STJS (Skills Task JSON System) Documentation

Created By: Aaron Wilk - https://github.com/tubaplayerdis, Vex Team 38535B

Overview

STJS uses json instead of literal source code to develop autonomus routines. The "Tasks" are stored on the micro sd card of the vex robotics v5 brain. Major advantages of this system include:

1. No Need to Recompile
2. Data is stored Externally
3. Hot Swapping data during program execution
4. Quick Developement of autonomus routines

Basics

STJS uses milimeters as a standard for the coordinate system and distance; degrees are used for angles. STJS uses the Vex V5 GPS coordinate and degree system.

STJS uses a standard format for tasks. An Example of this would look like:

```
{
    "tasks": [
        {
            "name": "Drive To 200, 200",
            "stype": "Driving",
            "destx": 200,
            "desty": 200,
            "desth": 30,
            "tp": 100,
            "vp": 100
        },
        {
            "name": "Drive Forward",
            "stype": "Segmenting",
            "segDist": 90,
            "vp": 100
        },
        {
            "name": "Turn 90 Degrees",
            "stype": "Turning",
            "turnBy": 90,
            "tp": 100
        },
        {
            "name": "Set mogo true",
            "stype": "Mogo",
            "setTo": true
        }
    ]

}
```

Format

1. One array in the json file that is called tasks. (Name is important)
2. Objects in the array have no name

Creating Objects

Objects have 2 very important data members.

1. name - Name of the Task. Displayed on the controller when running. Characters above index 17 wont be showed
2. stype - Type of the Task. Tells the engine to what values to use when interpreting

---

stype - Driving

Use this stype when driving to a location. takes in coordinates and a heading value to turn to at the end. use -1 to ignore this behavior. Also takes in values tp and vp for turn velocity percentage and drive velocity percentage

stype Driving Specific Members

- destx - destination x value (number)
- desty - destination y value (number)
- desth - destination heading value (number)
- tp - turn velocity percentage (integer)
- vp - driving velocity percentage (integer)

stype Driving Example

```
{
    "name": "Drive To 200, 200",
    "stype": "Driving",
    "destx": 200,
    "desty": 200,
    "desth": 30,
    "tp": 100,
    "vp": 100
}
```

stype Driving Example Explanation

This Example Will:

- Set the turn velocity to 100 percent
- Set the drive velocity to 100 percent
- Drive the bot to 200,200.
- Turn the bot to 30 degrees heading

---

stype - Turning

Use this stype to turn the bot x number of degrees. Also takes in turning velocity. Posistive or negative turning is decided by the GPS system.

stype Turning Specific Members

- turnBy - turn by degrees (number)
- tp - turn velocity percentage (integer)

stype Turning Example

```
{
    "name": "Turn 90 Degrees",
    "stype": "Turning",
    "turnBy": 90,
    "tp": 100
}
```

stype Driving Example Explanation

This Example Will:

- Set the turn velocity to 100 percent
- Turn the bot 90 Degrees

---

stype - Segmenting

Use this stype to drive the bot x milimeters forward or backward

stype Segmenting Specific Members

- segDist - drive by x degrees (number)
- vp - drive velocity percentage (integer)

stype Segmenting Example

```
{
    "name": "Drive Forward",
    "stype": "Segmenting",
    "segDist": 90,
    "vp": 100
}
```

stype Segmenting Example Explanation

This Example Will:

- Set the drive velocity to 100 percent
- Drive the bot forward 90mm

---

stype - Turning

Use this stype to turn the bot x number of degrees. Also takes in turning velocity. Posistive or negative turning is decided by the GPS system.

stype Turning Specific Members

- turnBy - turn by degrees (number)
- tp - turn velocity percentage (integer)

stype Turning Example

```
{
    "name": "Turn 90 Degrees",
    "stype": "Turning",
    "turnBy": 90,
    "tp": 100
}
```

stype Driving Example Explanation

This Example Will:

- Set the turn velocity to 100 percent
- Turn the bot 90 Degrees

---

stype - Mogo

Use this stype to actuate the mogo(Mobile Goal Mechanism)

stype Mogo Specific Members

- setTo - set Mogo to value (bool)

stype Mogo Example

```
{
    "name": "Set Mogo Down",
    "stype": "Mogo",
    "setTo": true
}
```

stype Mogo Example Explanation

This Example Will:

- Set the Mogo to true

---

stype - Arm

Use this stype to control the arm used for wallstakes

stype Arm Specific Members

- armTo - turn arm to degree (Number)

stype Arm Example

```
{
    "name": "Set Arm To 130D",
    "stype": "Arm",
    "armTo": 130
}
```

stype Arm Example Explanation

This Example Will:

- Set the Arm to 130 Degrees

---

stype - EndGame

Use this stype to control the PTO(Power Transfer Unit) that transfers power from the drivetrain to the endgame lift

stype EndGame Specific Members

- endTo - set PTO to value (Number)

stype EndGame Example

```
{
    "name": "Toggle On PTO",
    "stype": "EndGame",
    "endTo": true
}
```

stype EndGame Example Explanation

This Example Will:

- Set the PTO to true

Footnotes

A interface based example of this system is being developed for `c++`. The current implementation is only present on my bot and has not been made a redistributable resource.

A Python port might be done at some point if it is necessary.