

# robot\_state & robot\_command

Jun, 6th, 2023

消息类型，都是std\_msgs/String

## 1 本节点的静态参数(parameter)

### 1.1 state

robot\_state管理的参数可通过yaml设置，启动时默认使用yaml内的初值，并加载到参数服务器的robot\_state命名空间下。

yaml格式如下：

```
src > robot_state > cfg > ! robot_state_params.yaml > {} dynparam
1  parameter:
2    k1: 3
3    k2: yhh
4    k3:
5      k5: 45
6      k6: vi
7    pvm_length: 2500
8    pvm_width: 1234
9    test_ns:
10     n: 4
11     s: link
12 dynparam:
13   cmd_vel_filter: [filter_enabled, test_p]
14
```

参数服务器查看:



```
leizeng@leizeng2023:~$ rosparam list
/robot_state/k1
/robot_state/k2
/robot_state/k3/k5
/robot_state/k3/k6
/robot_state/pvm_length
/robot_state/pvm_width
/robot_state/test_ns/n
/robot_state/test_ns/s
/roscdistro
/roslaunch/uris/host_leizeng2023__38523
/rosversion
/run_id
```

信息也可通过话题robot\_state获取：

```
leizeng@leizeng2023:~$ rostopic echo /robot_state
data: {"dynparam": {"cmd_vel_filter": {"filter_enabled": null, "test_p": null}},
      "parameter": {"k1": 3, "k2": "yhh", "k3": {"k5": 45, "k6": "v
vvvvvvvvvvvvvvvvvvvvv"},
                    }, {"pvm_length": 2500, "pvm_width": 1234, "test_ns": {"n": 4, "s": "
link"}}}}
```

## 1.2 command

通过指定参数名和数值，可修改、新增参数，

具体topic指令示例如下，

```
leizeng@leizeng2023:~$ rostopic echo /robot_command
data: {"parameter": {"k1": 3, "k2": "yhh", "k3": {"k5": 45, "k6": "v
v"}},
      "v": "v"}}}
```

data内容是按级展开的json(字典)，可以同时下发多个key-value对。

```
{'parameter': {'k1': 3, 'k2': 'yhh', 'k3': {'k5': 45, 'k6': 'v1'}}}
```

## 1.3 更新机制

修改的结果会通过robot\_state更新发布一次。

修改的静态参数数值会被写入yaml文件，并作为下次启动的初始参数。

# 2. 其它节点的动态参数（dynparam）

## 2.1 state

可以将需要进行管理的动态参数写入yaml文件，不写入的动参不会被访问。

写入的格式为：client: parameter list

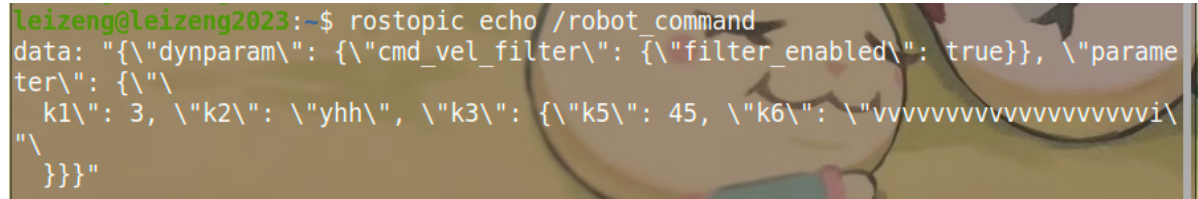
```
11 | s: link
12 | dynparam:
13 |   cmd_vel_filter: [filter_enabled, test_p]
14 |
```

启动后，将访问这些参数的数值，并通过/robot\_state进行发布，若client或对应参数不存在，则记录为None (null)

## 2.2 command

通过指定参数名和数值，可修改参数。（格式同1.2）

示例：



```
leizeng@leizeng2023:~$ rostopic echo /robot_command
data: {"dynparam": {"cmd_vel_filter": {"filter_enabled": true}}, "parameter": {"k1": 3, "k2": "yhh", "k3": {"k5": 45, "k6": "vvvvvvvvvvvvvvvvvi"}}
```

data是json格式，可以同时操作多个key

```
{'dynparam': {'cmd_vel_filter': {'filter_enabled': True}}, 'parameter': {'k1': 3, 'k2': 'yhh', 'k3': {'k5': 45, 'k6': 'vvvvvvvvvvvvvvvvvi'}}
```

## 2.3 更新机制

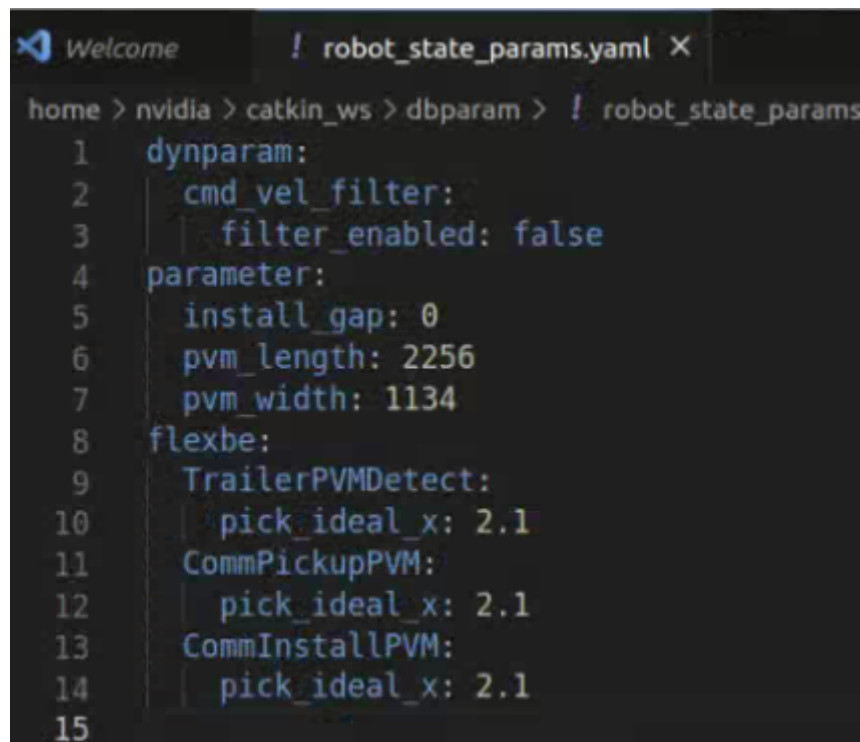
修改的结果会通过robot\_state更新发布一次。

修改的动态参数数值**不会被**写入yaml文件，每次启动的初始数值只依赖于它自己的client node。

## 3. flexbe参数管理

首先参数应该在flexbe中设置（sub关系的需要层层设置引出）

然后登记到yaml文件，并遵循以下demo格式



```
home > nvidia > catkin_ws > dbparam > ! robot_state_params
1  dynparam:
2    cmd_vel_filter:
3      filter_enabled: false
4  parameter:
5    install_gap: 0
6    pvm_length: 2256
7    pvm_width: 1134
8  flexbe:
9    TrailerPVMDetect:
10     pick_ideal_x: 2.1
11    CommPickupPVM:
12     pick_ideal_x: 2.1
13    CommInstallPVM:
14     pick_ideal_x: 2.1
15
```

flexbe参数将以topic中json的格式发布，作为web调用的信息源

```
nvidia@nvidia:~/zl_ws$ rostopic echo /robot_state
data: {"dynparam": {"cmd_vel_filter": {"filter_enabled": true}}, "flexbe": {"": {"": {"CommInstallPVM": {"pick_ideal_x": 2.1}, "CommPickupPVM": {"pick_ideal_x": 2.1}, "TrailerPVMDetect": {"pick_ideal_x": 2.1}}, "parameter": {"install_gap": 0, "pvm_length": 2256, "pvm_width": 1134}}}}
```

## 4. GIT 操作

## 4.1 自动更新

**参数：** repo\_path (仓库路径)

**操作：**先 `git reset --hard`，然后 `git pull`

接口：

收 /robot\_command ，并通过git字段控制

```
data: {"flexbe": {"beh1": {"install_gap": 12, "param3": false, "param6": "\st6\\"}, {"git": {"op": "pull"}}, "parameter": {"install_gap": 17}}}
```

```
{'flexbe': {'beh1': {'install_gap': 12, 'param6': 'st6', 'param3': False}},
'parameter': {'install_gap': 17}, 'git': {'op': 'pull'}}
```

反馈话题/trig

## 4.2 信息反馈

反馈项:

- head
- msg
- date
- branch

反馈话题/robot\_state, 示例如下:

## topic 监听示例

```

[erazem@ecs212ong002:~/catkin_ws/src/leapling_code/robot_state/launch$ rostopic echo /robot_state
data: {"cmd_vel_filter": {"filter_enabled": null, "test_p": null}, "\
\ "test": {"dyp1": null, "dyp2": null}, "flexbe": {"beh1": {"install_gap": "\
\ "17", "param3": "false", "param6": "st6"}, "beh2": {"detect": "\
\ false", "install_gap": "\ "17"}, "git": {"info": {"branch": "main", "\
\ date": "\ "20240223", "head": "\ "9dc22542a32f07fb6a79a14a3fffb4108f590d08", "\
\ msg": "\ "git feedback by trig\n\n", "parameter": {"install_gap": 17, "param3": "\
\ true, "pvm length": 2500, "pvm width": 1134}"

```

具体内容粘出来如下：

```
{'dynparam': {'cmd_vel_filter': {'filter_enabled': None, 'test_p': None}, 'test': {'dyp1': None, 'dyp2': None}}, 'flexbe': {'beh1': {'install_gap': '17', 'param3': 'false', 'param6': 'st6'}, 'beh2': {'detect': 'false', 'install_gap': '17'}}, 'parameter': {'install_gap': 17, 'param3': True, 'pvm_length': 2500, 'pvm_width': 1134}, 'git': {'info': {'head': '9dc22542a32f07fb6a79a14a3fffb4108f590d08', 'msg': 'git feedback by trig\n', 'date': '20240223', 'branch': 'main'}}
```

## 5 开关功能

---

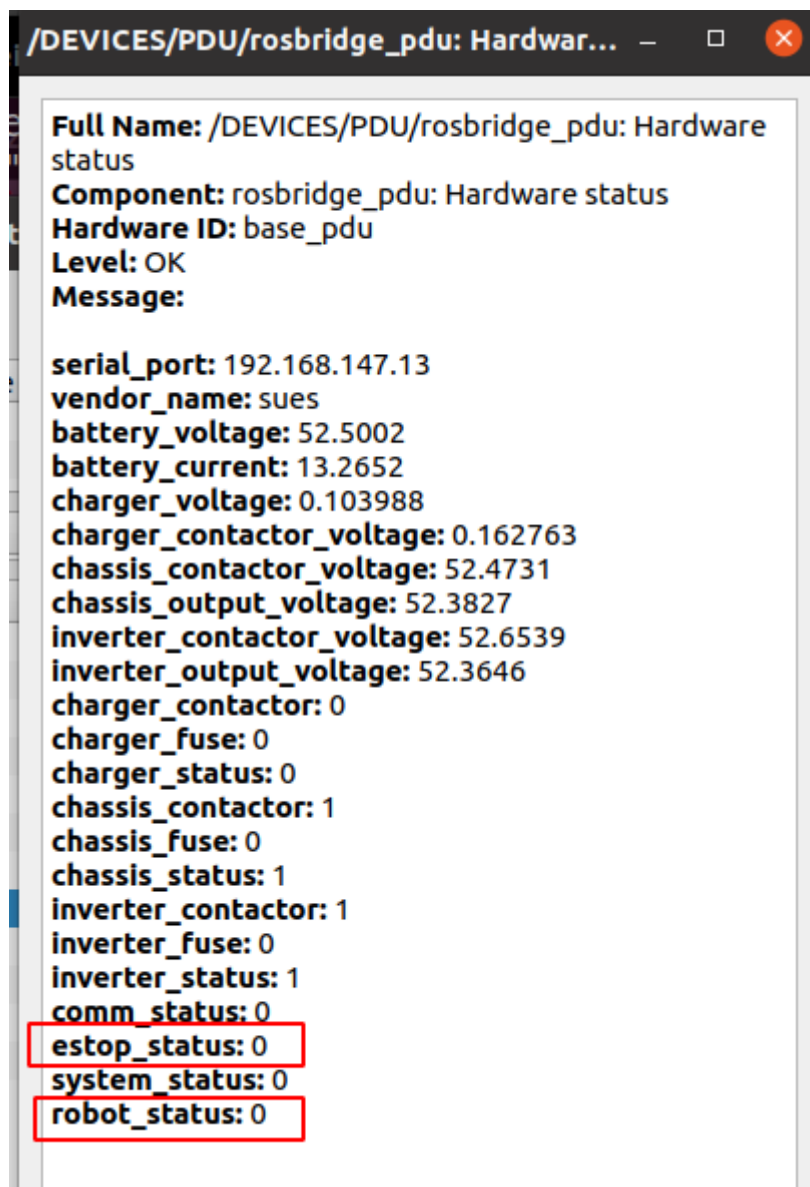
新增参数;

pdu\_launch\_path (没用)

arm\_launch\_path

接口：

弃用原来的trig。监听诊断agg信息



1. estop\_status 从 1 变 0 时：发布 inverter\_on, chassis\_on 到 pdu\_request
2. estop\_status 从 0 变 1 时：发布 inverter\_off, chassis\_off 到 pdu\_request
3. robot\_status 从 0 变 1 时：打开 arm launch
4. robot\_status 从 1 变 0 时：关闭 arm launch

一些逻辑保护：

- 重复启动不引起异常
- 重复关闭不引起异常
- 先关后开不引起异常
- 不同 launch 交替开关不引起异常