YASKAWA
PARTNER SUPPORT
Shared Integration Experience

This document captures ideas, experiences, and informal recommendations from the Yaskawa Partner Support team. It is meant to augment – not supersede manuals or documentation from motoman.com. Please contact the Partner Support team at partnersupport@motoman.com for updates or clarification.

Auto Home Recovery

Introduction

People sometimes assume that since a robot made its way to a position, the robot remembers how it got there and can easily return home. This is not the case. The robot is taught how to get somewhere from its starting position (Home) but must also be *taught* how to return to its Home position. Even if the robot did remember its path and could just reverse it to get back Home, many times things have changed along its journey. For example, gripper clamped/unclamped, parts welded together, fixture clamps moved, etc., that make it impossible to return Home the way it came.

Due to robotic systems being different, sometimes an automated return-to home, "AutoHome" is easy to develop. Other times it is not so easy, seemingly impossible, can take many hours to develop, and still not be 100% reliable. However sometimes, AutoHome must still be done. This is because it can reduce downtime by allowing the operator to simply press a button prompting the robot to return to the Home Position. The operator can then fix any issues within the cell, then restart the system with minimal training and time lost.

This document presents two different approaches to solve the AutoHome problem. Testing and implementation can still be time consuming and not 100% reliable, due to the multitude of scenarios that must be accounted for. No matter how hard the programmer tries some scenarios may be missed. Or sometimes the system is modified and the needed changes to the AutoHome routine are overlooked. These mistakes usually result in a crash.

Auto Home Method 1

This document explains how the job works and setup procedures that need to be completed before using job (including possible Job edits)

Each highlighted section refers to different axis yellow-X, green-Y, and blue-Z. User may not need all or multiple axes in job (unnecessary axes can be removed from job). The application and surrounding equipment should identify which axes are needed for the job. You may need multiple jobs or replace the range values and/or SETE value to variables to make the job versatile.

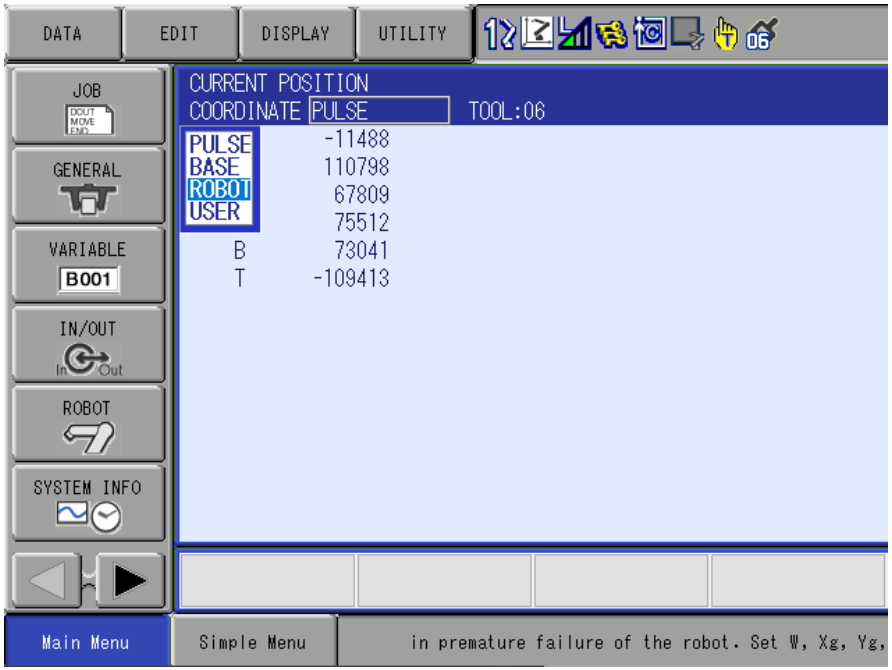
- In many applications, a single axis of motion could clear the robot from area. Remove unnecessary axis section(s) from job.
- In some applications, multiple single axis (x, y, and/or z) moves may be required to clear robot. If this is the case, a rearrangement of the order (x, y, and/or z) may be required to move robot safely. A MOVL LP000 instruction will need to be added after each SETE instruction to preform single axis motion.

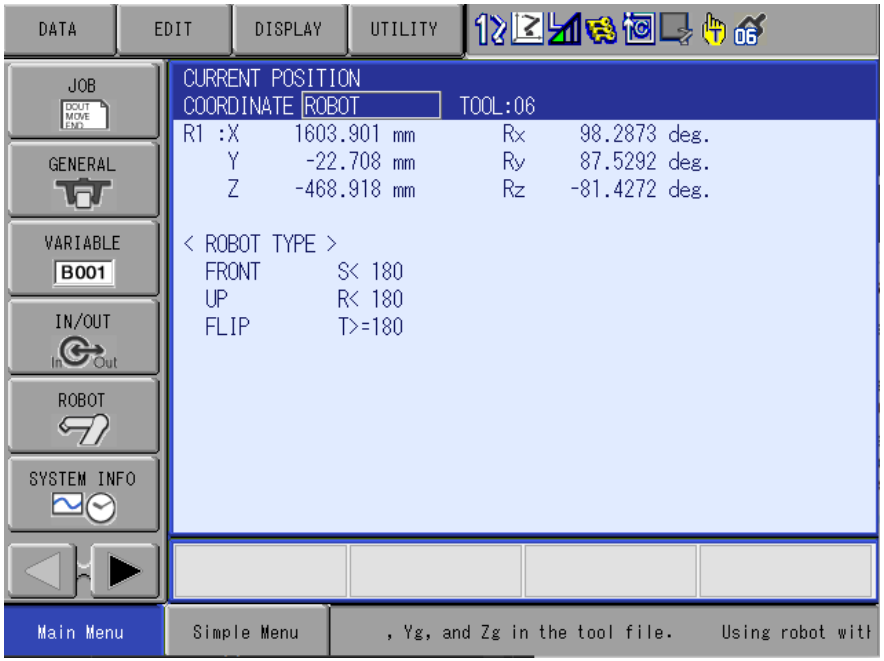
Setup required:

If equipment is not parallel or perpendicular to robot, user frame may be needed to simplify axis motion. CNVRT instruction will be needed in job below line 005. Changing LP000 from robot frame to UF. Instruction should read as follows: CNVRT LPX000 LPX000 UF#(x). Replace x with User frame setup for equipment. A variable can be used if the job needed to assign UF.

User will need to position robot in “safe” location using correct tool file. Review robot current position in robot frame or assigned user frame for equipment. To view Current position on standard pendant, select CURRENT POSITION in ROBOT section.

Change coordinate to robot or specific user frame being used.





Make note of required axes to relocate robot (x, Y, and/or z). The axis location will be needed to reference range and SETE instruction value(s).

The IFTHEN instruction values will need to be edited to accommodate for robot range (+/-) to determine location for new motion. SETE instruction will need to be edited with value of axis in “safe” position. User can change range to single value is desired. The new instruction would read: IFTHENEXP LDxxx<>100000 instead of IFTHENEXP LDxxx<100000. Xxx to be replaced with correct variable for axis.

The values of IFTHEN, GETE, and SETE instructions are in microns(100mm=100000). D variables can be used to set range and SETE instructions. Using variable still requires values of D variable in microns(100mm=100000)

The final position can be taught point or variable to get robot to home or pounce position. This position should outside/away from equipment.

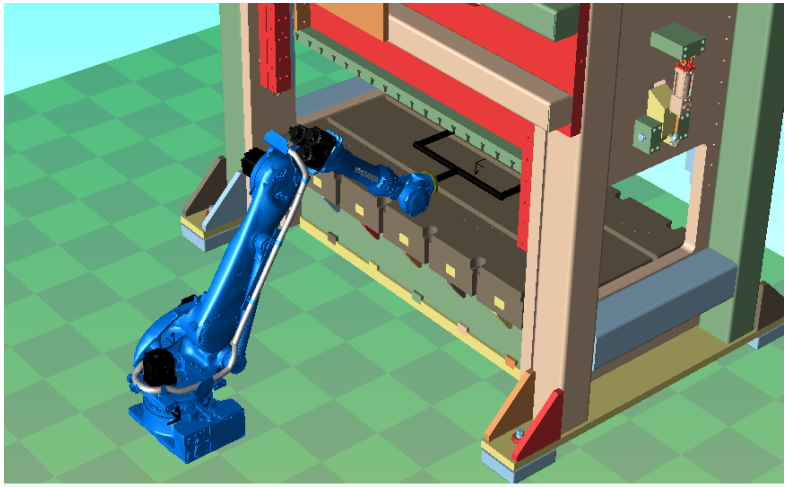
Recover_Home job contents:

```
000     NOP
001     'Safely get the robot back home
002     ' get robot frame current pos
003     GETS LPX000 $PX001
004     ' get robot current X pos
005     GETE LD001 LP000 (1)
006     ' confirm robot X position
007     IFTHENEXP LD001<100000
008             ' set robot X
009              SETE LP000 (1) 100000
010     ENDIF
011     ' get robot current Y pos
012     GETE LD002 LP000 (2)
013     ' confirm robot Y
014     IFTHENEXP LD002<100000
015             ' set robot Y
016              SETE LP000 (2) 100000
017     ENDIF
018     ' get robot current Z pos
019     GETE LD003 LP000 (3)
020     ' confirm robot height
021     IFTHENEXP LD003<100000
022             ' set robot height
023              SETE LP000 (3) 100000
024     ENDIF
025     ' move to clear position
026     MOVL LP000 V=150.0
027     '
028     'Move to Home
```

```
029     MOVJ P000 VJ=10.00
030     END
```

Example of use

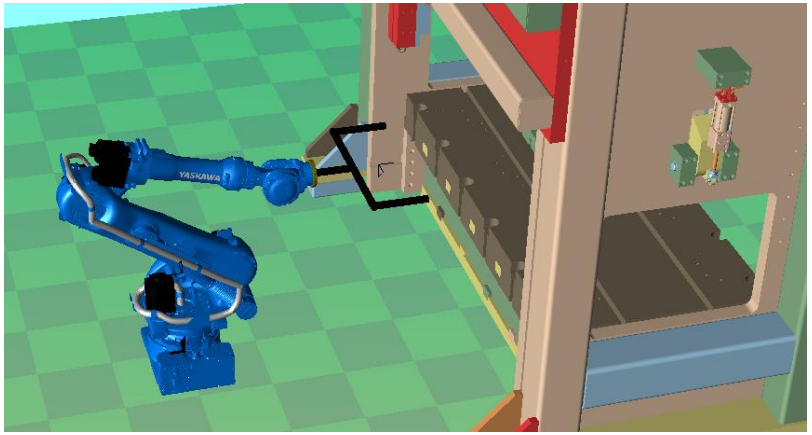
Scenario: Robot is in press and must be removed safely



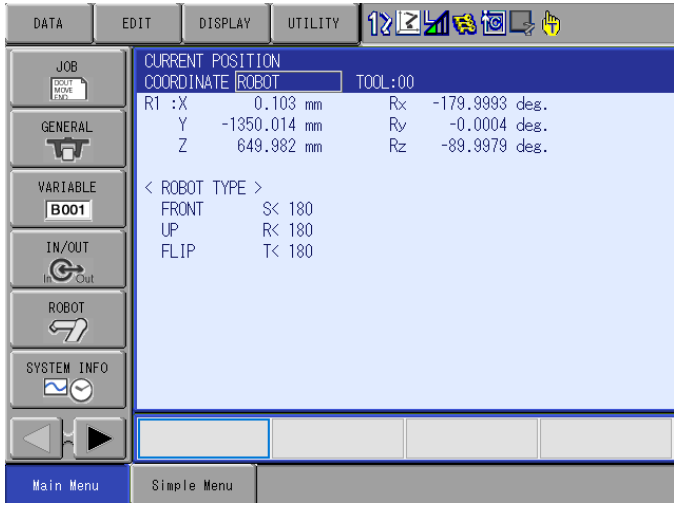
Setup Procedure

1. Set TCP for tooling as needed for application. (NOTE: Incorrect TCP will give incorrect position of robot, leaving opportunity for crash.)

2. Jog robot to “safe” position outside of equipment with correct TCP.



3. Log robot current robot frame position from controller. This information will be used to edit the job.



4. In current scenario, robot will need to first move up then move out of press.

5. Edited job should be as follows:

```
000     NOP
001     'Safely get the robot back home
002     ' get robot current pos
003     GETS LPX000 $PX001
004     ' get robot frame current Z pos
005     GETE LD001 LP000 (3)
006     ' confirm robot Z position
007     IFTHENEXP LD003<650000
008             ' set robot Z
009             SETE LP000 (3) 650000

010             ' move to clear position
011             MOVL LP000 V=150.0
012     ENDIF
013     ' get robot current Y pos
014     GETE LD002 LP000 (2)
015     ' confirm robot Y
016     IFTHENEXP LD002<-1350000
017             ' set robot Y
018              SETE LP000 (2) -1350000
019             ' move to clear position
020             MOVL LP000 V=150.0
021     ENDIF
022     ' move to clear position
023     MOVL LP000 V=150.0
024     '
```

```
025     'Move to Home
026     MOVJ P000 VJ=10.00
027     END
```

Review of changes in scenario

1. Moved Z section to beginning of job to move robot up first. (lines 006-012)
2. Edited line 007 with Z value from “safe’ position Z 649.982 (rounded value to 650000 equaling 650.000mm)
3. Edited line 009 to same value of 650000
4. Add line 011 MOVL LP000 V=150.0 to move robot to designated height.
5. Remove X section. (X is not required in scenario)
6. Edit line 016 value from ‘safe” position Y -1350000(-1350.000mm)
7. Edited line 018 to same value -1350000(-1350.000mm)
8. Add line 020 MOVL LP000 V=150.0 to move robot out of press.
9. Line 023 is redundant motion and can be removed, if desired.
10. Line 026 moves robot to predefined “safe” or pounce position.

Auto Home Method 2

This "cube-based" method was taken from an existing project, so the examples have been used but will need modified for the system it is to be used on.

Requirements include but are not limited to:

- Knowledge of using Cubes.
- Concurrent I/O ladder changes.
- Interrupt Job.
- A lot of testing.
- A lot of time.

The AUTOHOME Job

An AUTOHOME job needs to be created.

```
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'$       AUTOHOME JOB         $
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*TOP
'********DATA VALIDATION*********
'autohome if robot not jogged
JUMP *OUT IN#(1985)=ON
'check robot is in a safe
'   starting position
JUMP *OUT IN#(1969)=ON
'
'*************SETUP**************
' store current location
' check cube 30
DIN LB030 SOUT#(86)
' check cube 31
DIN LB031 SOUT#(87)
' check cube 32
DIN LB032 SOUT#(88)
' check cube 33
DIN LB033 SOUT#(89)
' check cube 34
DIN LB034 SOUT#(90)
' check cube 35
DIN LB035 SOUT#(91)
' check cube 36
DIN LB036 SOUT#(92)
' check cube 37
DIN LB037 SOUT#(93)
```

IN#(1985) will be ON if the Robot has been jogged. The CIO must be modified before this will occur. (see CIO modifications)

Checks if the robot is in a safe starting position

- Sets a Local B Variable to a 1 if the Specified Output (SOUT) is ON.
- The Specified Output will be ON if the robots active TCP is inside the associated Cube.
- If the robot has Multiple TCP's, the active TCP is used.
- Another option is that the 5XXXX bit associated with the Cube can be mapped to a Universal Input 0XXXX and the input can be used.

- Calls the appropriate HOME_FROM_CUBE job base on LB = 1.
- Since this is inside an IFTHEN structure, only one HOME_FROM_CUBE job will be called. So, if the robot's TCP is in more than one cube, the first HOME_FROM_CUBE job will be executed.
- Another option is that the 5XXXX bit associated with the Cube can be mapped to a Universal Input 0XXXX and the input can be used.

```
'**********ACTION**********
' find current location
' check cube 30
IFTHEN LB030=1
   CALL JOB:HOME_FROM_CUBE_30
ELSEIF LB031=1
   CALL JOB:HOME_FROM_CUBE_31
ELSEIF LB032=1
   CALL JOB:HOME_FROM_CUBE_32
ELSEIF LB033=1
   CALL JOB:HOME_FROM_CUBE_33
ELSEIF LB034=1
   CALL JOB:HOME_FROM_CUBE_34
ELSEIF LB035=1
   CALL JOB:HOME_FROM_CUBE_35
ELSEIF LB036=1
   CALL JOB:HOME_FROM_CUBE_36
ELSEIF LB037=1
   CALL JOB:HOME_FROM_CUBE_37
ELSEIF LB038=1
.
ENDIF
'*******************************
'
'*********HOUSE KEEPING*********
*OUT
' reset robot jogged bit
PULSE OT#(1985) T=0.20
'*******************************
```

Resets to Robot Jogged Bit. The CIO must be modified before this will occur. (see CIO modifications)

HOME_FROM_CUBE Job

Create each HOME_FROM_CUBE JOB

A Temporary JUMP to be removed after the Job is Taught and verified.

(Optional) Makes sure the TCP is in Cube #30. (See Cube Specified Outputs)

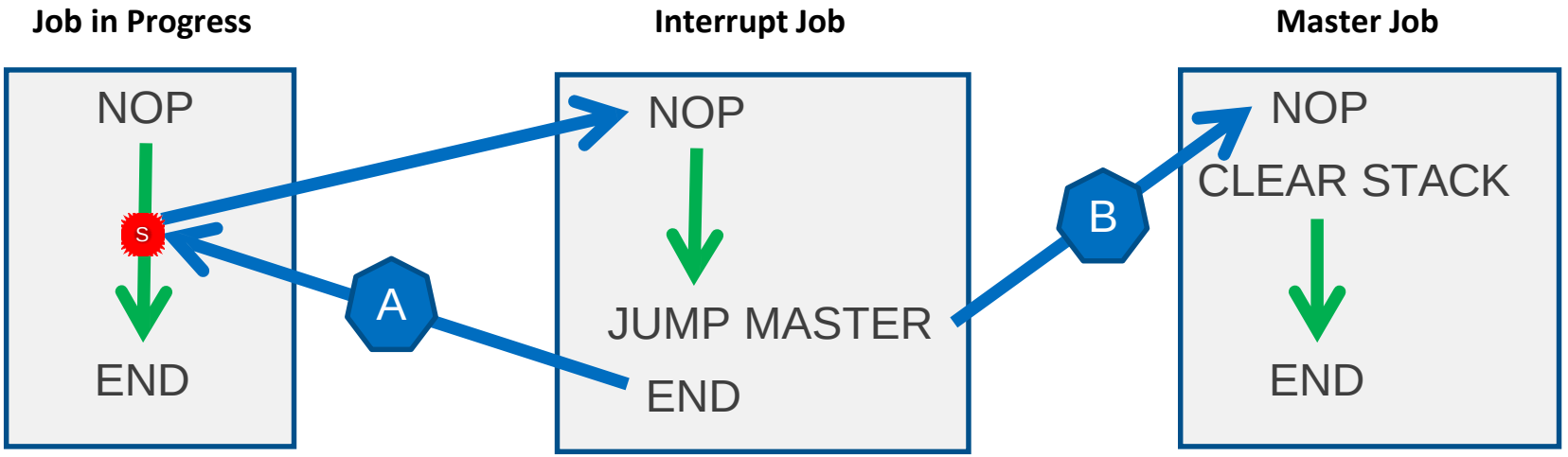
(Optional) Checks to see if the Robot is already Home. Cube#64 in this case is used for Home Position.

Taught Path Home from Cube #30.

```
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ '$      AUTOHOME JOB  $
'$                FROM CUBE#30                $
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ '
'********DATA VALIDATION********* ' REMOVE JUMP AFTER
TAUGHT
JUMP *OUT
' check in cube 30 DIN LB000 SOUT#(86)
JUMP *OUT IF LB000<>1
' check already at HOME CUBE#64 ' check cube 64
DIN LB000 SOUT#(120)
JUMP *OUT IF LB000=1
'******************************** '
'************ACTION************
' path back home '
CALL JOB:HOME
'******************************** '
'*********HOUSE KEEPING**********
*OUT
'********************************
```

What is the Interrupt Job?

The interrupt job is a job that is automatically called and executed based on the status of an input signal. The job that was currently in progress is momentarily suspended. In the case of using AutoHome the Interrupt job restarts the MASTER job and does not return to the job in progress. (See Interrupt Job Manual)



Path A: If the END of the INTERRUPT job is reached.

Path B: The AutoHome INTERRUPT job goes to the MASTER job before reaching the END.

Create INTERRUPT JOB

DI Disables all Interrupts

Goes to the top of the MASTER job.

```
NOP
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'$       INTERRUPT JOB        $
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'
'*************SETUP**************
' disable all interrupts
DI
'*********************************
'
'************ACTION*************
' restart master
JUMP JOB:MASTER
'*********************************
END
```

AutoHome_Master

Edit the Master Job.

Clears the job Stack. This must be done because the MASTER job has called from the interrupt job.

DI disables all Interrupts.

IN#(1985) will be ON if the Robot has been jogged. The CIO must be modified before this will occur. (see CIO modifications)
If the robot has not been manually jogged the AUTOHOME job will be executed.

Checks if the robot is in a safe starting position. If it is not in a safe starting position, the robot will display an ALARM. The CIO must be modified before this will occur. (see CIO modifications)

```
'********************************
'**          MASTER            **
'********************************
*TOP
'********DATA VALIDATION*********
' clear the job stack
CLEAR STACK
' disable all interrupts
DI
TIMER T=0.50
'autohome if robot not jogged
IFTHEN IN#(1985)=OFF
        CALL JOB:AUTOHOME
ENDIF
'check robot is in a safe
'   starting position
WHILE IN#(1969)=OFF
        '################################
         SETUALM 8000 "ROBOT NOT AT SAFE STARTING POS" 1
        'ROBOT NOT AT A SAFE STARTING
        'POSITION
        '################################
         JUMP *TOP
ENDWHILE
'********************************
'
'************SETUP**************
CALL JOB:HOMEPOS
```

AutoHome_Master (Continued)

EI enables all Interrupts

DI disables all Interrupts

```
CALL JOB:XFERDATA
CALL JOB:START-UP
'*******************************
'
'************ACTION*************
' prime loop
SET LB000 0
WHILE LB000=0
        ' enable all interrupts
        EI
        ' pick product
        CALL JOB:PICK
        ' place product
        CALL JOB:PLACE
        ' disable all interrupts
        DI
        '*********HOUSE KEEPING**********
        ' check request end of cycle
        IFTHEN IN#(41)=ON
                DI
                CALL JOB:HOMEPOS
                ' pulse at end of cycle
                PULSE OT#(41) T=1.00
                PAUSE
        ENDIF
ENDWHILE
'*******************************
*OUT
END
```

Concurrent I/O Modifications

Safe Starting Position (IN#1969)

STR #50157 ← Cube#64: If other cubes are used to indicate that the robot is in a safe starting position, they should be added here.

OUT #02470 ← Input # 1969

Robot Jogged (IN#1985)

STR #50076 ← Robot manually Jogged. Will be true if the Robot has been jogged in TEACH Mode. 50076 is automatically reset when the robot is restarted in PLAY Mode.

STR #02490 OR-STR ← Seal (Latch)

AND-NOT #12490 ← Unseal (Unlatch) Output # 1985 PULSED in INFORM Job.

OUT #02490

Cube Specific Outputs

Cube#1	50080	SOUT#57	Cube#17	50100	SOUT#73	Cube#33	50120	SOUT#89	Cube#49	50140	SOUT#105
Cube#2	50081	SOUT#58	Cube#18	50101	SOUT#74	Cube#34	50121	SOUT#90	Cube#50	50141	SOUT#106
Cube#3	50082	SOUT#59	Cube#19	50102	SOUT#75	Cube#35	50122	SOUT#91	Cube#51	50142	SOUT#107
Cube#4	50083	SOUT#60	Cube#20	50103	SOUT#76	Cube#36	50123	SOUT#92	Cube#52	50143	SOUT#108
Cube#5	50084	SOUT#61	Cube#21	50104	SOUT#77	Cube#37	50124	SOUT#93	Cube#53	50144	SOUT#109
Cube#6	50085	SOUT#62	Cube#22	50105	SOUT#78	Cube#38	50125	SOUT#94	Cube#54	50145	SOUT#110
Cube#7	50086	SOUT#63	Cube#23	50106	SOUT#79	Cube#39	50126	SOUT#95	Cube#55	50146	SOUT#111
Cube#8	50087	SOUT#64	Cube#24	50107	SOUT#80	Cube#40	50127	SOUT#96	Cube#56	50147	SOUT#112
Cube#9	50090	SOUT#65	Cube#25	50110	SOUT#81	Cube#41	50130	SOUT#97	Cube#57	50150	SOUT#113
Cube#10	50091	SOUT#66	Cube#26	50111	SOUT#82	Cube#42	50131	SOUT#98	Cube#58	50151	SOUT#114
Cube#11	50092	SOUT#67	Cube#27	50112	SOUT#83	Cube#43	50132	SOUT#99	Cube#59	50152	SOUT#115
Cube#12	50093	SOUT#68	Cube#28	50113	SOUT#84	Cube#44	50133	SOUT#100	Cube#60	50153	SOUT#116
Cube#13	50094	SOUT#69	Cube#29	50114	SOUT#85	Cube#45	50134	SOUT#101	Cube#61	50154	SOUT#117
Cube#14	50095	SOUT#70	Cube#30	50115	SOUT#86	Cube#46	50135	SOUT#102	Cube#62	50155	SOUT#118
Cube#15	50096	SOUT#71	Cube#31	50116	SOUT#87	Cube#47	50136	SOUT#103	Cube#63	50156	SOUT#119
Cube#16	50097	SOUT#72	Cube#32	50117	SOUT#88	Cube#48	50137	SOUT#104	Cube#64	50157	SOUT#120

Example Layout