/ElectronGUI  Make sure to change <Path>/<to>/<MACE>/ to the path where MACE was cloned on your machine. | The working directory is changed to the MACE Electron GUI directory. |  |  |
|  | *(WINDOWS)* Launch the MACE GUI by running the following:  $ yarn run launch  *(LINUX)* In the first terminal, run the following:  $ yarn run watch  In the second terminal, run the following:  $ yarn run start | The MACE GUI will open and a map should be displayed. |  |  |
|  | Create a configuration file (or modify an existing one in the MACE/MaceSetup\_Configs/ directory) and make sure the vehicle comms ListenAddress and GUI Host Address parameters are set to localhost (i.e. 127.0.0.1).  In addition to the IP addresses, take note of the ports—for the Vehicle Comms module, that port should match the output port in the sim\_vehicle.py script. For the Ground Station module, take note of the send and listen ports (to be used in the next step). | The configuration file is set up with correct addresses and ports. |  |  |
|  | Modify the GUIConfig.json file in the top-level MACE directory to have the ipAddress as localhost (i.e. 127.0.0.1) and make sure the listen/send port numbers match those in the MACE XML config from Step 6. | The configuration file is set up with correct addresses and ports. |  |  |
| *Test Execution* | | | | |
|  | Open a command prompt (Windows) or a terminal window (Linux) and navigate to the MACE/bin directory:  $ cd <Path>/<to>/<MACE>/bin  Make sure to change <Path>/<to>/<MACE>/ to the path where MACE was cloned on your machine. | The working directory is changed to the MACE bin directory. |  |  |
|  | Run the MACE executable with the desired XML configuration file as an argument:  $ MACE <Config>/<Path>/<filename>.xml  Make sure to change <Config>/<Path> to the path where MACE config file is located. Also change <filename> to the configuration file’s name. | MACE will start with the configuration loaded from <filename>.xml  Note that in this case, the xml file should only have a Vehicle Comms module and a Ground Station module loaded. |  |  |
|  | Observe the MACE GUI for updates from the connected vehicle. | The connected vehicle will have a Vehicle HUD appear on the right side of the screen. The state data and vehicle health will update at regular intervals.  The vehicle home position and current vehicle position will be populated on the map with a home icon and vehicle/quadrotor icon, respectively. |  |  |
|  | Click the “SYNC ALL” button on the top right of the GUI. | The GUI will initiate a data sync for all connected vehicles (in this case, only 1). The vehicle mode will update, the vehicle mission and home location will also be displayed on the map.  **NOTE**: If prior to this step the mission and home location were populated on the map, refresh the GUI (View->Reload) and then hit “SYNC ALL” to test this functionality. |  |  |
|  | Click on the vehicle icon on the map to select the vehicle. | The vehicle icon will highlight, as well as the vehicle HUD. |  |  |
|  | Click the “ARM” button at the bottom of the screen. | The ardupilot SITL console window will acknowledge the receipt of the ARM command and arm the vehicle.  The MACE GUI will display an “Arming motors” message in the vehicle HUD. The vehicle will be armed and its icon in the HUD will turn green. |  |  |
|  | With the vehicle still selected, click the “DISARM” button at the bottom of the screen. | The ardupilot SITL console window will acknowledge the receipt of the ARM command and disarm the vehicle.  The MACE GUI will display an “Disarming motors” message in the vehicle HUD. The vehicle will be disarmed and its icon in the HUD will turn orange. |  |  |
|  | With the vehicle still selected, click the “TAKEOFF” button at the bottom of the screen. | The takeoff dialog appears. The selected vehicle ID will appear in the dropdown. |  |  |
|  | Click the “TAKEOFF” button in the takeoff dialog. | The vehicle will perform the following sequence:   1. Arm 2. Switch to GUIDED mode 3. Takeoff to the takeoff altitude (default to 10m)   Monitor the vehicle HUD on the GUI for each of those steps and make sure the vehicle climbs to ~10m |  |  |
|  | With the vehicle still selected, click the “START MISSION” button at the bottom of the screen. | The vehicle changes to AUTO mode and starts moving towards its first waypoint. The word “active” will appear over the first waypoint. |  |  |
|  | Observer the vehicle move through its first two waypoints. | As the vehicle reaches each waypoint, a message will appear in the HUD indicating a new waypoint target. The “active” keyword will move to the next waypoint. |  |  |
|  | With the mission still in motion and the vehicle still selected, click the “PAUSE” button at the bottom of the screen. | The vehicle will change to “BRAKE” mode and stop its motion/hover in place. |  |  |
|  | With the vehicle still selected, click the “HOME” button at the bottom of the screen. | The vehicle will change to “RTL” mode and move towards its home position.  The vehicle will arrive above its home position and descend to the ground. The motors will be disarmed when landed. |  |  |

**Option 2: External Link setup and Execution**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected System Response** | **Pass/Fail** | **Comment(s)** |
| *Setup—Ground Instance* | | | | |
|  | Open a command prompt (Windows) or two (2) terminal windows (Linux) and navigate to the MACE GUI directory:  $ cd <Path>/<to>/<MACE>/ElectronGUI  Make sure to change <Path>/<to>/<MACE>/ to the path where MACE was cloned on your machine. | The working directory is changed to the MACE Electron GUI directory. |  |  |
|  | *(WINDOWS)* Launch the MACE GUI by running the following:  $ yarn run launch  *(LINUX)* In the first terminal, run the following:  $ yarn run watch  In the second terminal, run the following:  $ yarn run start | The MACE GUI will open and a map should be displayed. |  |  |
|  | Connect the XBee serial radio to a USB port. Take note of the COMM port.  (*WINDOWS*) Open the device manager and note the radio’s COM port number under “Ports (COM and LPT)”. It will be in the form of “COM4”  (*LINUX*) Open a terminal and type ls /dev/ttyU and then hit TAB a couple times until it tries to autocomplete. If the XBee is the only USB device plugged in, it should display the com port for the radio in the form of “ttyUSB0”. | The XBee is given a COMM port. |  |  |
|  | Create a configuration file (or modify an existing one in the MACE/MaceSetup\_Configs/ directory) and make sure the External Link “PortName” and GUI Host Address parameters are correct. The “PortName” should be set to the port noted in Step 3 above. The GUI host address should be set to localhost (i.e. 127.0.0.1). | The configuration file is set up with correct addresses and ports. |  |  |
|  | Modify the GUIConfig.json file in the top-level MACE directory to have the ipAddress as localhost (i.e. 127.0.0.1) and make sure the listen/send port numbers match those in the MACE XML config from Step 4. | The configuration file is set up with correct addresses and ports. |  |  |
|  | Take note of the IP address of this machine:  (*WINDOWS*) Open a command prompt and type ipconfig  (*LINUX*) Open a command prompt and type ifconfig | The machine is connected to the internet and an IP address is assigned. |  |  |
| *Setup—Airborne Instance* | | | | |
|  | *(WINDOWS)* Open a Cygwin terminal window and navigate to the ardupilot/ArduCopter directory.  *(LINUX)* Open a terminal window and navigate to the ardupilot/ArduCopter directory. | The working directory is changed to the ArduCopter directory. |  |  |
|  | Launch a simulated ArduCopter vehicle by running the following command (all one line):  $ sim\_vehicle.py -j4 -I 4 --console  --out=udp:127.0.0.1:14558 --custom-location=37.890425,-76.811910,0,240 | A MAVProxy window and accompanying console window will open. The simulated vehicle will go through its initial startup procedure. |  |  |
|  | In the SITL terminal window, load a mission with the following command (all one line):  $ wp load ..\Tools\autotest\copter\_mission\_simple.txt | The SITL terminal will indicate the waypoints were loaded successfully and display the loaded waypoints. |  |  |
|  | Connect the XBee serial radio to a USB port. Take note of the COMM port.  (*WINDOWS*) Open the device manager and note the radio’s COM port number under “Ports (COM and LPT)”. It will be in the form of “COM4”  (*LINUX*) Open a terminal and type ls /dev/ttyU and then hit TAB a couple times until it tries to autocomplete. If the XBee is the only USB device plugged in, it should display the com port for the radio in the form of “ttyUSB0”. | The XBee is given a COMM port. |  |  |
|  | Create a configuration file (or modify an existing one in the MACE/MaceSetup\_Configs/ directory) and make sure the External Link “PortName” and Vehicle Comms UDP parameters are correct.  The “PortName” should be set to the port noted in Step 10 above. For the Vehicle Comms module, that listen port number should match the output port in the sim\_vehicle.py script and the listen address should be set to localhost (i.e. 127.0.0.1). | The configuration file is set up with correct addresses and ports. |  |  |
| *Test Execution—Ground Instance* | | | | |
|  | Open a command prompt (Windows) or a terminal window (Linux) and navigate to the MACE/bin directory:  $ cd <Path>/<to>/<MACE>/bin  Make sure to change <Path>/<to>/<MACE>/ to the path where MACE was cloned on your machine. | The working directory is changed to the MACE bin directory. |  |  |
|  | Run the MACE executable with the desired XML configuration file as an argument:  $ MACE <Config>/<Path>/<filename>.xml  Make sure to change <Config>/<Path> to the path where MACE config file is located. Also change <filename> to the configuration file’s name (created in Step 4 above). | MACE will start with the configuration loaded from <filename>.xml  Note that in this case, the xml file should only have an External Link module and a Ground Station module loaded. |  |  |
| *Test Execution—Airborne Instance* | | | | |
|  | Open a command prompt (Windows) or a terminal window (Linux) and navigate to the MACE/bin directory:  $ cd <Path>/<to>/<MACE>/bin  Make sure to change <Path>/<to>/<MACE>/ to the path where MACE was cloned on your machine. | The working directory is changed to the MACE bin directory. |  |  |
|  | Run the MACE executable with the desired XML configuration file as an argument:  $ MACE <Config>/<Path>/<filename>.xml  Make sure to change <Config>/<Path> to the path where MACE config file is located. Also change <filename> to the configuration file’s name (created in Step 11 above). | MACE will start with the configuration loaded from <filename>.xml  Note that in this case, the xml file should only have an External Link module and a Ground Station module loaded. |  |  |
| *Test Execution—Ground Instance* | | | | |
|  | Observe the MACE GUI for updates from the connected vehicle. | The connected vehicle will have a Vehicle HUD appear on the right side of the screen. The state data and vehicle health will update at regular intervals.  The vehicle home position and current vehicle position will be populated on the map with a home icon and vehicle/quadrotor icon, respectively. |  |  |
|  | Click the “SYNC ALL” button on the top right of the GUI. | The GUI will initiate a data sync for all connected vehicles (in this case, only 1). The vehicle mode will update, the vehicle mission and home location will also be displayed on the map.  **NOTE**: If prior to this step the mission and home location were populated on the map, refresh the GUI (View->Reload) and then hit “SYNC ALL” to test this functionality. |  |  |
|  | Click on the vehicle icon on the map to select the vehicle. | The vehicle icon will highlight, as well as the vehicle HUD. |  |  |
|  | Click the “ARM” button at the bottom of the screen. | The MACE GUI will display an “Arming motors” message in the vehicle HUD. The vehicle will be armed and its icon in the HUD will turn green. |  |  |
|  | With the vehicle still selected, click the “DISARM” button at the bottom of the screen. | The MACE GUI will display an “Disarming motors” message in the vehicle HUD. The vehicle will be disarmed and its icon in the HUD will turn orange. |  |  |
|  | With the vehicle still selected, click the “TAKEOFF” button at the bottom of the screen. | The takeoff dialog appears. The selected vehicle ID will appear in the dropdown. |  |  |
|  | Click the “TAKEOFF” button in the takeoff dialog. | The vehicle will perform the following sequence:   1. Arm 2. Switch to GUIDED mode 3. Takeoff to the takeoff altitude (default to 10m)   Monitor the vehicle HUD on the GUI for each of those steps and make sure the vehicle climbs to ~10m |  |  |
|  | With the vehicle still selected, click the “START MISSION” button at the bottom of the screen. | The vehicle changes to AUTO mode and starts moving towards its first waypoint. The word “active” will appear over the first waypoint. |  |  |
|  | Observer the vehicle move through its first two waypoints. | As the vehicle reaches each waypoint, a message will appear in the HUD indicating a new waypoint target. The “active” keyword will move to the next waypoint. |  |  |
|  | With the mission still in motion and the vehicle still selected, click the “PAUSE” button at the bottom of the screen. | The vehicle will change to “BRAKE” mode and stop its motion/hover in place. |  |  |
|  | With the vehicle still selected, click the “HOME” button at the bottom of the screen. | The vehicle will change to “RTL” mode and move towards its home position.  The vehicle will arrive above its home position and descend to the ground. The motors will be disarmed when landed. |  |  |
| *Test Execution—Airborne Instance* | | | | |
|  | For each command issued from the GUI/MACE ground instance, observe the ardupilot SITL console window and airborne MACE instance terminal window. | The ardupilot SITL console window will acknowledge commands and update its data accordingly (e.g. the receipt of the ARM command and arming the vehicle will be displayed in the console after the ARM command is issued). |  |  |

|  |
| --- |
| **Post-conditions:**  MACE handles the dissemination of vehicle state data to and from ground/airborne instances of MACE in the External Link case. In the local link case, this is handled all within the ground MACE instance. In both setup cases, the GUI will update with up to date vehicle state and health data. The GUI will also be able to sync all data for connected vehicles (e.g. vehicle mission, home position, vehicle mode, etc.). The GUI can initiate commands for the vehicle (i.e. arm/disarm, takeoff, start mission, pause mission, return to launch) and the vehicle will execute each command, with updated status displayed on the GUI. During the test, no unexpected abort sequences occur, causing the test to end prematurely. |

|  |  |
| --- | --- |
| **Test Results:** *<What was the actual state/result after executing this test case? Describe system behavior after test execution>* | **Pass/Fail:** *<If the actual test results do NOT match the expected post-conditions, mark this test as failed. Otherwise, mark this test as passed.>* |

|  |  |
| --- | --- |
| **Defect ID** | **Defect Description** |
| *<If the test fails, include a reference to the defect ID (i.e. Issue #)>* | *<Description of the defect—cause(s), system behavior, potential mitigations, etc.>* |

|  |
| --- |
| **General Notes/Comments:** *<Include any additional commentary for this test case. Support the above fields with information that may not be described in any of the above fields, or if there are questions related to expected or actual results.>* |