INT-BALL2
TECHNOLOGY DEMONSTRATION USER PROGRAMING PLATFORM
USER'S MANUAL

Japan Aerospace Exploration Agency (JAXA)

Human Spaceflight Technology Center

Revision History

Code	Approved date	Approval	Inspection	Preparation	Revised parts	Revision details, reasons, etc.
NC					N/A	Establishment of the first edition
A	See the Cover Page					English Translation added

Contents

1. Objectives of this document

The JEM Internal Ball Camera 2 System (Int-Ball2) is a camera robot that flies in the ISS JEM module by remote control from the ground to take video images. Int-Ball2 can run user-developed software as its extended functionality and can be used as a platform for demonstrating robotic technology in space. This manual describes how to prepare the environment for using technology demonstration.

2. Related Documents

N/A

3. Overview of Int-Ball2 Technology Demonstration Platform

The Int-Ball2 flight software provides users with sensor data held by Int-Ball2 and control input interfaces to the actuators, as the response to the technology demonstration environment. This allows the user to customize Int-Ball2's flight software by implementing their own navigation (Visual SLAM, sensor fusion, etc.) and guidance control functions (PID control, visual feedback control) and so on. In addition, when the user-implemented program is applied to the Int-Ball2 body, it can be linked to some of the existing functions, and the scope of functions to be implemented by the user as a technology demonstration can be adjusted.

The technology demonstration platform consists of the following:

- Technology Demonstration Platform Software
- Ground Support Equipment for Technology Demonstration Platform
- Technology Demonstration Platform Simulation
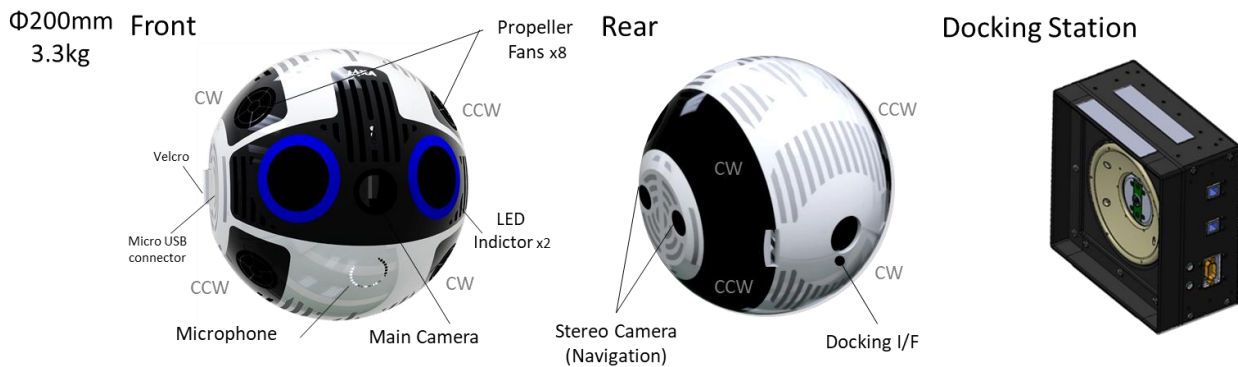
Figure 1: Overview of Int-Ball2 hardware

Each function of the Int-Ball2 flight software is implemented as a node of ROS (Robot Operating System), and it is possible to control whether a function (node) is activated according to the scope of the technical demonstration by simply changing the settings without modifying the existing source code. In addition, since the internal interfaces for data output and function calls in existing functions are unified in the ROS message format (topic, service, action), the output data and processing of existing functions can be utilized by using the implemented internal communication interfaces when adding new functions.

The Int-Ball2 Technology Demonstration Platform is operated through telemetry and command transmission and reception between orbit and the ground via a communication line like the existing Int-Ball2 flight software.

The demonstration platform functions run with Docker and are virtually separated from the native environment. User programs are uplinked to the Int-Ball2 body by the operations controllers and executed by activating them via the Ground Support Equipment (GSE). This allows the addition or replacement of new programs and logics as well as technology demonstrations on Int-Ball2 without affecting the functions of existing flight software and the GSE.
In addition, these programs can run in the Int-Ball2 simulation environment.

[Constraints necessary to be considered when using this demonstration platform]
- Software running on the demonstration platform is activated with Docker. The activation command is sent from the ground. No real-time communication with user software on the ground is performed during the basic demonstration. (If demonstration software that requires communication is to be implemented, this shall be coordinated with the Int-Ball2 base station.)
- The size of the demonstration program shall be coordinated with the Int-Ball2 base station. (Subject to the availability of the Int-Ball2 body and SD card capacity.)
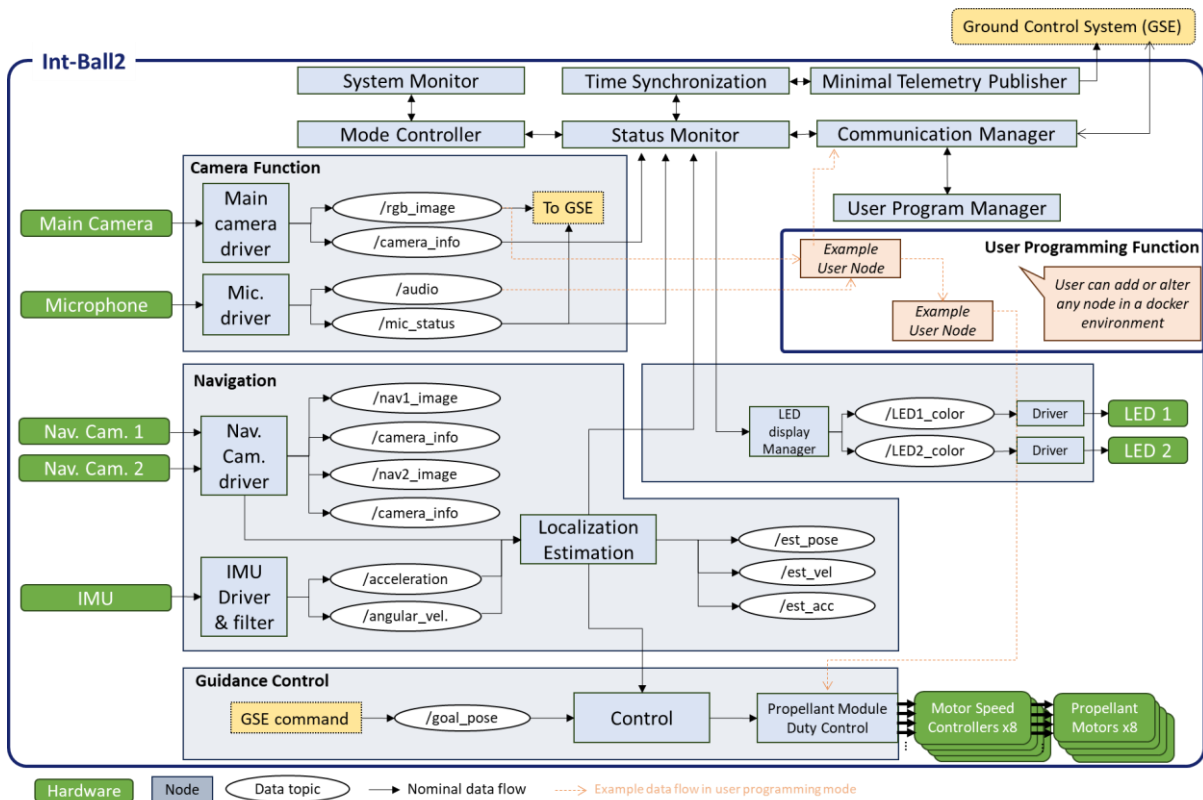

Figure 5: Overview of the Int-Ball2 Existing Flight Software System Configuration

Currently, the existing Visual-SLAM (Simultaneous Localization and Mapping) implemented in Int-Ball2 cannot be used when separately using the video from the navigation camera. (If using the video from a navigation camera, it is necessary to implement SLAM by yourself.)

4. Environment building procedure

4.1 Operation environment

- OS: Ubuntu 18.04
- Middleware: Robot Operating System (ROS/Melodic)
- Int-Ball2 Body Processor: Nvidia Jetson TX2, Linux for Tegra (Ubuntu Based File System)

4.2 Installing OS

The Int-Ball2 Technology Demonstration Platform is operated on Ubuntu 18.04.

Install Ubuntu 18.04 in your own environment with reference to the following website:

Ubuntu 18.04.6 LTS（Bionic Beaver） (as of Mar. 28, 2024)

4.3 Installing ROS

The Int-Ball2 Technology Demonstration Platform and its simulator are executed with ROS/Gazebo. Therefore, these shall be installed. Since it is necessary to use a Gazebo version higher than 9.0.0, it shall be installed with the following procedure:

1. Preparation for acquisition of the latest version, Gazebo 9: to enable acquiring the latest version, Gazebo9, execute 1 and 2 of the "Alternative installation: step-by-step" on the following website:
   Gazebo : Tutorial : Ubuntu (gazebosim.org) (as of Mar. 28, 2024)

2. Installing ROS: Install ROS by executing the procedure through 1.1 to 1.6 on the following website:
   melodic/Installation/Ubuntu – ROS Wiki (as of Mar. 28, 2024)
   In the step 1.4."Installation", select "sudo apt install ros-melodic-desktop".

3. Installing the ROS related package: install the package for collaboration with ROS-Gazebo by the following command:

```
sudo apt install ros-melodic-gazebo-*
```

4.4 Installing Python3

Since Python3 and its related package are used in this system, install the related package and change the settings file with the following procedure:

1. Install Python3.

```
sudo apt install python3 python3-pip
```

2. Install the ROS related package of Python3.

```
pip3 install rospkg
pip3 install empy==3.3.4
```

※When "empy" is installed with "pip3 install rospkg empy", the latest version (4.1 at the time of writing this) is installed. However, as described in the reference website

(http://soup01.com/ja/2023/12/25/post-9789/), it has been confirmed that this version has a bug that causes an error called "AttributeError: 'module' object has no attribute 'RAW_OPT'". Therefore, it is necessary to install it while specifying the version as 3.3.4.

3. The Python to be used for building the ROS package (catkin) shall be changed to Python3. Newly prepare the settings file and add the variables.

```
sudo vi /opt/ros/melodic/etc/catkin/profile.d/1.ros_python_version.sh
…
export ROS_PYTHON_VERSION=3
…
```

4.5 Ground Support Equipment

Software of the Technology Demonstration Platform is executed by the Ground Support Equipment (GSE) of Int-Ball2.

The Int-Ball2 GSE operation environment shall be prepared with the following procedure:

4.5.1 Netwide Assembler (NASM)

The assembler called "NASM" shall be built and installed from the source file through the following procedure:

1. Decompress and place the nasm source files under "/usr/local/src" and the following directory:

```
cd /usr/local/src
sudo wget https://www.nasm.us/pub/nasm/releasebuilds/2.15.05/nasm-2.15.05.tar.gz
sudo tar zxvf nasm-2.15.05.tar.gz
```

2. Move to the following directory where the source file is placed and conduct the building settings:

```
cd nasm-2.15.05
sudo ./configure
```

3. Build the source file and install it.

```
sudo make install
```

※ As the reference description (https://trans-it.net/centos7-ffmpeg43-h264-fdkaac/), it is necessary to install "nasm" in advance to install ffmpeg.

4.5.2 Video reception environment

"x264" shall be built from the source file through the following procedure:

1. Decompress and place the source file under "/usr/local/src".

```
sudo tar jxvf x264-master.tar.bz2 -C /usr/local/src/
```

2. Move to the directory where the source file is placed and conduct the building settings.

```
cd /usr/local/src/x264-master
sudo ./configure ¥
--disable-asm --enable-shared --enable-static --enable-pic
```

3. Build the source file and install it.

```
sudo make install
```

ffmpeg shall be built from the source file through the following procedure:

4. Decompress and place the source file of "ffmpeg" under "/usr/local/src".

```
sudo tar xvzf ffmpeg-4.1.3.tar.gz -C /usr/local/src/
```

5. Move to the directory where the source file is placed and conduct the building settings.

```
cd /usr/local/src/ffmpeg-4.1.3
sudo ./configure ¥
--extra-cflags="-I/usr/local/include" ¥
--extra-ldflags="-L/usr/local/lib" ¥
--extra-libs="-lpthread -lm -ldl -lpng" ¥
--enable-pic ¥
--disable-programs ¥
--enable-shared ¥
--enable-gpl ¥
--enable-libx264 ¥
--enable-encoder=png ¥
--enable-version3
```

6. Build the source file and install it.

```
sudo make install
```

4.5.3 Installing VLC media player

The VLC media player shall be installed and built through the following procedure:

1. Install the dependency packages.

```
sudo apt install libasound2-dev libxcb-shm0-dev libxcb-xv0-dev ¥
libxcb-keysyms1-dev libxcb-randr0-dev libxcb-composite0-dev ¥
lua5.2 lua5.2-dev protobuf-compiler bison libdvbpsi-dev
```

2. Decompress and place the downloaded source file under "/usr/local/src".

```
sudo tar Jxvf vlc-3.0.7.1.tar.xz -C /usr/local/src/
```

3. Move to the directory where the source file is placed and conduct the building settings.

```
cd /usr/local/src/vlc-3.0.7.1
# Execute setting variables for building and "configure" on one line.
CFLAGS="-I/usr/local/include" ¥
LDFLAGS="-L/usr/local/lib" ¥
X264_CFLAGS="-L/usr/local/lib -I/usr/local/include" ¥
X264_LIBS="-lx264" ¥
X26410b_CFLAGS="-L/usr/local/lib -I/usr/local/include" ¥
X26410b_LIBS="-lx264" ¥
AVCODEC_CFLAGS="-L/usr/local/lib -I/usr/local/include" ¥
AVCODEC_LIBS="-lavformat -lavcodec -lavutil" ¥
AVFORMAT_CFLAGS="-L/usr/local/lib -I/usr/local/include" ¥
AVFORMAT_LIBS="-lavformat -lavcodec -lavutil" ¥
sudo ./configure ¥
--disable-a52 ¥
--enable-merge-ffmpeg ¥
--enable-x264 ¥
--enable-x26410b ¥
--enable-dvbpsi
```

4. Build the source file and install it.

```
sudo make install
```

5. Prepare the symbolic link to the downloaded source file.

```
sudo ln -s /usr/local/src/vlc-3.0.7.1 /usr/local/src/vlc
```
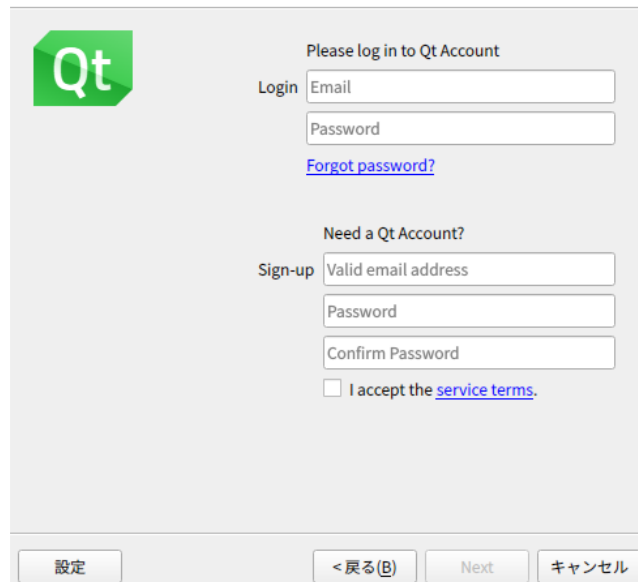
4.5.4 Qt

"Qt" shall be installed through the following procedure:

1. Prepare the installing directory.

```
sudo mkdir /opt/Qt
```

2. Activate the downloaded installer for Linux. (Account registration shall be necessary.)
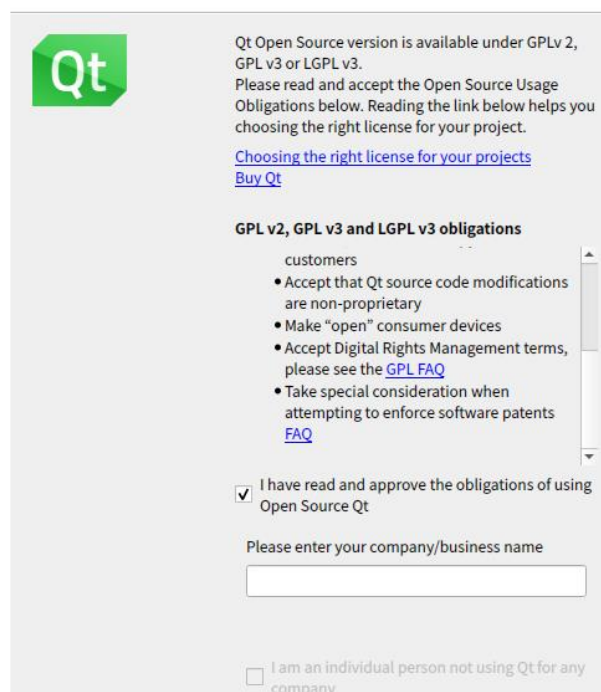

3. Check the box of "Using Open Source Qt".

4. Specify the installation directory as "/opt/Qt".
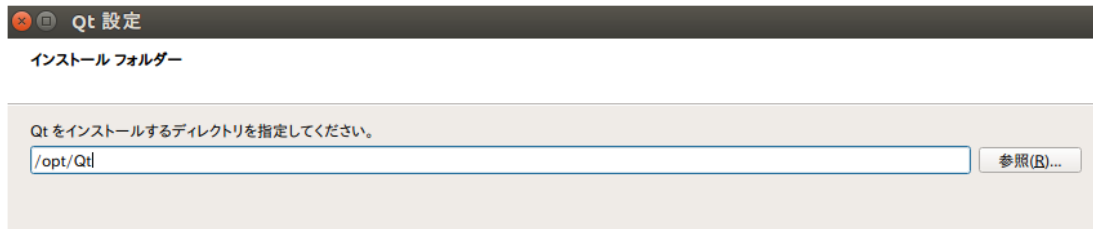
5. Select "5.12.3 Desktop gcc 64-bit" as the targeted version for installation and follow the installer's instructions thereafter:
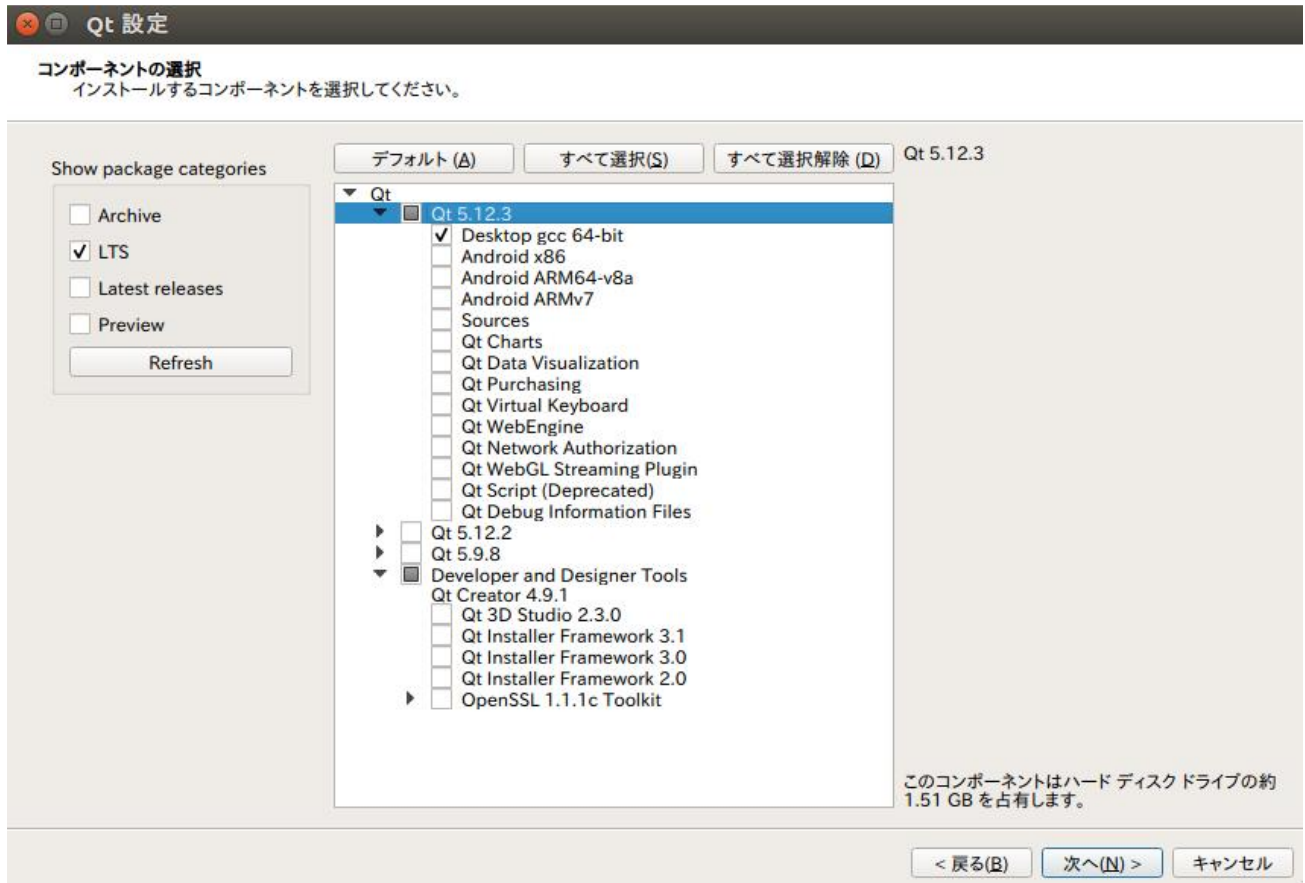
6. After the installation is completed, prepare the symbolic link.

```
sudo ln -s /opt/Qt/5.12.3 /opt/Qt/5
```

4.5.5 Font file

Since the Int-Ball2 GSE uses the Roboto font, copy the source file of the font to the following path and install it:

```
sudo cp Roboto*.ttf /usr/local/share/fonts/.
fc-cache -f -v
```

4.5.6 Source code deployment

Deploy "Int-Ball2_platform_gse" to an arbitrary directory.

4.5.7 Parameter settings

To exchange telemetry and commands between the Int-Ball2 Technology Demonstration Platform Simulator and the GSE, the communication setting parameters shall be set as follows:

- Command Transmission Settings

  Set the IP address of the computer for activating this system at "intball2_tele command_target_ip" on "IntBall2_platform_gse¥src¥ground_system¥communicati on_software¥config¥params.yml". And set an arbitrary command transmission port at "intball2_telecommand_target_port".

- Telemetry Receiving Settings

  Set an arbitrary telemetry receiving port at "intball2_telemetry_receive_port" on "Int-Ball2_ platform_gse¥src¥ground_system¥communication_software¥config¥params.yml".

4.6 Int-Ball2 Technology Demonstration Platform Simulator

Prepare the operation environment for using the Int-Ball2 Technology Demonstration Platform Simulator.

4.6.1 Docker

Install "Docker" with reference to the procedure on the following website:

[Install Docker Engine on Ubuntu | Docker Docs](#) (as of Mar. 28, 2024)

Follow steps 1 and 2 of "Install using the apt repository".

4.6.2 Python

The Docker-related package shall be installed on Python3.

```
pip3 install docker defusedxml netifaces
```

4.6.3 Container activation settings

Services for containers shall always be effective.

```
sudo systemctl enable docker docker.socket
sudo systemctl start docker.socket
```

Grant the authority for activating containers to the user executing this system. The following shows the procedure assuming that the target user is "nvidia":

```
# Authority setting
sudo gpasswd -a nvidia docker
sudo chgrp docker /var/run/docker.sock
# Restart services for containers
```

```
sudo service docker restart
```

Execute the following command with the user executing this system and confirm that no error including the phrase "permission denied" occurs. If operating via the SSH connection, disconnect the SSH once and restart the operation after reconnection.

```
docker ps
```

4.6.4 Source code deployment

Deploy "Int-Ball2_platform_simulator" in an arbitrary directory.

Also, deploy "platform_works" under the home directory of the user executing this system.

4.6.5 Parameter settings

To exchange telemetry and commands between the Int-Ball2 Technology Demonstration Platform Simulator and the GSE, the communication setting parameters shall be set as follows:

- Command transmission settings

  Set the same value of "intball2_telecommand_target_port" set at "receive port" in 5.4.7.1 on "Int-Ball2_platform_simulator¥src¥flight_software¥trans_communication¥launch¥11bringup.launch".

- Telemetry receiving settings

  Set the IP address of the computer for activating this system at "ocs_host" on "Int-Ball2_platform_simulator¥src¥flight_software¥trans_communication¥launch¥ bringup.launch".

  And set the same value of "intball2_telemetry_receive_port" in 5.4.7.1. at "ocs port".

4.6.6 Docker settings

The Int-Ball2 Technology Demonstration Platform executes the user program with using Docker. The following shows the settings related to the exchange between the host and the container in this case:

1. IP of the host computer

    Set the IP address of the computer which activates this system at "container_ros_master_uri" on "Int-Ball2_platform_simulator¥src¥platform_sim¥platform_sim_tools¥launch¥platform_manager_bringup.launch".

2. Container workspace in the host computer

    Set the path to "Int-Ball2_platform_simulator" as "platform works" at "host_ib2_workspace" on "Int-Ball2_platform_simulator¥src¥platform_sim¥platform_sim_tools¥launch¥platform_manager_bringup.launch".

4.6.7 Setting up of containers

Since the Technology Demonstration Platform executes the user program on a container, it is necessary to execute the container building for execution of the User Demonstration Platform.

The tag name (ib2_user) part can be set arbitrarily but must be a tag name with the prefix "ib2" for containers

handled by the flight software.

```
cd ~/platform_works/platform_docker/template
docker build . -t ib2_user:0.1
```

Docker images are not supported on the cross-platform and cannot be used with those prepared on another CPU architecture (it seems that those are prepared on Intel/AMD for normal computers, but for Int-Ball2, those are prepared on ARM, so they cannot be used). Therefore, in order to prepare images to run on the actual Int-Ball2, it is necessary to prepare those for ARM while using the extended plug-in called Docker buildx as shown in the following command, instead of the above mentioned command. See the reference website "https://qiita.com/engishoma/items/0c7c6c3ae3dc173b01b2".

```
docker buildx build --platform linux/arm64/v8 -t ib2_user:(version) --load.
```

After the building is completed, execute the following command and confirm that "ib2_user" appears in the "REPOSITORY":

```
docker images ib2_user
```

4.7 User program deployment

The following is the deployment method for the user program used in the Technology Demonstration Platform. User program shall be deployed in the following directory:

```
[Int-Ball2_platform_simulator_deployment_folder]/Int-Ball2_platform_simulator/src/user/
```

The group of files to be deployed should follow the format described below:

User programs are defined as the ROS packages. See the official procedure below for how to prepare a new package.

https://wiki.ros.org/ROS/Tutorials/CreatingPackage

The following is an example of the ROS package configuration for the Technology Demonstration Platform. In this configuration, it is assumed that the package name is "user001" and the user programs included in the package are "program001.launch" and "program002.launch". The files to be placed under "launch" must follow the template of the "roslaunch" for user programs as described below:

Table 1: Example of ROS Package Configuration for Technology Demonstration Platform

Directory		Contents
user001/ * This can be set arbitrarily.		This is a package that contains a group of programs for a specific user. It can contain multiple user programs inside. Its directory name should be the same as the package name in the ROS package definition file described below.
	launch/	Deployment directory for "roslaunch" files. * The directory name is fixed as "launch".

		program001.launch	Activation settings for user programs (nodes).
		program002.launch	Activation settings for user programs (nodes).
	package.xml		ROS package definition file. The "package name" to be defined must match the directory name "user001". See the official procedure for how to define it. http://wiki.ros.org/catkin/package.xml
	CMakeLists.txt		ROS package building settings file. See the official procedure for how to define it. http://wiki.ros.org/catkin/CMakeLists.txt
	(Others)		Arbitrary source code and various configuration files can be deployed.

4.8 "roslaunch" for user programs

A template (example of the definition) of the "roslaunch" file for the Technology Demonstration Platform is shown below.

<group ns="platform_launch"> is a mandatory item; other items can be set arbitrarily.

```xml
<?xml version="1.0"?>
<launch>

<group ns="platform_launch">
<!--
If value set to false, the target nodes will be terminated when user logic is started.
-->
<param name="sensor_fusion" value="false" />
<param name="slam_wrapper" value="false" />
<param name="ctl_only" value="true" />
<param name="fsm" value="true" />
<!--
If you want to start up camera_left and camera_right,
you need to stop (set false) slam_wrapper.
-->
<param name="camera_left" value="true" />
<param name="camera_right" value="true" />
</group>

<node name="user_template" pkg="user_template" type="user_template.py" output="screen">
<!-- The parameters to be used in the user's program can be set -->
<param name="custom_parameter_integer" value="1" />
```

```
<param name="custom_parameter_float" value="2.0" />
<param name="custom_parameter_string" value="custom" />
<param name="custom_parameter_boolean" value="true" />
</node>


</launch>
```

Specify whether to use the existing functions on the flight software side in <group ns="platform_launch">. The ROS node corresponding to the function defined as "not used" (value="false") is stopped immediately before the start of the user implementation logic.

The correspondence between the item names in <group ns="platform_launch"> and the flight software functions is as follows:

- sensor_fusion   : Sensor fusion
- slam_wrapper   : Visual SLAM
- ctl_only   : Thrust Calculation
- fsm   : Thrust Allocation

To call the newly prepared program while using GSE, add the necessary information to the "user_package_list.json" in the folder "Int-Ball2_platform_gse¥src¥ground_system¥platform_gui¥config" where the GSE configuration files are stored. (It does not search the user program automatically, so it is necessary to write the user program in the list beforehand.)

4.9 Building procedure

4.9.1 Int-Ball2 GSE

Execute "catkin_make" in the directory where "Int-Ball2_platform_gse" is deployed.

```
$ source /opt/ros/melodic/setup.bash
$ catkin_make
```

4.9.2 Int-Ball2 Technology Demonstration Platform

Execute the "catkin_make -DWITH_PCA9685=OFF" in the directory where "Int-Ball2_platform_simulator" is deployed. Note that the "-DWITH_PCA9685=OFF" is an option to build the thrust function as a simulated node to operate the inductive control function in an environment where the PWM control board is not connected.

5. Operation procedure

The procedure to execute this system is shown below.

1. Execute the following command at the terminal to activate the Int-Ball2 GSE:

```
cd [Int-Ball2 GSE deployment path] source devel/setup.bash roslaunch platform_gui bringup.launch
```

2. Execute the following command at the terminal different from 1 above to activate the Int-Ball2 Technology Demonstration Platform Simulator:

```
cd [Int-Ball2 Technology Demonstration Platform Simulator deployment path] source devel/setup.bash rosrun platform_sim_tools simulator_bringup.sh
```

3. Press the "Play" button of Gazebo to start the simulation.

4. Press the "Navigation ON" button in the "Operation Type" on the "Platform command" panel of the Int-Ball2 GSE to activate the navigation function.

5. Set User Node, User Launch File, and User Container at "User programming platform" on the "Platform command" panel of the Int-Ball2 GSE to the user programming function to be executed and press the "start" button.

   Example：

User Node	sample_tests
User Launch File	simple_test.launch
User Container	ib2_user:0.1

6. Set User Logic at "User programming platform" on the "Platform command" panel of the Int-Ball2 GSE to the user logic to be executed and press the "start" button.

7. When execution of 6 is completed, press Ctrl+C in terminals 1 and 2 to exit this system.

   (Note)
   At the ISS URDF loading, Segmentation Fault may occur when trying to load the node_1 and node_2 models. In such a case, comment out the relevant section in "iss.urdf".

6. Sample programs

(Sample programs will be prepared and added in the future.)

7. Contact Information

Japan Aerospace Exploration Agency (JAXA)

space_IVR@jaxa.jp

<u>End.</u>