Additional information (EN version) Update: 15 September 2021

This document was created in order to provide additional information concerning the work done during this project.

IO addressing

Regarding the speed and position registers, we redirected their signals to the MRegisters (as shown in the ‘Mapping Mémoire Yaskawa’ pdf) using the functions below (Appendix A) :

- *Present Manipulator Position Output Function*
- *Function for Outputting TCP Speed to Register*
- *Function for Outputting Each Axis Speed to Register*
- *Function for Outputting Each Axis Position to Register*

In order to access these functions, you must be working on “Safety Mode”, then access the “S1CxG” option in the “parameters” menu of the teach pendant.

The addresses must be saved in the “motoman_memory.h” file to keep coherence. Finally, the robot must be rebooted and the catkin workspace compiled.

Note: These addresses may change depending on the controller. Refer to the Yaskawa Motoman documentation on IO addressing and configuration, if you are not using the YRC1000 controller.

Workspace

By default, when running the file "moveit_planning_execution.launch" of the package "motoman_hc10_moveit_config" the working area of our workshop is displayed. If you want to display your own working area, just generate a new script describing the obstacles in it and edit the launch file.

Connection with the robot

In order to communicate with the robot, you must be connected to the same network and declare the name of the controller in use (as described in §4.2 of the report). For this purpose, you must modify the default parameters that have been defined in the file "moveit_planning_execution.launch".

Running Trajectories with Moveit

Note: As described in the HC notes document (Appendix B), it is not recommended to use FSU speed limitation with ROS. Since motion will be affected and the robot will not reach it's expected end position.
If you do need to use it, be careful to properly monitor feedback position to detect if the robot trajectory is not following the expected commanded trajectory.

Trajectories with the script example

You can use the MoveIt GUI to plan and execute trajectories. Also, you can do this through a script. The execute_trajectory.py file is an example of this. ***Remember to always plan before executing any trajectory !***

Here we show different ways in which you can move the "manipulator" group.

- Using "set_named_targets": As the example script, you can plan and execute trajectories to predefined configurations (defined in the srdf file.)

```
#Planning and executing with set_named_target
print "Number of group states in srdf file: %i \n" % len(group_state)
for n in range(1,len(group_state)):
    print "group state %i: %s" %(n,group_state[n])
    print "Joint Values %s" %group.get_named_target_values(group_state[n])
    group.set_named_target(group_state[n])
    print "New target has been set"
    plan2 = group.plan()
    rospy.sleep(1)
#   print "Plannig done, now executing \n"
#   group.go(wait=True)
#   group.stop()
#   rospy.sleep(1)
```

- Setting the desired positions directly on the script:

```
#Example 1: Other ways to move your group
#You must take into account the number of joints
joint_values = group.get_current_joint_values()
#Note that the values must be in radians
joint_values[0] = 0
joint_values[1] = 0
joint_values[2] = 0
joint_values[3] = 0
joint_values[4] = -1.57
joint_values[5] = 0
group.set_joint_value_target(joint_values)
plan2 = group.plan()
#group.go(wait=True)
```

- Setting the predefined configurations on the srdf file in a different order: These trajectories will be executed one after the order.

```
#Example 2: Other ways to move your group
group.set_joint_value_target(group.get_named_target_values(group_state[2]))
plan2 = group.plan()
#group.go(wait=True)

group.set_joint_value_target(group.get_named_target_values(group_state[3]))
plan2 = group.plan()
#group.go(wait=True)

group.set_joint_value_target(group.get_named_target_values(group_state[4]))
plan2 = group.plan()
#group.go(wait=True)
```

Information Supplémentaire (Version FR) Mise à jour:15 Septembre 2021

Ce document a été créé afin de fournir des renseignements supplémentaires sur le travail effectué dans le cadre de ce projet.

Addressage E/S

En ce qui concerne les registres de vitesse et de position, nous avons redirigé leurs signaux vers les “MRegisters” (comme indiqué dans le pdf « Mapping Mémoire Yaskawa ») en utilisant les fonctions ci-dessous (Annexe A) :

- *Present Manipulator Position Output Function*
- *Function for Outputting TCP Speed to Register*
- *Function for Outputting Each Axis Speed to Register*
- *Function for Outputting Each Axis Position to Register*

Pour accéder à ces fonctions, vous devez être en « Safety Mode », puis accéder à l’option « S1CxG » dans le menu « paramètres » du boîtier de commande.

Les adresses doivent être enregistrées dans le fichier motoman_memory.h pour conserver la cohérence. Enfin, le robot doit être redémarré et le « catkin workspace » compilé.

Remarque : Ces adresses peuvent changer en fonction du contrôleur. Si vous n’utilisez pas le contrôleur YRC1000, reportez-vous à la documentation de Yaskawa Motoman sur la configuration et l’adressage E/S.

Zone de travail

Par défaut, lors de l'exécution du fichier « moveit_planning_execution.launch » du paquet « motoman_hc10_moveit_config » la zone de travail est affichée. Si vous désirez afficher votre propre zone de travail, il suffit de générer un nouveau script décrivant les obstacles placés sur cette zone et faire appel à ce fichier dans le fichier « .launch ».

Connexion avec le robot

Afin de communiquer avec le robot, il faut déclarer le nom du contrôleur et se placer sur le même réseau (Comme décrit dans le §4.2 du rapport). A cette fin, il faut modifier les paramètres par défaut qui ont été définis dans le fichier « moveit_planning_execution.launch ».

Exécution de trajectoires avec Moveit

Remarque: Tel que décrit dans le document « HC notes » (Annexe B), il n'est pas recommandé d'utiliser la limitation de vitesse FSU avec ROS. Puisque le mouvement sera affecté et le robot n'atteindra pas la position finale attendue.
Si vous avez besoin de l'utiliser, veillez à bien surveiller la position de retour pour détecter si la trajectoire du robot ne suit pas la trajectoire attendue.

Trajectoires avec un script

Vous pouvez utiliser l'interface graphique de MoveIt pour planifier et exécuter des trajectoires. Vous pouvez également le faire via un script. Le fichier execute_trajectory.py en est un exemple. ***N'oubliez pas de toujours planifier avant d'exécuter une trajectoire !***

Nous montrons ici différentes façons de déplacer le groupe « manipulateur ».

- Utiliser « set_named_targets » : Comme le fichier execute_trajectory.py, vous pouvez planifier et exécuter des trajectoires selon les configurations prédéfinies dans le fichier srdf.

```
#Planning and executing with set_named_target
print "Number of group states in srdf file: %i \n" % len(group_state)
for n in range(1,len(group_state)):
   print "group state %i: %s" %(n,group_state[n])
   print "Joint Values %s" %group.get_named_target_values(group_state[n])
   group.set_named_target(group_state[n])
   print "New target has been set"
   plan2 = group.plan()
   rospy.sleep(1)
#  print "Plannig done, now executing \n"
#  group.go(wait=True)
#  group.stop()
#  rospy.sleep(1)
```

- Définition des positions désirées directement sur le script:

```
#Example 1: Other ways to move your group
#You must take into account the number of joints
joint_values = group.get_current_joint_values()
#Note that the values must be in radians
joint_values[0] = 0
joint_values[1] = 0
joint_values[2] = 0
joint_values[3] = 0
joint_values[4] = -1.57
joint_values[5] = 0
group.set_joint_value_target(joint_values)
plan2 = group.plan()
#group.go(wait=True)
```

- Planification et exécution des configurations du fichier srdf dans un ordre différent: Les trajectoires s'exécutent l'une après l'autre.

```
#Example 2: Other ways to move your group
group.set_joint_value_target(group.get_named_target_values(group_state[2]))
plan2 = group.plan()
#group.go(wait=True)

group.set_joint_value_target(group.get_named_target_values(group_state[3]))
plan2 = group.plan()
#group.go(wait=True)

group.set_joint_value_target(group.get_named_target_values(group_state[4]))
plan2 = group.plan()
#group.go(wait=True)
```

Appendix A / Annexe A

6.12 Present Manipulator Position Output Function

6.12.1 Function for Outputting Present Cartesian Position of Manipulator to Register

6.12.1.1 Outline

The present Cartesian position of the manipulator (values in the base coordinates) is output to the specified registers.

6.12.1.2 Parameters

The following parameters specify the details of the function and output register numbers.

S1CxG	Description
208	Enables/Disables the function for outputting the present Cartesian position (in the base coordinates) to registers. (command value) 0: disable 1: enable
209	Specifies the output size to the register. 0: output in 2 bytes 1: output in 4 bytes
210	Cartesian position (command value) X register number of output destination
211	Cartesian position (command value) Y register number of output destination
212	Cartesian position (command value) Z register number of output destination
213	Cartesian position (command value) Rx register number of output destination
214	Cartesian position (command value) Ry register number of output destination
215	Cartesian position (command value) Rz register number of output destination
216	Cartesian position (command value) Re register number of output destination
217	Enables/Disables the function for outputting the present Cartesian position (in the base coordinates) to registers. (FB value) 0: disable 1: enable
218	Specifies the output size to the register. 0: output in 2 bytes 1: output in 4 bytes
219	Cartesian position (FB value) X register number of output destination
220	Cartesian position (FB value) Y register number of output destination
221	Cartesian position (FB value) Z register number of output destination
222	Cartesian position (FB value) Rx register number of output destination
223	Cartesian position (FB value) Ry register number of output destination
224	Cartesian position (FB value) Rz register number of output destination
224	Cartesian position (FB value) Re register number of output destination

<Example 1>

S1C1G	Setting value
208	1
209	0
210	10
211	11
212	12
213	13
214	14
215	15
216	16

When the parameters are set as shown in the above table, the present position is output to the registers as follows:

M010 = Manipulator's present Cartesian position (command value) X [unit: mm]
M011 = Manipulator's present Cartesian position (command value) Y [unit: mm]
M012 = Manipulator's present Cartesian position (command value) Z [unit: mm]
M013 = Manipulator's present Cartesian position (command value) Rx [unit: deg]
M014 = Manipulator's present Cartesian position (command value) Ry [unit: deg]
M015 = Manipulator's present Cartesian position (command value) Rz [unit: deg]
M016 = Manipulator's present Cartesian position (command value) Re [unit: deg]

<Example 2>

S1C1G	Setting value
217	1
218	1
219	10
220	12
221	14
222	16
223	18
224	20
225	22

When the parameters are set as shown in the above table, the present position is output to the registers as follows:

M010 = Lower 2 bytes of the manipulator's present Cartesian position (FB value) X [unit: µm]
M011 = Upper 2 bytes of the manipulator's present Cartesian position (FB value) X [unit: µm]
M012 = Lower 2 bytes of the manipulator's present Cartesian position (FB value) Y [unit: µm]
M013 = Upper 2 bytes of the manipulator's present Cartesian position (FB value) Y [unit: µm]
M014 = Lower 2 bytes of the manipulator's present Cartesian position (FB value) Z [unit: µm]
M015 = Upper 2 bytes of the manipulator's present Cartesian position (FB value) Z [unit: µm]
M016 = Lower 2 bytes of the manipulator's present Cartesian position (FB value) Rx [unit: 0.001 deg]
M017 = Upper 2 bytes of the manipulator's present Cartesian position (FB value) Rx [unit: 0.001 deg]

M018 =	Lower 2 bytes of the	manipulator's present Cartesian position (FB value)	Ry	[unit: 0.001 deg]
M019 =	Upper 2 bytes of the	manipulator's present Cartesian position (FB value)	Ry	[unit: 0.001 deg]
M020 =	Lower 2 bytes of the	manipulator's present Cartesian position (FB value)	Rz	[unit: 0.001 deg]
M021 =	Upper 2 bytes of the	manipulator's present Cartesian position (FB value)	Rz	[unit: 0.001 deg]
M022 =	Lower 2 bytes of the	manipulator's present Cartesian position (FB value)	Re	[unit: 0.001 deg]
M023 =	Upper 2 bytes of the	manipulator's present Cartesian position (FB value)	Re	[unit: 0.001 deg]

NOTE

- When this function for command values is enabled (S1CxG208=1), be sure to set the register number of output destination for each coordinate value (S1CxG210 to 216).
- When this function for FB values is enabled (S1CxG217=1), be sure to set the register number of output destination for each coordinate value (S1CxG219 to 225).
- When the output size to the register is set to 2 bytes (S1CxG209=0 or S1CxG218=0), the unit for X, Y, Z coordinate values is "mm", and the unit for Rx, Ry, Rz, Re coordinate values is "deg". If the coordinate value exceeds 2 bytes, only the lower 2 bytes will be output.
- When the output size to the register is set to 4 bytes (S1CxG209=1 or S1CxG218=1), the unit for X, Y, Z coordinate values is "μmm", and the unit for Rx, Ry, Rz, Re coordinate values is "0.0001 deg".
- When the output size to the register is set to 4 bytes (S1CxG209=1 or S1CxG218=1), the upper bytes of the coordinate value will be output to the next number of the specified register number. Before performing setting, check the usage status of the registers.

6.12.2 Function for Outputting Present Pulse Position to Register

6.12.2.1 Outline

The present position of the robot axis, the base axis, or the station axis in pulses is output to the specified registers.

6.12.2.2 Parameters

The following parameters specify the details of the function and output register numbers.

S1CxG	Description
202	Specifies the axis to apply the function 1 for outputting the present position in pulses to registers. (command value) The axis is specified in bits. Bit OFF: disable Bit ON: enable
203	Specifies the axis to apply the function 1 for outputting the present position in pulses to registers. (FB value) The axis is specified in bits. Bit OFF: disable Bit ON: enable
204	Specifies the output size to the register. Bit OFF: output in 2 bytes Bit ON: output in 4 bytes
205	Specifies the axis to apply the function 2 for outputting the present position in pulses to registers. (command value) The axis is specified in bits. Bit OFF: disable Bit ON: enable
206	Specifies the axis to apply the function 2 for outputting the present position in pulses to registers. (FB value) The axis is specified in bits. Bit OFF: disable Bit ON: enable
207	Specifies the output size to the register. Bit OFF: output in 2 bytes Bit ON: output in 4 bytes
1090 to 1097	Function 1 register number of output destination
1100 to 1107	Function 1 resolution setting
1110 to 1117	Function 1 offset value setting
1120 to 1127	Function 2 register number of output destination
1130 to 1137	Function 2 resolution setting
1140 to 1147	Function 2 offset value setting

2-byte output specification:

- (specified M register) = (pulse (command or FB)) / (resolution) + (offset value) [unit: pulse]

NOTE

When the output size to the register is set to 2 bytes (no axis specified in S1CxG204 or S1CxG207), the pulse value in the size of 2 bytes will be output to the specified register number. If the size exceeds 2 bytes, only the lower 2 bytes will be output.

4-byte output specification:

- (specified M register) = Lower 2 bytes of {(pulse (command or FB)) / (resolution) + (offset value)} [unit: pulse]
- (specified M register + 1) = Upper 2 bytes of {(pulse (command or FB)) / (resolution) + (offset value)} [unit: pulse]

NOTE

When the output size to the register is set to 4 bytes (an axis specified in S1CxG204 or S1CxG207), the lower 2 bytes will be output to the specified register number, and the upper 2 bytes will be output to the next number of the specified register number. Before performing setting, check the usage status of the registers.

<Example 1>

S1C1G	Setting value
202	63
203	0
204	0
1090	10
1091	11
1092	12
1093	13
1094	14
1095	15

When the parameters are set as shown in the above table, the present position is output to the registers as follows:

M010	= Present pulse position (command value)	S (1st axis)	[unit: pulse]
M011	= Present pulse position (command value)	L (2nd axis)	[unit: pulse]
M012	= Present pulse position (command value)	U (3rd axis)	[unit: pulse]
M013	= Present pulse position (command value)	R (4th axis)	[unit: pulse]
M014	= Present pulse position (command value)	B (5th axis)	[unit: pulse]
M015	= Present pulse position (command value)	T (6th axis)	[unit: pulse]

<Example 2>

S1C1G	Setting value
202	0
203	63
204	63
1090	10
1091	12
1092	14
1093	16
1094	18
1095	20

When the parameters are set as shown in the above table, the present position is output to the registers as follows:

M010 = Lower 2 bytes of the Present pulse position (FB value) S (1st axis) [unit: pulse]
M011 = Upper 2 bytes of the Present pulse position (FB value) S (1st axis) [unit: pulse]
M012 = Lower 2 bytes of the Present pulse position (FB value) L (2nd axis) [unit: pulse]
M013 = Upper 2 bytes of the Present pulse position (FB value) L (2nd axis) [unit: pulse]
M014 = Lower 2 bytes of the Present pulse position (FB value) U (3rd axis) [unit: pulse]
M015 = Upper 2 bytes of the Present pulse position (FB value) U (3rd axis) [unit: pulse]
M016 = Lower 2 bytes of the Present pulse position (FB value) R (4th axis) [unit: pulse]
M017 = Upper 2 bytes of the Present pulse position (FB value) R (4th axis) [unit: pulse]
M018 = Lower 2 bytes of the Present pulse position (FB value) B (5th axis) [unit: pulse]
M019 = Upper 2 bytes of the Present pulse position (FB value) B (5th axis) [unit: pulse]
M020 = Lower 2 bytes of the Present pulse position (FB value) T (6th axis) [unit: pulse]
M021 = Upper 2 bytes of the Present pulse position (FB value) T (6th axis) [unit: pulse]

NOTE

- If the pulse is a negative value, the pulse will be output to the register in 2's complement notation.
- Even in one control group, "command value" or "FB value" can be specified differently for each axis. However, if "command value" and "FB value" are specified for the same axis, the value "0" will be output to the register.
- If "0" is set as the resolution setting parameter (S1CxG1110 to 1117, S1CxG1130 to 1137), it will be treated as "1" when output to the register is performed.
- If "0" is set as the register number of output destination (S1CxG1090 to 1097, S1CxG1120 to 1127), the present pulse position will not be output to the register. Thus, no value can be output to the register number M000. Also, if the same register number of output destination is used more than twice, the former data will be overwritten by the latter data.

6.12.3 Function for Outputting TCP Speed to Register

6.12.3.1 Outline

The TCP (tool center point) speed of the manipulator is output to the specified registers.

6.12.3.2 Parameters

The following parameters specify the details of the function and output register numbers.

S1CxG	Description
330	Enables/Disables the function for outputting the TCP speed to registers. (command value) 0: no output to register 1: output in 2 bytes [unit: mm/sec] 2: output in 4 bytes [unit: µm/sec]
331	TCP speed (command value) register number of output destination
332	Enables/Disables the function for outputting the TCP speed to registers. (FB value) 0: no output to register 1: output in 2 bytes [unit: mm/sec] 2: output in 4 bytes [unit: µm/sec]
333	TCP speed (FB value) register number of output destination

<Example 1>

S1C1G	Setting value
330	1
331	10
332	2
333	11

When the parameters are set as shown in the above table, the speed is output to the registers as follows:

M010 =	TCP speed (command value)	[unit: mm/sec]
M011 =	Lower 2 bytes of the TCP speed (FB value)	[unit: µm/sec]
M012 =	Upper 2 bytes of the TCP speed (FB value)	[unit: µm/sec]

NOTE

- When the output size to the register is set to 2 bytes (“1” is set in S1CxG330 or S1CxG332), the TCP speed in the size of 2 bytes will be output to the specified register number. If the size exceeds 2 bytes, only the lower 2 bytes will be output.
- When the output size to the register is set to 4 bytes (“2” is set in S1CxG330 or S1CxG332), the lower 2 bytes will be output to the specified register number, and the upper 2 bytes will be output to the next number of the specified register number. Before performing setting, check the usage status of the registers.
- If “0” is set as the register number of output destination (S1CxG331 or S1CxG333), the present TCP speed will not be output to the register. Thus, no value can be output to the register number M000. Also, if the same register number of output destination is used more than twice, the former data will be overwritten by the latter data.

6.12.4 Function for Outputting Each Axis Speed to Register

6.12.4.1 Outline

The speed of each axis of the robot axis, the base axis, and the station axis is output to the specified registers.

6.12.4.2 Parameters

The following parameters specify the details of the function and output register numbers.

S1CxG	Description
334	Enables/Disables the function for outputting each axis speed to registers. (command value) 0: no output to register 1: output in 2 bytes [unit: deg/sec (mm/sec for a linear motion axis)] 2: output in 4 bytes [unit: 0.0001 deg/sec (μm/sec for a linear motion axis)]
335	Enables/Disables the function for outputting each axis speed to registers. (FB value) 0: no output to register 1: output in 2 bytes [unit: deg/sec (mm/sec for a linear motion axis)] 2: output in 4 bytes [unit: 0.0001 deg/sec (μm/sec for a linear motion axis)]
1270 to 1277	Each axis speed (command value) register number of output destination
1280 to 1287	Each axis speed (FB value) register number of output destination

<Example 1>

S1C1G	Setting value
334	1
335	0
1270	10
1271	11
1272	12
1273	13
1274	14
1275	15

When the parameters are set as shown in the above table, the speed is output to the registers as follows:

M010	= Each axis speed (command value)	S (1st axis)	[unit: deg/sec]
M011	= Each axis speed (command value)	L (2nd axis)	[unit: deg/sec]
M012	= Each axis speed (command value)	U (3rd axis)	[unit: deg/sec]
M013	= Each axis speed (command value)	R (4th axis)	[unit: deg/sec]
M014	= Each axis speed (command value)	B (5th axis)	[unit: deg/sec]
M015	= Each axis speed (command value)	T (6th axis)	[unit: deg/sec]

<Example 2>

S1C1G	Setting value
334	0
335	2
1280	10
1281	12
1282	14
1283	16
1284	18
1285	20

When the parameters are set as shown in the above table, the speed is output to the registers as follows:

M010	= Lower 2 bytes of the	Each axis speed (FB value)	S (1st axis)	[unit: 0.0001 deg/sec]
M011	= Upper 2 bytes of the	Each axis speed (FB value)	S (1st axis)	[unit: 0.0001 deg/sec]
M012	= Lower 2 bytes of the	Each axis speed (FB value)	L (2nd axis)	[unit: 0.0001 deg/sec]
M013	= Upper 2 bytes of the	Each axis speed (FB value)	L (2nd axis)	[unit: 0.0001 deg/sec]
M014	= Lower 2 bytes of the	Each axis speed (FB value)	U (3rd axis)	[unit: 0.0001 deg/sec]
M015	= Upper 2 bytes of the	Each axis speed (FB value)	U (3rd axis)	[unit: 0.0001 deg/sec]
M016	= Lower 2 bytes of the	Each axis speed (FB value)	R (4th axis)	[unit: 0.0001 deg/sec]
M017	= Upper 2 bytes of the	Each axis speed (FB value)	R (4th axis)	[unit: 0.0001 deg/sec]
M018	= Lower 2 bytes of the	Each axis speed (FB value)	B (5th axis)	[unit: 0.0001 deg/sec]
M019	= Upper 2 bytes of the	Each axis speed (FB value)	B (5th axis)	[unit: 0.0001 deg/sec]
M020	= Lower 2 bytes of the	Each axis speed (FB value)	T (6th axis)	[unit: 0.0001 deg/sec]
M021	= Upper 2 bytes of the	Each axis speed (FB value)	T (6th axis)	[unit: 0.0001 deg/sec]

NOTE

- When the output size to the register is set to 2 bytes (“1” is set in S1CxG334 or S1CxG335), the axis speed in the size of 2 bytes will be output to the specified register number. If the size exceeds 2 bytes, only the lower 2 bytes will be output.
- When the output size to the register is set to 4 bytes (“2” is set in S1CxG334 or S1CxG335), the lower 2 bytes will be output to the specified register number, and the upper 2 bytes will be output to the next number of the specified register number. Before performing setting, check the usage status of the registers.
- If “0” is set as the register number of output destination (S1CxG1270 to 1277, S1CxG1280 to 1287), the present axis speed will not be output to the register. Thus, no value can be output to the register number M000. Also, if the same register number of output destination is used more than twice, the former data will be overwritten by the latter data.

6.12.5 Function for Outputting Each Axis Position to Register

6.12.5.1 Outline

The position of each axis of the robot axis, the base axis, and the station axis is output to the specified registers.

6.12.5.2 Parameters

The following parameters specify the details of the function and output register numbers.

S1CxG	Description
336	Enables/Disables the function for outputting each axis position to registers. (command value) 0: no output to register 1: output in 2 bytes [unit: deg (mm for a linear motion axis)] 2: output in 4 bytes [unit: 0.0001 deg (μm for a linear motion axis)]
337	Enables/Disables the function for outputting each axis position to registers. (FB value) 0: no output to register 1: output in 2 bytes [unit: deg (mm for a linear motion axis)] 2: output in 4 bytes [unit: 0.0001 deg (μm for a linear motion axis)]
1290 to 1297	Each axis position (command value) register number of output destination
1300 to 1307	Each axis position (FB value) register number of output destination

<Example 1>

S1C1G	Setting value
336	1
337	0
1290	10
1291	11
1292	12
1293	13
1294	14
1295	15

When the parameters are set as shown in the above table, the position is output to the registers as follows:

M010	= Each axis position (command value)	S (1st axis)	[unit: deg]
M011	= Each axis position (command value)	L (2nd axis)	[unit: deg]
M012	= Each axis position (command value)	U (3rd axis)	[unit: deg]
M013	= Each axis position (command value)	R (4th axis)	[unit: deg]
M014	= Each axis position (command value)	B (5th axis)	[unit: deg]
M015	= Each axis position (command value)	T (6th axis)	[unit: deg]

<Example 2>

S1C1G	Setting value
336	0
337	2
1300	10
1301	12
1302	14
1303	16
1304	18
1305	20

When the parameters are set as shown in the above table, the position is output to the registers as follows:

M010	= Lower 2 bytes of the	Each axis position (FB value)	S (1st axis)	[unit: 0.0001 deg]
M011	= Upper 2 bytes of the	Each axis position (FB value)	S (1st axis)	[unit: 0.0001 deg]
M012	= Lower 2 bytes of the	Each axis position (FB value)	L (2nd axis)	[unit: 0.0001 deg]
M013	= Upper 2 bytes of the	Each axis position (FB value)	L (2nd axis)	[unit: 0.0001 deg]
M014	= Lower 2 bytes of the	Each axis position (FB value)	U (3rd axis)	[unit: 0.0001 deg]
M015	= Upper 2 bytes of the	Each axis position (FB value)	U (3rd axis)	[unit: 0.0001 deg]
M016	= Lower 2 bytes of the	Each axis position (FB value)	R (4th axis)	[unit: 0.0001 deg]
M017	= Upper 2 bytes of the	Each axis position (FB value)	R (4th axis)	[unit: 0.0001 deg]
M018	= Lower 2 bytes of the	Each axis position (FB value)	B (5th axis)	[unit: 0.0001 deg]
M019	= Upper 2 bytes of the	Each axis position (FB value)	B (5th axis)	[unit: 0.0001 deg]
M020	= Lower 2 bytes of the	Each axis position (FB value)	T (6th axis)	[unit: 0.0001 deg]
M021	= Upper 2 bytes of the	Each axis position (FB value)	T (6th axis)	[unit: 0.0001 deg]

NOTE

- If the axis position is a negative value, the axis position will be output to the register in 2's complement notation.
- When the output size to the register is set to 2 bytes ("1" is set in S1CxG336 or S1CxG337), the axis position in the size of 2 bytes will be output to the specified register number. If the size exceeds 2 bytes, only the lower 2 bytes will be output.
- When the output size to the register is set to 4 bytes ("2" is set in S1CxG336 or S1CxG337), the lower 2 bytes will be output to the specified register number, and the upper 2 bytes will be output to the next number of the specified register number. Before performing setting, check the usage status of the registers.
- If "0" is set as the register number of output destination (S1CxG1290 to 1297, S1CxG1300 to 1307), the present axis position will not be output to the register. Thus, no value can be output to the register number M000. Also, if the same register number of output destination is used more than twice, the former data will be overwritten by the latter data.

Appendix B / Annexe B

(http://wiki.ros.org/motoman_driver?action=AttachFile&do=view&target=HC+Notes.pdf)

Note concerning the MotoROS with HC robots

Update: 2019-November-21

WARNING!!!

Even though the motion of a Human Collaborative robot using the MotoRos driver is possible with the current driver, IT DOES NOT MAKE THE ROBOT INTRINSICALLY SAFE for all operational conditions. Proper risk assessment must be conducted following the current industrial standards. Refer to the proper documentation for PFL settings and usage.

Compatibility Check

Yaskawa Human Collaborative robots (HC) make use of the PFL (Power and Force Limiting) function. For safety reasons, the use of MotoROS with Human Collaborative robots was prevented in early controller software versions until proper safety testing was performed. In those versions, the MotoROS driver generates a systematic alarm and is prevented from running.

The latest MotoROS driver checks controller software to ensure it is version YAS2.80 or newer on YRC1000. For the YRC1000micro, YBS2.30 or newer is required. The MotoROS driver is only prevented from running if the software is older than the required version.

PFL Interaction

When the robot detects an unexpected contact (with a human or object), PFL causes the current motion to be cancelled and if clamping is detected the robot will move slightly in the opposite direction to release pressure. Further motion from ROS will not be accepted until the PFL sequence is completed and the ROS motion is restarted.

Following the activation of the PFL, further motion command sent by ROS will be rejected with a return value of ROS_RESULT_NOT_READY (5). A new subcode ROS_RESULT_NOT_READY_PFL_ACTIVE (5011) was added. Depending of the timing in the sequence, the return might be the new code or one of the existing subcode such as ROS_RESULT_NOT_READY_NOT_STARTED or ROS_RESULT_NOT_READY_SKILLSEND.

Tool Change Simple Message

The PFL function is highly dependent on proper tool definition and selection. In the case where the robot is handling parts, it may be required to change tool selection to properly reflect the current load on the robot. Until now the MotoROS driver was limited to the use of a single tool, Tool#0. To increase

flexibility, a new Yaskawa/Motoman specific simple message was added to select tool. **The implementation was done for the MotoROS driver side, but it is left to the ROS user to implement the necessary message on the ROS side.** Below are the new message specifications:

Message Type:

```
ROS_MSG_MOTO_SELECT_TOOL = 2018
```

Structure:

Standard header plus:

```
struct SmBodySelectTool
{
	// Robot/group ID;  0 = 1st robot
	int groupNo;

	// Tool no (0 to 63) for the selected group
	int tool;

	// Optional message tracking number that will be echoed back in the response.
	int sequence;
}
```

The message is sent to the MotionServer. The MotionServer will return a message of type:

```
ROS_MSG_MOTO_MOTION_REPLY = 2002
```

Return values can be:

```
ROS_RESULT_SUCCESS = 0

ROS_RESULT_INVALID = 3; ROS_RESULT_INVALID_GROUPNO = 3004

ROS_RESULT_INVALID = 3; ROS_RESULT_INVALID_DATA_TOOLNO = 3017
```

Once a tool is selected, the selected tool is used for all following motion until a new selection command is send or the controller is rebooted. **Every time the controller is rebooted, the selected tool will default to 0 for all control group.** Only robot group can be assigned tools.

FSU Interaction

The FSU (Functional Safety Unit) works independently from the controller, therefore all FSU function will work properly whether MotoROS is used or not. When activated, there is no explicit feedback to ROS. In the case that alarms are generated by the FSU, the ROS side will be notified of the alarm state like any other alarm. The alarm will need to be reset, and motion reinitialized from the current position.

The issue is when a function such as Speed Limit is enabled, any motion sent by ROS at speed higher than the speed limit will be reduced. This will affect the robot motion and the robot will not reach it's expected end position. **The only feedback to ROS is the standard position-monitor stream**. There is no

explicit feedback that the speed was dynamically reduced. This is a limitation that we are aware of. So it is not recommended to use FSU speed limitation with ROS. If you do need to use it, be careful to properly monitor feedback position to detect if the robot trajectory is not following the expected commanded trajectory.

Conveyor Tracking

The HC series of robots is not compatible with standard Yaskawa Conveyor Tracking function. If Conveyor Tracking is enabled, then MotoROS will alarm and fail during initialization.