NxGen Automated Builder

The automation process needs the following steps:

. Recognizing the different parts involved (static data) and define them.

. Recognizing how the parts are used in the process (dynamic data)

Entity Relationship is used for the static data involving the creation of tables in a SQL system

Coloured Petri Net is used for the dynamic data involving the creation of Views for Places and calling functions for the Transitions

Coloured Petri Net could be replaced by a State Machine but it’s the most recommended.
State Machine is serializable.
Coloured Petri Net is concurrent.

Notice that the use of a database is not mandatory but recommended.

The different Rows in the database are represented by different colours.

Alan El-Sabbagh WHATSAPP: +971551574255

Every transition has an input and produce an output.
When status change:
Automatically the row is deleted from input view and new one is inserted to the output view.
In every transition a command is sent to the Robot to do a certain job, the robot execute the job and the computer wait for it to complete the job.

A command to the robot could be formed from many basic commands to the robot.

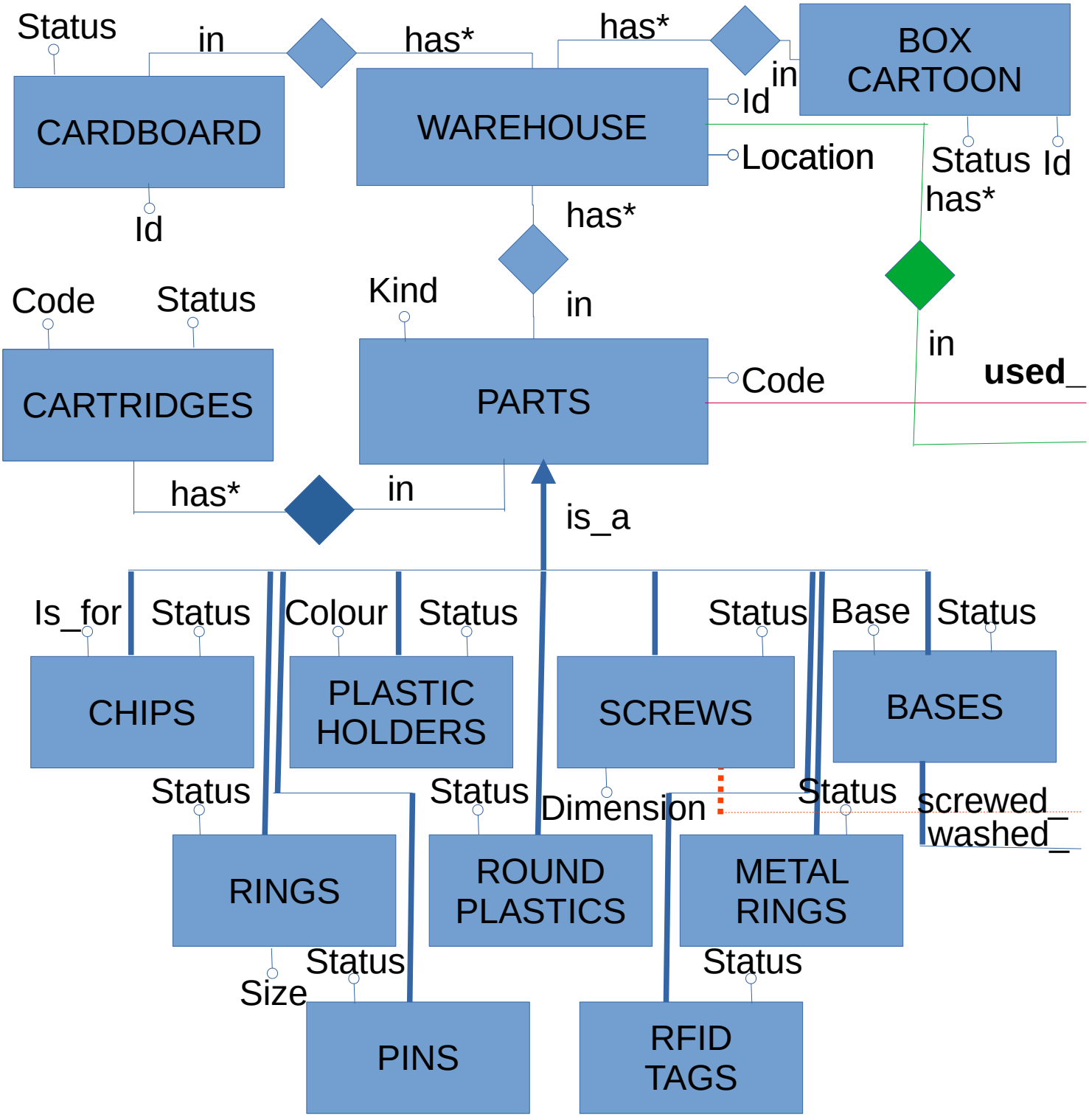
A transition with high level command could be expanded in many transitions / places in which each transition has a call to a single basic command but it is recommended to use a high level command. Note that it will expand by starting with one transition and ending in a transition.

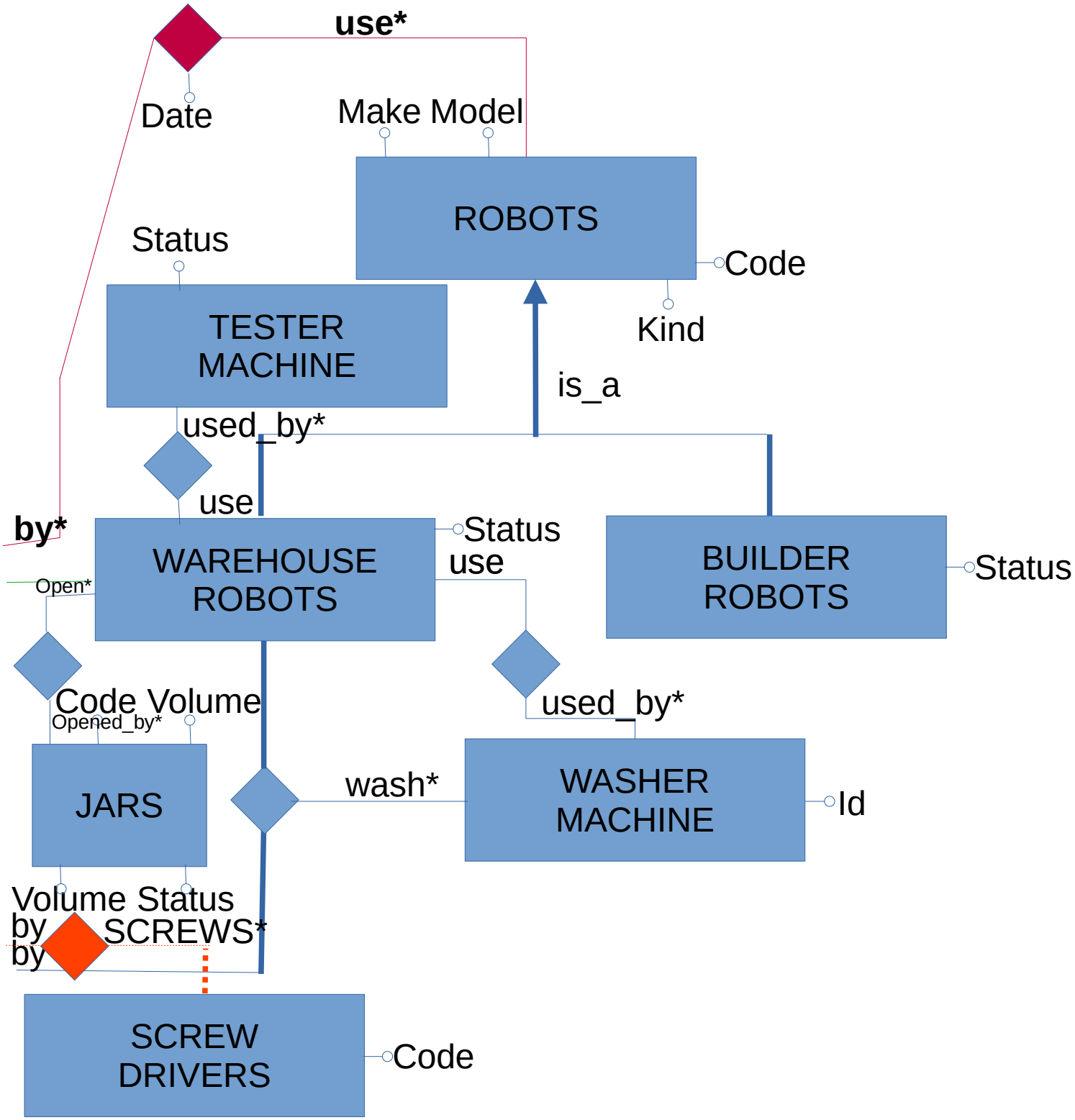
A place could be expanded in many places / transitions like in the case of the robot bringing the bases to the washer machine; could be manually done by the employee the wash process by using one place or extended in many places / transitions to describe the robot using the washer machine. Note that it will expand by starting with one place and ending in a place.

Craig Machinery and Design tool Inc are leaders in Robots manufacturer but any robot arm and a hand robot could be used.
http://www.k-team.com/khepera-iv
Amazon: 6DOF Mechanical Arm Claw Kit, DOF Manipulator Industrial Robot Mechanical Arm Gripper Automatic Robot Parts

Main program
- **Create the tables**
- **Create the views**
- **Create one thread for every routine**
  Every thread check:
  If exists the minimum Rows in the View for every Input View then run the Routine

STATIC DATA ENTITY RELATIONSHIP




use*
Date
Make Model
ROBOTS
Status
Code
TESTER MACHINE
Kind
is_a
used_by*
use
by*
WAREHOUSE ROBOTS
Status
use
BUILDER ROBOTS
Status
Open*
Code Volume
Opened_by*
used_by*
JARS
wash*
WASHER MACHINE
Id
Volume Status
by
by
SCREWS*
SCREW DRIVERS
Code

```
create table WAREHOUSE
        (
           Id                      INT,
           Location                VARCHAR(25)
        );

create table PARTS
        (
           Code                    VARCHAR(10),
           Kind                    VARCHAR(25),
           Id_Warehouse            INT                 /* join link */
        );

/* Every row in a part table has an Id, it is auto incremented,
      it is useful to distinguish every individual part, it work like
      the colour function in the Petri Net */

create table CHIPS
        (
           Id                      INT    AUTOINCREMENT,
           Code_Part               VARCHAR(10),   /* is_a link */
           Is_for                  VARCHAR(15),
           Status                  VARCHAR(15),
           Code_Plastic_Holder     VARCHAR(10)    /* join link */
        );

create table PLASTIC_HOLDERS
        (
           Id                      INT    AUTOINCREMENT,
           Code_Part               VARCHAR(10),   /* is_a link */
           Colour                  VARCHAR(10),
           Status                  VARCHAR(15)
        );
```

```
create table SCREWS
        (
           Id                      INT    AUTOINCREMENT,
           Code_Part               VARCHAR(10),   /* is_a link */
           Dimension               VARCHAR(25),
           Status                  VARCHAR(15),
           Code_Screw_Driver       VARCHAR(10),   /* join link */
           Code_Plastic_Holder     VARCHAR(10)    /* join link*/
        );

create table BASES
        (
           Id                      INT    AUTOINCREMENT,
           Code_Part               VARCHAR(10),   /* is_a link */
           Base                    VARCHAR(10),
           Status                  VARCHAR(15),
           Code_Plastic_Holder     VARCHAR(10)    /* join link */
        );

create table RINGS
        (
           Id                      INT    AUTOINCREMENT,
           Code_Part               VARCHAR(10),   /* is_a link */
           Size                    VARCHAR(10),
           Status                  VARCHAR(15),
           Code_Plastic_Holder     VARCHAR(10)     /* join link */
        );
```

```
create table ROUND_PLASTIC
        (
           Id                    INT    AUTOINCREMENT,
           Code_Part             VARCHAR(10),   /* is_a link */
           Status                VARCHAR(15),
           Code_Plastic_Holder   VARCHAR(10)    /* join link */
        );

create table METAL_RINGS
        (
           Id                    INT    AUTOINCREMENT,
           Code_Part             VARCHAR(10),   /* is_a link */
           Status                VARCHAR(15),
           Code_Plastic_Holder   VARCHAR(10)   /* join link */
        );

create table PINS
        (
           Id                    INT    AUTOINCREMENT,
           Code_Part             VARCHAR(10),
           Status                VARCHAR(15),
           Code_Plastic_Holder   VARCHAR(10)
        );

create table RFID_TAGS
        (
           Id                    INT    AUTOINCREMENT,
           Code_Part             VARCHAR(10),
           Status                VARCHAR(15),
           Code_Plastic_Holder   VARCHAR(10)
        );
```

```
create table ROBOTS
	(
	   Code                 INT,
	   Make                 VARCHAR(25),
	   Model                VARCHAR(25),
	   Kind                 VARCHAR(10),
	);

create table WAREHOUSE_ROBOTS
	(
	   Code_Robot           INT,            /* is_a link */
	   Id_Warehouse         INT,            /* join link */
	   Status               VARCHAR(15),
	   Id_Washer_Machine    INT             /* join Link */
	);

create table BUILDER_ROBOTS
	(
	   Code_Robot           INT,              /* is_a link */
	   Status               VARCHAR(15)
	);
create table PARTS_ROBOTS                     /* join link */
	(
	   Code_Part            VARCHAR(10),
	   Code_Robot           VARCHAR(10),
	   Date                 DATE
	);

create table WASHER_MACHINE
	(
	   Id                       INT,
	   Status                   VARCHAR(15)
	);
```

```
create table WASHING                                    /* join link */
        (
           Code_Warehouse_Robot    INT,
           Id_Washer_Machine       INT,
           Code_Base               VARCHAR(10)
        );

create table SCREW_DRIVERS
        (
           Code                    VARCHAR(10),
           Status                  VARCHAR(15)
        );

create tabe SCREWED
        (
           Code_Screws             VARCHAR(10),
           Code_Screw_Driver       VARCHAR(10) /* join link */
        );

create table CARTRIDGES
        (
           Id                      INT    AUTOINCREMENT,
           Code                    VARCHAR(10),
           Status                   VARCHAR(15)
        );

create table PARTS_CARTRIDGES                           /* join link */
        (
           Code_Part               VARCHAR(10),
           Code_Cartridge          VARCHAR(10)
        );

/* Table PARTS_CARTRIDGES is not used because it is better
     to delete the parts used in building in database when the
     cartridge is built */
```

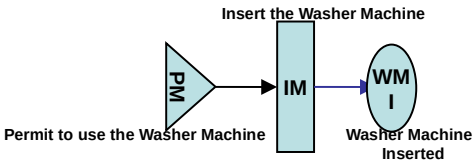
```
create table JARS
	(
	   Code                              INT,
	   Status                            VARCHAR(15),
	   Volume                            VARCHAR(15),
	);

create table JARS_WAREHOUSE_ROBOT      /* join link */
	(
	   Code_Jar                          INT,
	   Code_Warehouse_Robot              INT
	);

create table TESTER_MACHINE
	(
	   Code                              INT,
	   Status                            VARCHAR(15),
	);

create table TESTER_MACHINE_ROBOT      /* join link */
	(
	   Code_Tester_Machine               INT,
	   Code_Warehouse_Robot              INT
	);
```

```
create table BOX_CARTOON
	(
	   Id                 INT            AUTOINCREMENT,
	   Status             VARCHAR(15),
	   Code_Warehouse     INT            /* join link */
	);

create table CARDBOARD
	(
	   Id                 INT            AUTOINCREMENT,
	   Status             VARCHAR(15),
	   Code_Warehouse     INT            /* join link */
	);
```

DYNAMIC DATA COLOURED PETRI NETS

Boxes received, processed and be ready

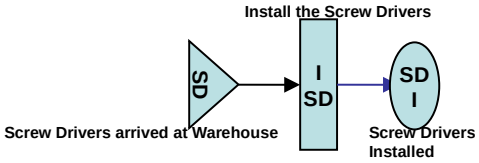
Alan El-Sabbagh WHATSAPP: +971551574255



Message PM: Permit to use the Washer Machine

Transition IM: Insert a record for the Washer Machine
routine IM (accept PM; return WMI)
insert into Washer_Machine
(Id, Status)
values
(1, 'idle');

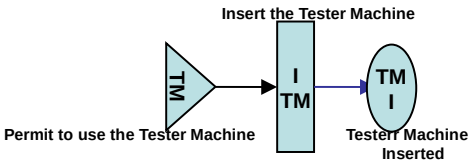
Place WMI: Washer Machine ready to use
create view WMI as
select *
from Washer_Machine
where Status = 'idle';



Message SD: Screw Drivers arrived at the Warehouse

Transition ISD: Install the Screw Drivers
routine ISD (accept SD; return SDI)
/* Install the Screw Drivers on the wall on Top of the Table */
insert into SCREW_DRIVERS
(Code, Status)
values
('1', 'idle');
insert into SCREW_DRIVERS
(Code, Status)
values
('2', 'idle');

Place SDI: Screw Driver Installed
create view SDI as
select *
from SCREW_DRIVERS
where Status = 'idle';



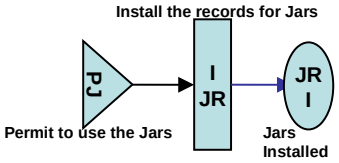
Message TM: Permit to use the Tester Machine

Transition ITM: Insert record for the Tester Machine

```
routine ITM(accept TM; return TMI)
   insert into TESTER_MACHINE
      ( Code, Status )
    values
      ( 1,       'off'    );
```

Place TMI: Tester Machine ready to use

```
create view TMI as
 select *
    from  Tester_Machine
      where Status = 'off';
```



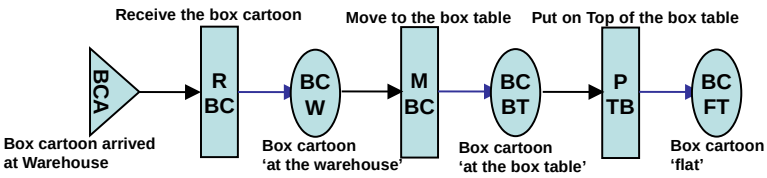
Message PJ: Premit to use the jar bottles

Transition IJR: Install the records for jar bottles

```
routine IJR (accept PJ; return JRI)
   /* Install the records for the Jar bottles */
  insert into JARS
      ( Code, Status, Volume      )
    values
      ( 1,     'close', 'not empty');
insert into JARS
      ( Code, Status, Volume      )
    values
      ( 2,     'close', 'not empty');
```

Place JRI: Records for Jars Inserted

```
create view SDI as
 select *
    from  JARS
      where (Status = 'close') and (Volume = 'not empty');
```



Message BCA: Box cartoon arrived at the Warehouse with the Quantity

Transition RBC: receive the box cartoon
routine RBC (accept BCA; return BCW)
/* *Put the box cartoon in warehouse* */
for i = 1, (i <= QuantityInLabel) , i++
insert into BOX_CARTOON
(Status, Code_Warehouse)
values
('at the warehouse', 1);

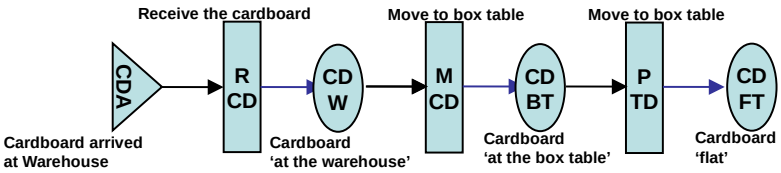
Place BCW: Box cartoon in warehouse
create view BCW as
select *
from BOX_CARTOON
where Status = 'at the warehouse';

Transition MBC: move the box cartoon to the Box Table
routine MBC (accept BCW; return BCBTB)
/* *Move the cartoon boxes to the box table* */
for i = 1, (i <= QuantityToMove) , i++
update BOX_CARTOON
set Status = 'at the box table'
where Status = 'at the warehouse';

Place BCBT: Box cartoon boxes at the box table
create view BCBT as
select *
from BOX_CARTOON
where Status = 'at the box table';

Transition PTB: move a box cartoon on top of the Box Table
routine PTB (accept BCBT; return BCFT)
/* *Move a box cartoon on top of the box table* */
update BOX_CARTOON
set Status = 'flat' where
where Status = 'at the box table';

Place BCFT: A box cartoon box on top of the box table
create view CBTP as
select *
from BOX_CARTOON
where Status = 'flat';



Message CDA: Cardboard arrived at the Warehouse with the Quantity

Transition RCD: receive the cardboard
routine RCD (accept CDA; return CDW)
/* *Put the cardboard in warehouse* */
for i = 1, (i <= QuantityInLabel) , i++
insert into CARDBOARD
(Status, Code_Warehouse)
values
('at the warehouse', 1);

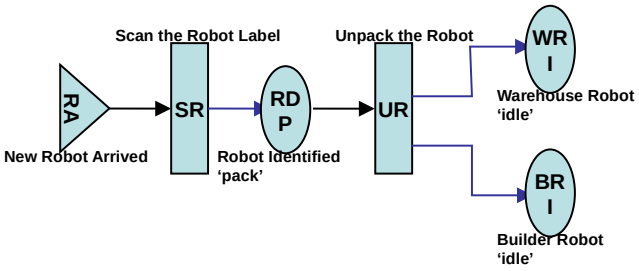
Place CDW: Cardboard in warehouse
create view CDW as
select *
from CARDBOARD
where Status = 'at the warehouse';

Transition MCD: move the cardboard to the Box Table
routine MCD (accept CDW; return CDBT
/* *Move the cardboard to the box table* */
for i = 1, (i <= QuantityToMove) , i++
update CARDBOARD
set Status = 'at the box table' where
where Status = 'at the warehouse';

Place CDBT: Cardboard at the box table
create view CDBT as
select *
from CARDBOARD
where Status = 'at the box table';

Transition PTD: move a cardboard on top of the Box Table
routine PTD (accept CDBT; return CBFT)
/* *Move a cardboard on top of the box table* */
update CARDBOARD
set Status = 'flat' where
where Status = 'at the box table';

Place CDFT: A cardboard on top of the box table
create view CDFT as
select *
from CARDBOARD
where Status = 'flat';



Message RA: New Robot Arrived

Transition SR: Scan the Robot Label

```
routine SR (accept RA; return RIP)
   scan_label(label)
   /* Use a scanner to read the label */
   if is_warehouse then
      insert into ROBOTS
         ( Code,  Make,   Model,   Kind          )
       values
         ( 1,      'make', 'model', 'warehouse' );
      insert into WAREHOUSE_ROBOT
         ( Code_Robot, Id_Warehouse, Status, Id_Washer_Machine )
       values
         ( 1,                1,                'pack', 1                        );
   else
      insert into ROBOTS
         ( Code,  Make,   Model,   Kind      )
       values
         ( 2,       'make', 'model', 'builder' );
      insert into  BUILDER_ROBOT
         ( Code_Robot, Status )
       values
         ( 2,                'pack' );
```

Place RDP: New Robot

```
create view RDP as
 select *
    from  ROBOTS
      where ( CHILD(ROBOTS).Status = 'pack' );
```

Transition UR: Unpack the Robot's package

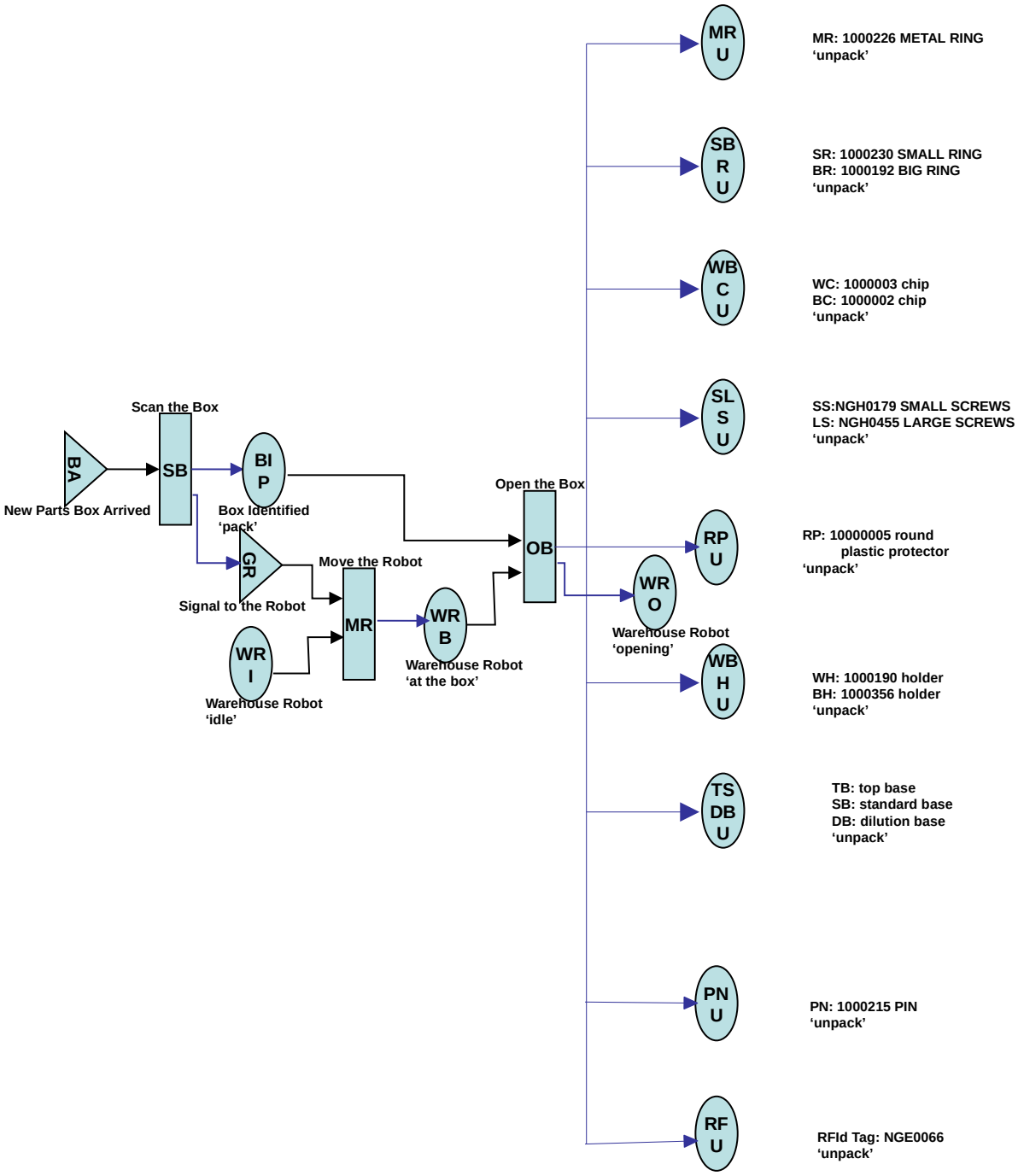
```
routine UR (accept RDP; return WRI guard ROBOTS.Kind = 'warehouse',
                                BRI  guard ROBOTS.Kind = 'builder')
   if RDP.Kind = 'warehouse' then
       move_robot(R,corner)
        /* move_robot(robot, distance) move the robot by the 'distance',
           'corner' just a way to go to the beginning place. */
       waitfor(ProcessTime)
       update WAREHOUSE_ROBOTS set
          Status = 'idle';
   else
      install_robot(R,table);
        /* install_robot(robot, place) could be a manual installation of a hand
            robot or the Place BRI could be expand in Places/Transitions
            where the warehouse robot could be used to build it */
       waitfor(ProcessTime)
       update BUILDER_ROBOTS set
          Status = 'idle';
```

Place WRI: Warehouse Robot with status 'idle'

```
create view WRI as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = 'idle';
```

Place BRI: Builder Robot with status 'idle'

```
create view BRI as
 select *
    from  BUILDER_ROBOTS
      where Status = 'idle';
```


BA
New Parts Box Arrived
SB
Scan the Box
BI P
Box Identified
'pack'
GR
Signal to the Robot
Move the Robot
MR
WR I
Warehouse Robot
'idle'
WR B
Warehouse Robot
'at the box'
OB
Open the Box
WR O
Warehouse Robot
'opening'
MR U
MR: 1000226 METAL RING
'unpack'
SBR U
SR: 1000230 SMALL RING
BR: 1000192 BIG RING
'unpack'
WBC U
WC: 1000003 chip
BC: 1000002 chip
'unpack'
SLS U
SS:NGH0179 SMALL SCREWS
LS: NGH0455 LARGE SCREWS
'unpack'
RP U
RP: 10000005 round
plastic protector
'unpack'
WBH U
WH: 1000190 holder
BH: 1000356 holder
'unpack'
TSDB U
TB: top base
SB: standard base
DB: dilution base
'unpack'
PN U
PN: 1000215 PIN
'unpack'
RF U
RFId Tag: NGE0066
'unpack'

Message BA: Box arrived with new parts mentioned in label with the quantity

Transition SB: Scan the box

```
routine SB (accept BA; return BIP)
begin
   scan_label(label)
   /* Use a scanner to read the label */
   for i = 1, ( i <= QuantityInLabel ) , i++
    case CodeInLabel of
      ‘1000226’:
        insert into PARTS ( Code,       Kind,             Id_Warehouse ) values
                          ( ‘1000226’, ‘metal rings’,  1                  );
        insert into METAL_RINGS ( Code_Part, Status, Code_Plastic_Holder ) values
                                ( ‘1000226’,   ‘pack’,  NULL                      );
     ‘1000230’:
        insert into PARTS ( Code,       Kind,            Id_Warebouse ) values
                          ( ‘1000230’, ‘small rings’, 1);
        insert into RINGS ( Code_Part, Size,   Status, Code_Plastic_Holder ) values
                          ( ‘1000230’,   ‘size’, ‘pack’,  NULL                      );
     ‘1000192’:
        insert into PARTS ( Code,       Kind,        Id_Warehouse ) values
                          ( ‘1000192’, ‘big rings’ 1                  );
        insert into RINGS ( Code_Part, Size,   Status, Code_Plastic_Holder ) values
                          ( ‘1000192’,   ‘size’, ‘pack’,  NULL                      );
    ‘1000003’:
       insert into PARTS( Code,       Kind ) values
                        ( ‘1000003’, ’400/400D Chip’);
        insert into CHIPS ( Code_Part, Is_For,        Status, Code_Plastic_Holder ) values
                          ( ‘1000003’,   ‘400/400D’, ‘pack’    1000190                );
    ‘1000002’:
        insert into PARTS(Code,       Kind                ) values
                         (‘1000002’, ‘500/500D Chip’ );
        insert into CHIPS ( Code_Part, Is_For,        Status, Code_Plastic_Holder ) values
                          ( ‘1000002’,   ‘500/500D’, ‘pack’,   1000356                );
    ‘NGH0179’:
        insert into PARTS( Code,         Kind              ) values
                         ( ‘NGH0179’, ‘small screws’ );
        insert into SCREWS ( Code_Part, Dimension,   Status,
                             Code_Screw_Driver, Code_Plastic_Holder ) values
                           ( ‘NGH0179’, ‘dimension’,  ‘pack’,
                             1,                       NULL                    );
    ‘NGH0455’:
        insert into PARTS( Code,         Kind              ) values
                         ( ‘NGH0455’, ‘large screws’ );
        insert into SCREWS ( Code_Part, Dimension,   Status,
                             Code_Screw_Driver, Code_Plastic_Holder ) values
                           ( ‘NGH0455’, ‘dimension’,  ‘pack’,
                             2,                       NULL                    );
```

```
‘1000005’:
   insert into PARTS ( Code,       Kind                        ) values
                     ( ‘1000005’, ‘Round Plastic Tape’ );
   insert into ROUND_PLASTIC ( Code_Part,  Status,
                               Code_Plastic_Holder ) values
                             ( ‘1000005’,    ‘pack’,
                               NULL                  );
‘1000190’:
   insert into PARTS ( Code,       Kind                        ) values
                     ( ‘1000190’, ‘White Plastic Holder’ );
   insert into PLASTIC_HOLDERS ( Code_Part, Colour, Status ) values
                               ( ‘1000190’,  ‘white’,  ‘pack’  );
‘1000356’:
   insert into PARTS ( Code,       Kind                        ) values
                     ( ‘1000356’, ‘Black Plastic Holder’ );
   insert into PLASTIC_HOLDERS ( Code_Part, Colour, Status ) values
                               ( ‘1000356’,   ‘black’, ‘pack’  );
‘codetopbase’:
   insert into PARTS ( Code,             Kind          ) values
                     ( ‘codetopbase’, ‘Top Base’ );
   insert into BASES  ( Code_Part,       Base, Status,
                        Code_Plastic_Holder ) values
                      ( ‘codetopbase’, ‘top’,  ‘pack’,
                        NULL                  );
 ‘codestandardbase’:
   insert into PARTS ( Code,                   Kind                ) values
                     ( ‘codestandardbase’, ‘Standard Base’  );
   insert into BASES  ( Code_Part,             Base,         Status,
                        Code_Plastic_Holder ) values
                      ( ‘codestandardbase’, ‘standard’,  ‘pack’,
                        NULL                  );

‘codedilutionbase’:
   insert into PARTS ( Code,                  Kind              ) values
                     ( ‘codedilutionbase’, ‘Dilution Base’ );
   insert into BASES  ( Code_Part,            Base,      Status,
                        Code_Plastic_Holder ) values
                      (  ‘codedilutionbase’, ‘dilution’, ‘pack’,
                        NULL                  );
‘1000215’:
   insert into PARTS ( Code,       Kind ) values
                     ( ‘1000215’, ‘PIN’ );
   insert into PINS ( Code_Part, Status, Code_Plastic_Holder ) values
                    ( ‘1000215’,   ‘pack’,  NULL                  );
‘NGE0066’:
   insert into PARTS ( Code,         Kind         ) values
                     ( ‘NGE0066’, ‘RFID Tag’ );
   insert into RFID_TAGS ( Code_Part, Status, Code_Plastic_Holder ) values
                         ( ‘NGE066’,   ‘pack’,  NULL                  );
  insert into PARTS_ROBOTS ( Code_Part,     Code_Robot, Date ) values
                           (  CodeInLabel, 1                , now );

create_signal(GR) /* GR is a signal to wake up a warehouse robot */
```

Place BIP: Box identified and inserted into table PARTS

```
create view BIP as
 select *
    from  PARTS where ( Code = CodeInLabel ) and
                          ( child(PARTS).Status = 'pack' );
```

Message GR: A Signal message that a box has been arrived and identified

Transition MR: Move the Robot

```
routine MR (accept GR, WRI; return WRB)
   move_robot(WRI,box_location)
   waitfor(ProcessTime)
   update WAREHOUSE_ROBOTS set
      Status = 'at the box';
```

Place WRB: Warehouse Robot with status 'at the box'

```
create view WRB as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = 'at the box';
```

/* After opening the box, the robot take from the box, the bag of parts */

Transition OB: Open the Box

```
routine OB ( accept BIP, WRB;
                return  MRU     guard PARTS.Code = '1000226',
                        /*  SBRU   = SRU  and BRU  */
                        SRU      guard PARTS.Code = '1000230',
                        BRU      guard PARTS.Code = '1000192',
                        /*  WBCU  = WCU  and BCU  */
                        WCU     guard PARTS.Code = '1000003',
                        BCU      guard PARTS.Code = '1000002',
                        /*  SLSU    = SSU  and LSU */
                        SSU      guard PARTS.Code = 'NGH0179',
                        LSU      guard PARTS.Code = 'NGH0455',
                        PRU      guard PARTS.Code = '1000005',
                        /*  WBHU  = WHU and BHU  */
                        WHU     guard PARTS.Code = '1000190',
                        BHU      guard PARTS.Code = '1000356',
                        /*  TSDBU = TBU  and SBU and DBU  */
                        TBU      guard PARTS.Code = 'codetopbase',
                        SBU      guard PARTS.Code = 'codestandardbase',
                        DBU      guard PARTS.Code = 'codedilutionbase',
                        PNU      guard PARTS.Code = '1000215',
                        RFU      guard PARTS.Code = 'NGE0066',
                        WRO
             )
```

```
    cutbox(WR,box)
    /*
       cutbox is a command to the robot to grab a knife or a tool
       with a cutter depend on the model of a robot (some of robots came
       with many tools for grab, cut, ...) setting the steps to cut the box,
       some robots have the capabilities of automatic knowledge of the
       dimension due to the camera installed
    */
   waitfor(ProcessTime)
   flipbox(WR,box)
   /*
    flipbox is a commant to turn the box 180 degrees to empty it
   */
   waitfor(ProcessTime)
   update child(PARTS) set
      Status = 'unpack' where ( Code = CodeInLabel )

Place MRU: The bag of Metal Rings
create view MRU as
 select *
    from  METAL_RINGS
      where ( Code_Part = '1000226' ) and ( Status = 'unpack' );

Place SRU: The bag of Small Rings
create view SRU as
 select *
    from  RINGS
      where ( Code_Part = '1000230' ) and ( Status = 'unpack' );

Place BRU: The bag of Large Rings
create view BRU as
 select *
    from  RINGS where
      where ( Code_Part = '1000192' ) and ( Status = 'unpack' );

Place WCU: The bag of 1000003 Chips
create view WCU as
 select *
    from  CHIPS
      where ( Code_Part = '1000003' ) and ( Status = 'unpack' );

Place BCU: The bag of 1000002 Chips
create view BCU as
 select *
    from  CHIPS
      where ( Code_Part = '1000002' ) and ( Status = 'unpack' );

Place SSU: The boxes of Small Screws
create view SSU as
 select *
    from  SCREWS where
      where ( Code_Part = 'NGH0179' ) and ( Status = 'unpack' );
```

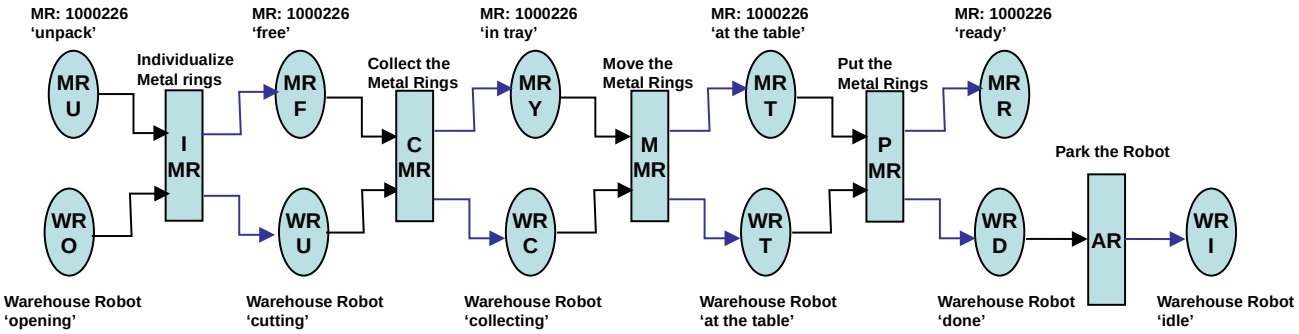
```
Place LSU: The boxes of Large Screws
create view LSU as
 select *
    from  SCREWS
      where ( Code_Part = 'NGH0455' ) and ( Status = 'unpack' );

Place RPU: The Tape of Round Plastic Protector
create view PRU as
 select *
    from  ROUND_PLASTIC
      where ( Code_Part = '1000005' ) and ( Status = 'unpack' );

Place WHU: The bag of White Plastic Holder
create view WHU as
 select *
    from  PLASTIC_HOLDERS
      where ( Code_Part = '1000190' ) and ( Status = 'unpack' );

Place BHU: The bag of Black Plastic Holder
create view BHU as
 select *
    from  PLASTIC_HOLDERS
      where ( Code_Part = '1000356' ) and ( Status = 'unpack' );

Place TBU: Top Bases wrapped
create view TBU as
 select *
    from  BASES
      where ( Code_Part = 'codetopbase')  and ( Status = 'unpack' );

Place SBU: Standard Bases wrapped
create view SBU as
 select *
    from  BASES
      where ( Code_Part = 'codestandardbase')  and ( Status = 'unpack' );

Place DBU: Dilution Bases wrapped
create view DBU as
 select *
    from  BASES
      where ( Code_Part = 'codedilutionbase')  and ( Status = 'unpack' );

Place PNU: The bag of Pins
create view PNU as
 select *
    from  PINS
      where ( Code_Part = '1000215' ) and ( Status = 'unpack' );
```

Place RFU: The Roll of RFID Tags

```
create view RFU as
 select *
    from  RFID_TAGS
     where ( Code_Part = 'NGE0066' ) and ( Status = 'unpack' );
```

Place WRO: Warehouse Robot with status 'opening'

```
create view WRO as
 select *
    from  WAREHOUSE_ROBOTS
       where Status = 'opening';
```



Transition IMR: Get the Metal Rings out of the bag

```
routine IMR (accept MRU, WRO;
                 return  MRF, WRU)
   emptybag(WR,bag)
   /*
     emptybag is a command to the robot to grab a bag, cut it  and empty it.
     The transaction itself, like any other transaction where a command to a
     robot has issued and this command require many steps, could be expanded
     in many Transactions/Places to follow the various moves of the robot.
   */
   waitfor(ProcessTime)
   update METAL_RINGS
      set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
      set status = 'cutting' where Status = 'opening';
```

Place MRF: Metal Rings out of the bag

```
create view MRF as
 select *
    from  METAL_RINGS
      where Status = 'free';
```

Place WRU: Warehouse Robot with status 'cutting'

```
create view WRU as
 select *
    from  WAREHOUSE_ROBOTS
       where Status = 'cutting';
```

Transition CMR: Put the Metal Rings in a Tray

```
routine CMR (accept MRF, WRU;
                 return  MRY, WRC)
   collectrings(WR,rings)
   /*
     Collectrings make the robot hold a tray under the table and aside,
      and in the table there is a pusher positioned vertically that push all
      the metal rings to the tray.
   */
   waitfor(ProcessTime)
```

```
    update METAL_RINGS
      set status = ‘in tray’ where Status = ‘free’;
    update WAREHOUSE_ROBOTS
      set status = ‘collecting’ where Status = ‘cutting’;

Place MRY: Metal Rings places in tray
create view MRY as
 select *
    from  METAL_RINGS
      where Status = ‘in tray’;

Place WRC: Warehouse Robot  with status ‘collecting’
create view WRC as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = ‘collecting’;

Transition MMR: Move the Metal Rings to the table
routine MMR (accept MRY, WRC;
                    return  MRT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update METAL_RINGS
      set status = ‘at the table’ where Status = ‘in tray’;
   update WAREHOUSE_ROBOTS
      set status = ‘at the table’ where Status = ‘collecting’;

Place MRT: Metal Rings at the table
create view MRT as
 select *
    from  METAL_RINGS
      where Status = ‘at the table’;

Place WRT: Warehouse Robot  with status ‘at the table’
create view WRU as
 select *
    from  WAREHOUSE_ROBOTS
       where Status = ‘at the table’;
```

Transition PMR: Put the Metal Rings in a container

```
routine PMR (accept MRT, WRT;
                    return  MRR, WRD)
   PutInTubes(WR,metal_container)
    /*
      The warehouse robot has a camera and use it to check the side of the metal ring
        with pumb to know the top side of a metal ring. The warehouse robot put the
        metal rings on a plastic tube that hold 25 rings separated by gaps between each
        other to make them grab it later easily by the builder robot.
      There is a tray with 5 by 5 plastic tubes. The insertion would be from end to the
         begin. While the builder robot take from the first available to the end. The
         computer keep track of which tube and how much in plastic tube are taken.
   */
   waitfor(ProcessTime)
   update METAL_RINGS
     set status = 'ready' where Status = 'at the table';
   update WAREHOUSE_ROBOTS
     set status = 'done' where Status = 'at the table';
```

Place MRR: Metal Rings are ready in the plastic tubes

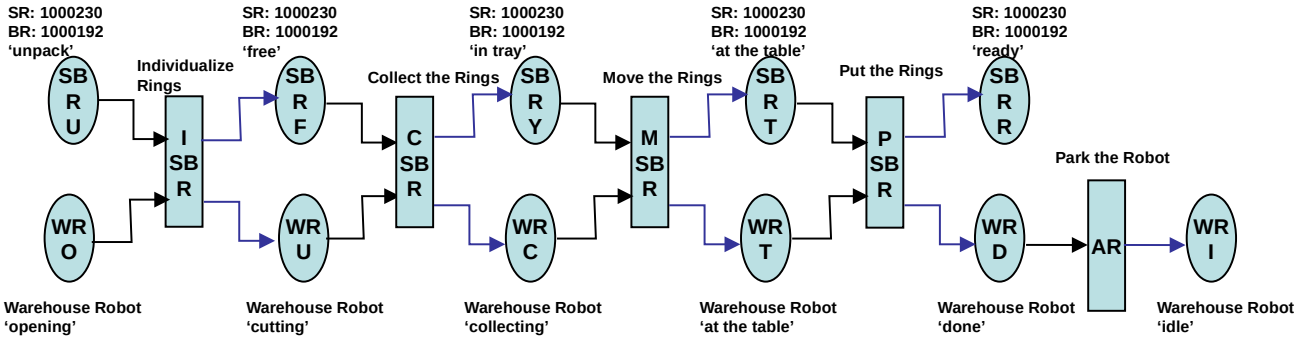
```
create view MRR as
 select *
    from  METAL_RINGS
      where Status = 'ready';
```

Place WRD: Warehouse Robot complete the job

```
create view WRD as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = 'done';
```

Transition AR: Move the Warehouse Robot to its original place

```
routine AR (accept WRD;
                  return  WRI)
   move_robot(WR,corner)
   waitfor(ProcessTime)
   update WAREHOUSE_ROBOTS
     set status = 'idle' where Status = 'done';
```



Transition ISBR: Get the Small Rings or the Big Rings out of the bag

```
routine ISBR (accept SBRU, WRO;
                  return  SBRF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update RINGS
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place SBRF: Small Rings or Big Rings out of the bag

```
create view SBRF as
 select *
    from  RINGS
      where Status = 'free';
```

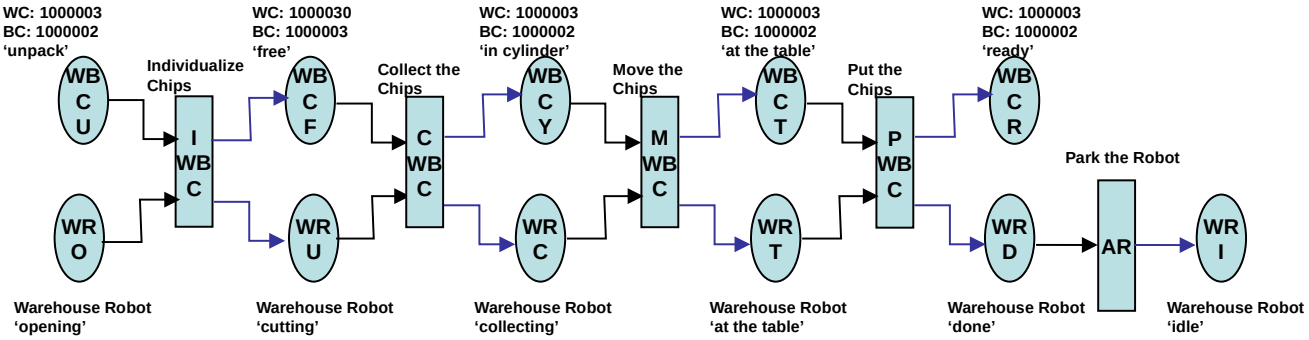
Transition CSBR: Put the Small Rings or the Big Rings in a Tray

```
routine CSBR (accept SBRF, WRU;
                   return  SBRY, WRC)
   collectrings(WR,rings)
   waitfor(ProcessTime)
   update RINGS
     set status = 'in tray' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place SBRY: Small Rings or Big Rings places in tray

```
create view SBRY as
 select *
    from RINGS
      where Status = 'in tray';
```

```
Transition MSBR: Move the Small Rings or the Big Rings to the table
routine MSBR (accept SBRY, WRC;
                      return  SBRT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update RINGS
     set status = ‘at the table’ where Status = ‘in tray’;
   update WAREHOUSE_ROBOTS
     set status = ‘at the table’ where Status = ‘collecting’;

Place SBRT: Metal Rings at the table
create view SBRT as
 select *
    from  RINGS
      where Status = ‘at the table’;

Transition PSBR: Put the Small Rings or the Big Rings in a container
routine PSBR (accept SBRT, WRT;
                     return  SBRR, WRD)
   PutInTubes(WR,rings_container)
    /*
       PutInTubes make the robot put all the Small Rings or the Big Rings
       in their Plastic Tubes for building purpose.
   */
   waitfor(ProcessTime)
   update RINGS
     set status = ‘ready’ where Status = ‘at the table’;
   update WAREHOUSE_ROBOTS
     set status = ‘done’ where Status = ‘at the table’;

Place SBRR: Small Rings or Big Rings are ready in the plastic tubes
create view SBRR as
 select *
    from  RINGS
       where Status = ‘ready’;
```



Transition IWBC: Get the Chips out of the bag
routine IWBC (accept WBCU, WRO;
return WBCF, WRU)
emptybag(WR,bag)
waitfor(ProcessTime)
update CHIPS
set status = 'free' where Status = 'unpack';
update WAREHOUSE_ROBOTS
set status = 'cutting' where Status = 'opening';

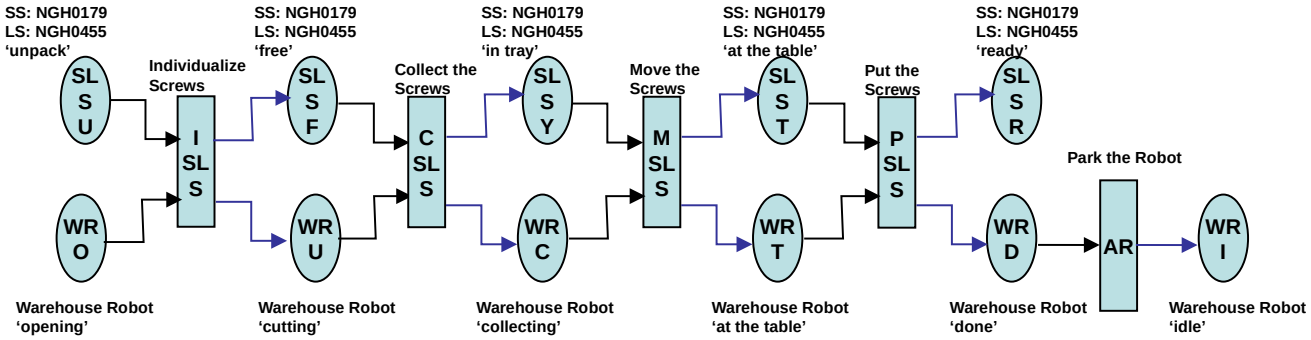
Place WBCF: Chips out of the bag
create view WBCF as
select *
from CHIPS
where Status = 'free';

Transition CWBC: Put the Chips in a cylinder
routine CWBC (accept WBCF, WRU;
return WBCY, WRC)

collectchips(WR,chips)
/*
collectChips make the robot use a plastic cylinder with a spiral inside to stack the chips.
***/**
waitfor(ProcessTime)
update CHIPS
set status = 'in cylinder' where Status = 'free';
update WAREHOUSE_ROBOTS
set status = 'collecting' where Status = 'cutting';

Place WBCY: Chips stacked in cylinder
create view WBCY as
select *
from CHIPS
where Status = 'in cylinder';

```
Transition MWBC: Move the Chips Cylinder to the table
routine MWBC (accept WBCY, WRC;
                      return  WBCT, WRT)
    move_robot(WR,table_location)
    waitfor(ProcessTime)
    update CHIPS
      set status = ‘at the table’ where Status = ‘in cylinder’;
    update WAREHOUSE_ROBOTS
      set status = ‘at the table’ where Status = ‘collecting’;

Place WBCT: Chips Cylinder at the table
create view WBCT as
 select *
     from  CHIPS
       where Status = ‘at the table’;

Transition PWBC: Put the Chips Cylinder to the wall
routine PWBC (accept WBCT, WRT;
                     return  WBCR, WRD)
    PutCylinder(WR,cylinder)
     /*
      PutCylinder make the robot put the chips cylinder in a grabber attached
      to the wall for building purpose.
    */
    waitfor(ProcessTime)
    update CHIPS
      set status = ‘ready’ where Status = ‘at the table’;
    update WAREHOUSE_ROBOTS
      set status = ‘done’ where Status = ‘at the table’;

Place WBCR: Chips Cylinder is attached to the wall
create view WBCR as
 select *
     from  CHIPS
      where Status = ‘ready’;
```



Transition ISLS: Get the Small Screws or the Large Screws out of the bag

```
routine ISLS (accept SLSU, WRO;
                  return  SLSF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update SCREWS
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place SLSF: Small Screws or Large Screws out of the bag

```
create view SLSF as
 select *
    from  SCREWS
      where Status = 'free';
```

Transition CSLS Put the Small Screws or the Large Screws in a Tray

```
routine CSLS (accept SLSF, WRU;
                   return  SLSY, WRC)
   collectscrews(WR,screws)
   /*
     Collectscrews make the robot hold a tray under the table and aside,
      and in the table there is a pusher positioned vertically that push all
      the screws to the tray.
   */
   waitfor(ProcessTime)
   update SCREWS
     set status = 'in tray' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place SLSY: Small Screws or Large Screws places in tray

```
create view SLSY as
 select *
    from  SCREWS
      where Status = 'in tray';
```

Transition MSLS: Move the Small Screws or the Large Screws to the table

```
routine MSLS (accept SLSY, WRC;
                    return  SLST, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update SCREWS
     set status = ‘at the table’ where Status = ‘in tray’;
   update WAREHOUSE_ROBOTS
     set status = ‘at the table’ where Status = ‘collecting’;
```

Place SLST: Small Screws or Large Screws at the table

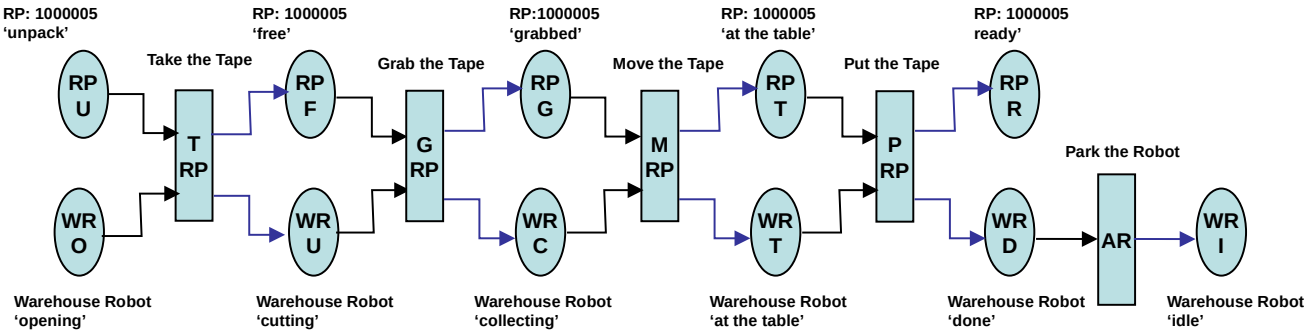
```
create view SLST as
 select *
    from  SCREWS
      where Status = ‘at the table’;
```

Transition PSLS: Put the Small Screws or the Large Screws in a container

```
routine PSLS (accept SLST, WRT;
                    return  SLSR, WRD)
   PutInTray(WR,screws_container)
    /*
      The warehouse robot has a camera and it can recognize the top side of a screw. It
        put the screws in a tray with hole that the screws go in, so they can be taken
        easily by the robot builder.
   */
   waitfor(ProcessTime)
   update SCREWS
     set status = ‘ready’ where Status = ‘at the table’;
   update WAREHOUSE_ROBOTS
     set status = ‘done’ where Status = ‘at the table’;
```

Place SLSR: Small Screws or Large Screws are ready in the tray

```
create view SLSR as
 select *
    from  SCREWS
      where Status = ‘ready’;
```



Transition TRP: Get the Plastic Round Tape out of the bag

```
routine TRP (accept RPU, WRO;
                  return  RPF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update ROUND_PLASTIC
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place RPF: Plastic Round Tape out of the bag

```
create view RPF as
 select *
    from  ROUND_PLASTIC
      where Status = 'free';
```

Transition GRP: Grab the Plastic Round Tape

```
routine GRP (accept RPF, WRU;
                   return  RPG, WRC)
   collectTape(WR,tape)
  /*
     Collecttape make the robot grab the tape.
 */
   waitfor(ProcessTime)
   update ROUND_PLASTIC
     set status = 'grabbed' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place RPG: Plastic Round Tape grabbed by the robot

```
create view RPG as
 select *
    from  ROUND_PLASTIC
      where Status = 'grabbed';
```

Transition MRP: Move Round Plastic Tape to the table

```
routine MRP (accept RPG, WRC;
                    return  RPT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update ROUND_PLASTIC
     set status = ‘at the table’ where Status = ‘grabbed’;
   update WAREHOUSE_ROBOTS
     set status = ‘at the table’ where Status = ‘collecting’;
```

Place RPT: Round Plastic Tape at the table

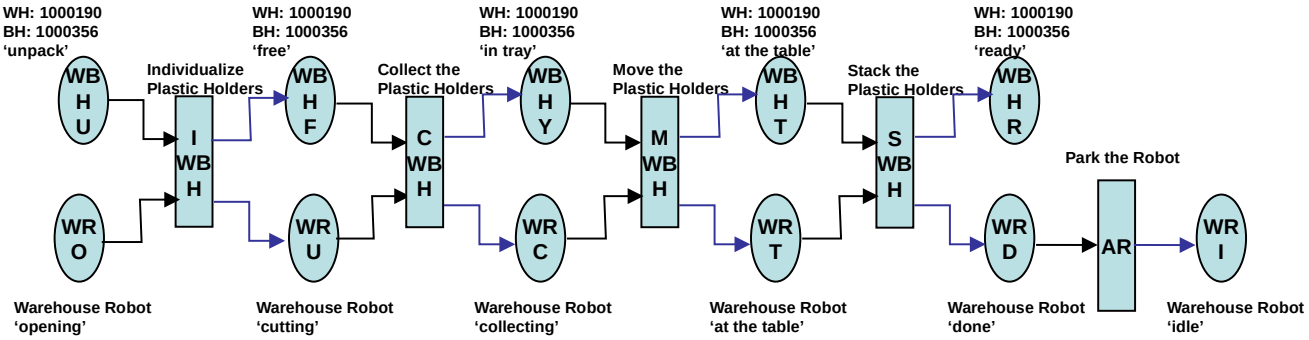
```
create view RPT as
 select *
    from  ROUND_PLASTIC
      where Status = ‘at the table’;
```

Transition PRP: Put the Round Plastic Tape in the device at the wall

```
routine PRP (accept RPT, WRT;
                  return  RPR, WRD)
begin
   Puttapeatwall(WR,tape)
  /*
     Puttapeatwall put the tape in a device at the wall, the device can
       separate the two layers of plastics one upper and one lower layer with
       the round plastic. The device moves one by one the round plastics, so
       the builder robot can take one at a time.
 */
   waitfor(ProcessTime)
   update ROUND_PLASTIC
     set status = ‘ready’ where Status = ‘at the table’;
   update WAREHOUSE_ROBOTS
     set status = ‘done’ where Status = ‘at the table’;
```

Place RPR: Plastic Round Tape at the wall

```
create view RPR as
 select *
    from  PARTS
      where Status = ‘ready’;
```



Transition IWBH: Get the White Plastic Holders or the Black Plastic Holders out of the bag

```
routine IWBH (accept WBHU, WRO;
                  return  WBHF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place WBHF: White Plastic Holders or Black Plastic Holders out of the bag

```
create view WBHF as
 select *
    from  PLASTIC_HOLDERS
      where Status = 'free';
```

Transition CWBH Put the White Plastic Holders or the Black Plastic Holders in a Tray

```
routine CWBH(accept WBHF, WRU;
                    return  WBHY, WRC)
   collectholders(WR,Holders)
   /*
     collectholders make the robot hold a tray under the table and aside,
      and in the table there is a pusher positioned vertically that push all
      the holders to the tray.

  */
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
     set status = 'in tray' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place WBHY: White Plastic Holders or the Black Plastic Holders places in tray

```
create view WBHY as
 select *
   from  PLASTIC_HOLDERS
     where Status = 'in tray';
```

Transition MWBH: Move the White Plastic Holders or the Black Plastic Holders to the table.

```
routine MWBH (accept WBHY, WRC;
                    return  WBHT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
     set status = 'at the table' where Status = 'in tray';
   update WAREHOUSE_ROBOTS
     set status = 'at the table' where Status = 'collecting';
```

Place WBHT: Small Screws or Large Screws at the table

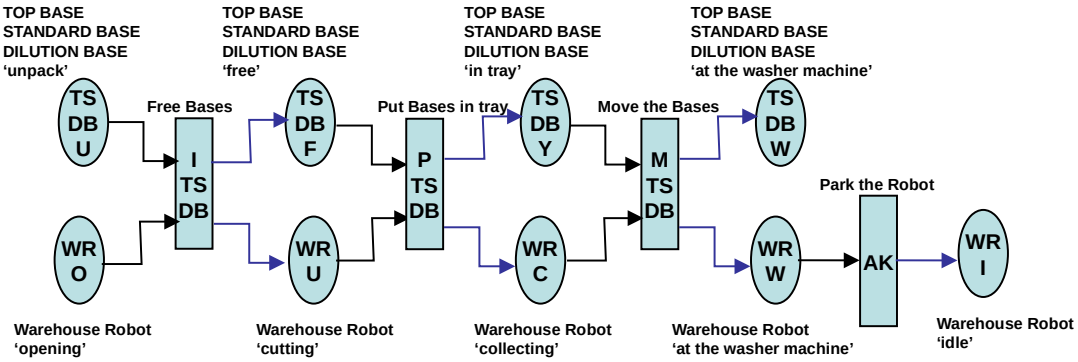
```
create view WBHT as
 select *
   from  PLASTIC_HOLDERS
     where Status = 'at the table';
```

Transition SWBH: Stack the White Plastic Holders or the Black Plastic Holders face up At the table.

```
routine SWBH (accept WBHT, WRT;
                    return  WBHR, WRD)
    stackHolders(WR,plasticHolders)
    /*
     stackHolders make the robot stack all the White Plastic Holders or the Black Plastic
       Holders in the table faced up for building purpose.
   */
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
     set status = 'ready' where Status = 'at the table';
   update WAREHOUSE_ROBOTS
     set status = 'done' where Status = 'at the table';
```

Place WBHR: White Plastic Holders or the Black Plastic Holders are stacked in the table

```
create view WBHR as
 select *
   from  PLASTIC_HOLDERS
     where Status = 'ready';
```




```
Transition ITSDB: Get the Bases out of the wraps
routine ITSDB(accept TSDBU, WRO;
                    return  TSDBF, WRU;
                    guard  qtyBases <= 10)

   emptybases( WR,PlasticHolders)
   /*
      emptybases make the robot cut the wrap from the bases and throw away
      the wrap. It will do 10 bases at a time.
       The transaction itself, like any other transaction where a command to a
       robot has issued and this command require many steps, could be expanded
       in many Transactions/Places to follow the various moves of the robot.
   */
   waitfor(ProcessTime)
   update BASES
      set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
      set status = 'cutting' where Status = 'opening';

Place TSDBF: Bases Free
create view TSDBF as
 select *
    from  BASES
      where Status = 'free';

Transition PTSDB: Put the Bases in a Tray
routine PTSDB(accept TSDBF, WRU;
                    return  TSDBY, WRC)
begin
   putbases(WR,Bases)
   /*
     PutBases make the robot grab the bases one at time and put them
     in a tray.
  */
    waitfor(ProcessTime)
    update BASES
      set status = 'in tray' where Status = 'free';
    update WAREHOUSE_ROBOTS
      set status = 'collecting' where Status = 'cutting';
```

Place TSDBY: Bases in tray

```
create view TSDBY as
 select *
    from  BASES
      where Status = ‘in tray’;
```

Transition MTSDB: Move the Bases to the Washer Machine.

```
routine MTSDB(accept TSDBY, WRC;
                      return  TSDBW, WRW)
   move_robot(WR,washer_machine)
   waitfor(ProcessTime)
   update BASES
      set status = ‘at the washer machine’
         where Status = ‘in tray’;
   update WAREHOUSE_ROBOTS
      set status = ‘at the washer machine’ where Status = ‘collecting’;
```

Place TSDBW: Bases at the washer machine waiting to be cleaned

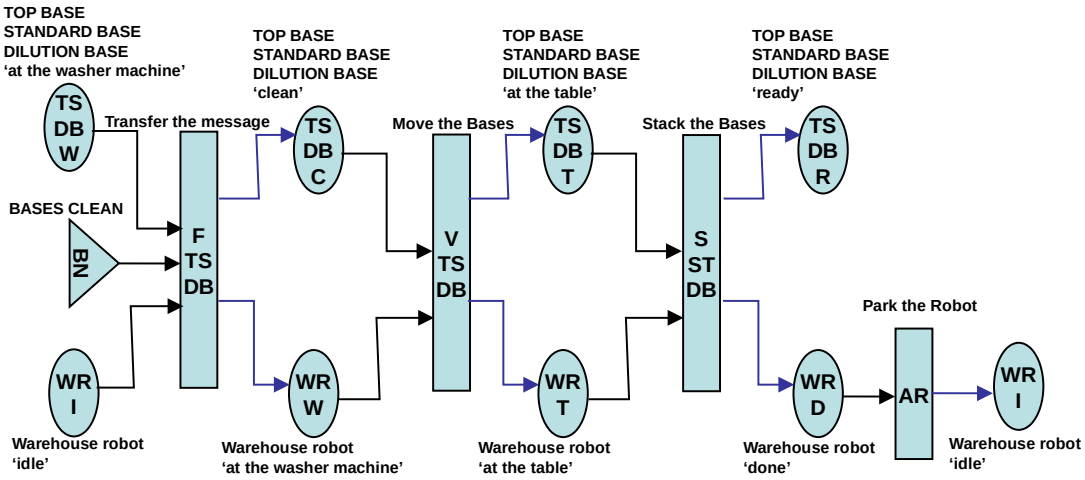
```
create view TSDBW as
 select *
   from  BASES
     where Status = ‘at the washer machine’;
```

Transition AK: Move the Warehouse Robot to its original place

```
routine AK (accept WRW;
                  return  WRI)
   move_robot(WR,corner)
   waitfor(ProcessTime)
   update WAREHOUSE_ROBOTS
      set status = ‘idle’ where Status = ‘at the washer machine’;
```

Place WRW: Warehouse Robot at the washer machine

```
create view WRU as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = ‘at the washer machine’;
```



Message BN: Washer Machine stopped. Bases are clean

Transition FTSDB: Transfer the message
routine FTSDB(accept BN, TSDBW, WRI;
return TSDBC, WRW;
guard qtyBases <= 10)

update BASES
set status = 'clean' where Status = 'at the washer machine';
update WAREHOUSE_ROBOTS
set status = 'at the washer machine' where Status = 'idle';

Place TSDBC: Bases Cleaned
create view TSDBC as
select *
from BASES
where Status = 'clean';

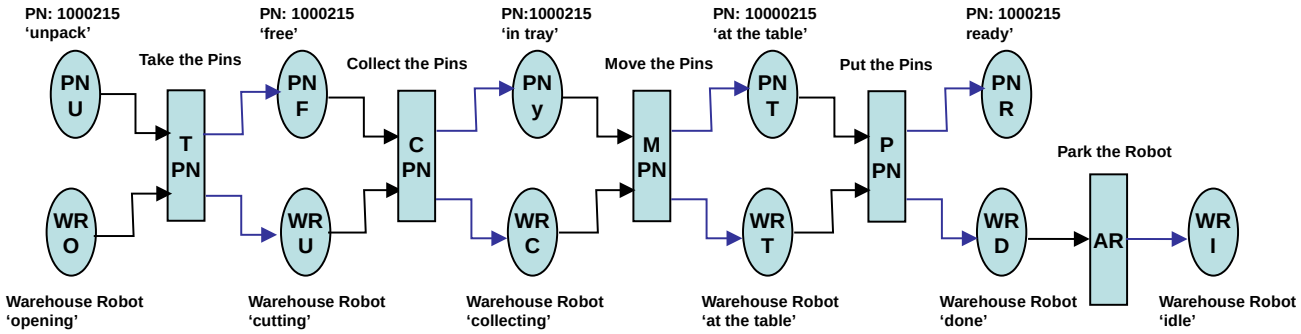
Transition VTSDB: Move the Bases to the table
routine VTSDB(accept TSDBC, WRW;
return TSDBT, WRT)
move_robot(WR,table_location)
waitfor(ProcessTime)
update BASES
set status = 'at the table' where Status = 'clean';
update WAREHOUSE_ROBOTS
set status = 'at the table' where Status = 'at the washer machine';

Place TSDBT: Bases at the table
create view TSDBT as
select *
from BASES
where Status = 'at the table';

Transition STSDB: Stack the Bases in the table.
routine STSDB(accept TSDBT, WRT;
return TSDBR, WRD)
stackbases(WR,Bases)
/*
stackbases make the robot stack the bases in the table faced up for building purpose.
***/**
waitfor(ProcessTime)
update BASES
set status = ‘ready’ where Status = ‘at the table’;
update WAREHOUSE_ROBOTS
set status = ‘done’ where Status = ‘at the table’;

Place TSDBR: Bases ready at the table
create view TSDBR as
select *
from BASES
where Status = ‘ready’;

/* The washer machine should be in the same plane as the warehouse robot to avoid the use of an elevator. The warehouse robot should have its own route to avoid obstacles recognition and avoid *|



Transition TPN: Get the Pins out of the bag

```
routine TPR (accept PNU, WRO;
                  return  PNF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update PINS
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place PNF: Pins out of the bag

```
create view PNF as
 select *
    from  PINS
      where Status = 'free';
```

Transition CPN: Collect the pins

```
routine CPN (accept PNF, WRU;
                   return  PNY, WRC)
  collectpins(WR,pins)
  /*
     collectpins make the robot hold a tray under the table and aside,
      and in the table there is a pusher positioned vertically that push all
      the pins to the tray.
 */
   waitfor(ProcessTime)
   update PINS
     set status = 'in tray' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place PNY: Pins in a tray

```
create view PNY as
 select *
    from  PINS
      where Status = 'in tray';
```

Transition MPN: Move pins to the table

```
routine MPN (accept MPY, WRC;
                    return  MPT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update PINS
     set status = ‘at the table’ where Status = ‘in tray’;
   update WAREHOUSE_ROBOTS
     set status = ‘at the table’ where Status = ‘collecting’;
```

Place PNT: Pins at the table

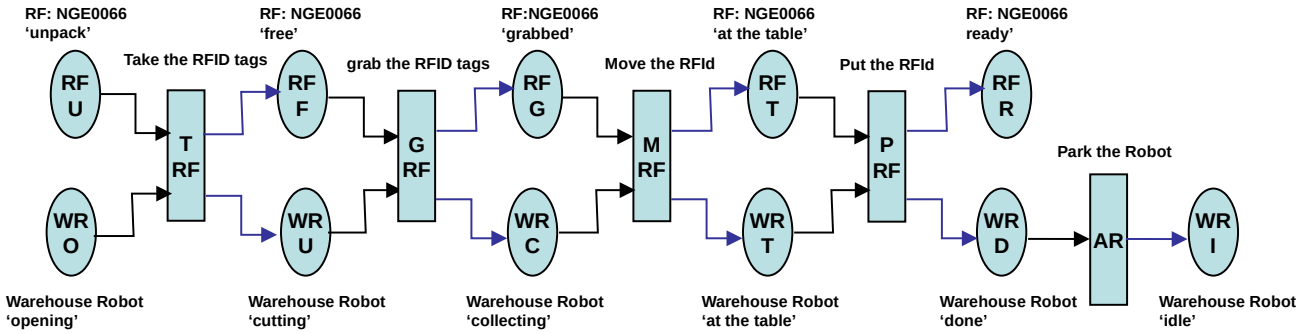
```
create view PNT as
 select *
    from  PINS
     where Status = ‘at the table’;
```

Transition PPN: Put the Pins in a tray

```
routine PPR (accept PRT, WRT;
                  return  PRR, WRD)
   PutInTray(WR,pins)
    /*
      PutInTray make the robot put all the pins in a tray for building purpose.
   */
   waitfor(ProcessTime)
   update PINS
     set status = ‘ready’ where Status = ‘at the table’;
   update WAREHOUSE_ROBOTS
     set status = ‘done’ where Status = ‘at the table’;
```

Place PNR: Pins are ready in container

```
create view PNR as
 select *
    from  PINS
     where Status = ‘ready’;
```



Transition TRF: Get the RFId Tags out of the bag

```
routine TRF (accept RFU, WRO;
                  return  RFF, WRU)
   emptybag( WR,bag)
   waitfor(ProcessTime)
   update RFID_TAGS
     set status = 'free' where Status = 'unpack';
    update WAREHOUSE_ROBOTS
     set status = 'cutting' where Status = 'opening';
```

Place RFF: RFId out of the bag

```
create view RFF as
 select *
    from  RFID_TAGS
      where Status = 'free';
```

Transition GRF: Grab the RFID Tags

```
routine GRF (accept RFF, WRU;
                   return  RFG, WRC)
  grabtags(WR,tags)
  /*
     grabtags make the robot grab the cylinder packet of RFID tags.
 */
   waitfor(ProcessTime)
   update RFID_TAGS
     set status = 'grabbed' where Status = 'free';
   update WAREHOUSE_ROBOTS
     set status = 'collecting' where Status = 'cutting';
```

Place RFG: RFID tags grabbed by the robot

```
create view RFG as
 select *
    from  RFID_TAGS
      where Status = 'grabbed';
```

```
Transition MRF: Move RFID tags to the table
routine MRF (accept RFG, WRC;
                    return  RFT, WRT)
   move_robot(WR,table_location)
   waitfor(ProcessTime)
   update RFID_TAGS
     set status = ‘at the table’ where Status = ‘grabbed’;
   update WAREHOUSE_ROBOTS
     set status = ‘at the table’ where Status = ‘collecting’;

Place RFT: RFID tags at the table
create view RFT as
 select *
    from  RFID_TAGS
      where PARTS.Status = ‘at the table’;

Transition PRF: Put the RFID tags packet cylinder in a device to the wall
routine PRF (accept RFT, WRT;
                  return  RFR, WRD)
   PutToTheWall(WR,tags)
    /*
      PutToTheWall make the robot put the packet cylinder of RFId tags in a device to the
        wall.
   */
   waitfor(ProcessTime)
   update RFID_TAGS
     set status = ‘ready’ where Status = ‘at the table’;
   update WAREHOUSE_ROBOTS
     set status = ‘done’ where Status = ‘at the table’;

Place RFR: RFId Tags at the wall
create view RFR as
 select *
    from  RFID_TAGS
      where PARTS.Status = ‘ready’;
```

Building the cartridge

Alan El-Sabbagh WHATSAPP: +971551574255

HAND ROBOT

The Hand Robot to build the NEXGEN could have at least 6 degrees of freedom:

- BASE: turn to the left or right in circular mode from -180 degrees to 180 degrees (Initially 0 degree)

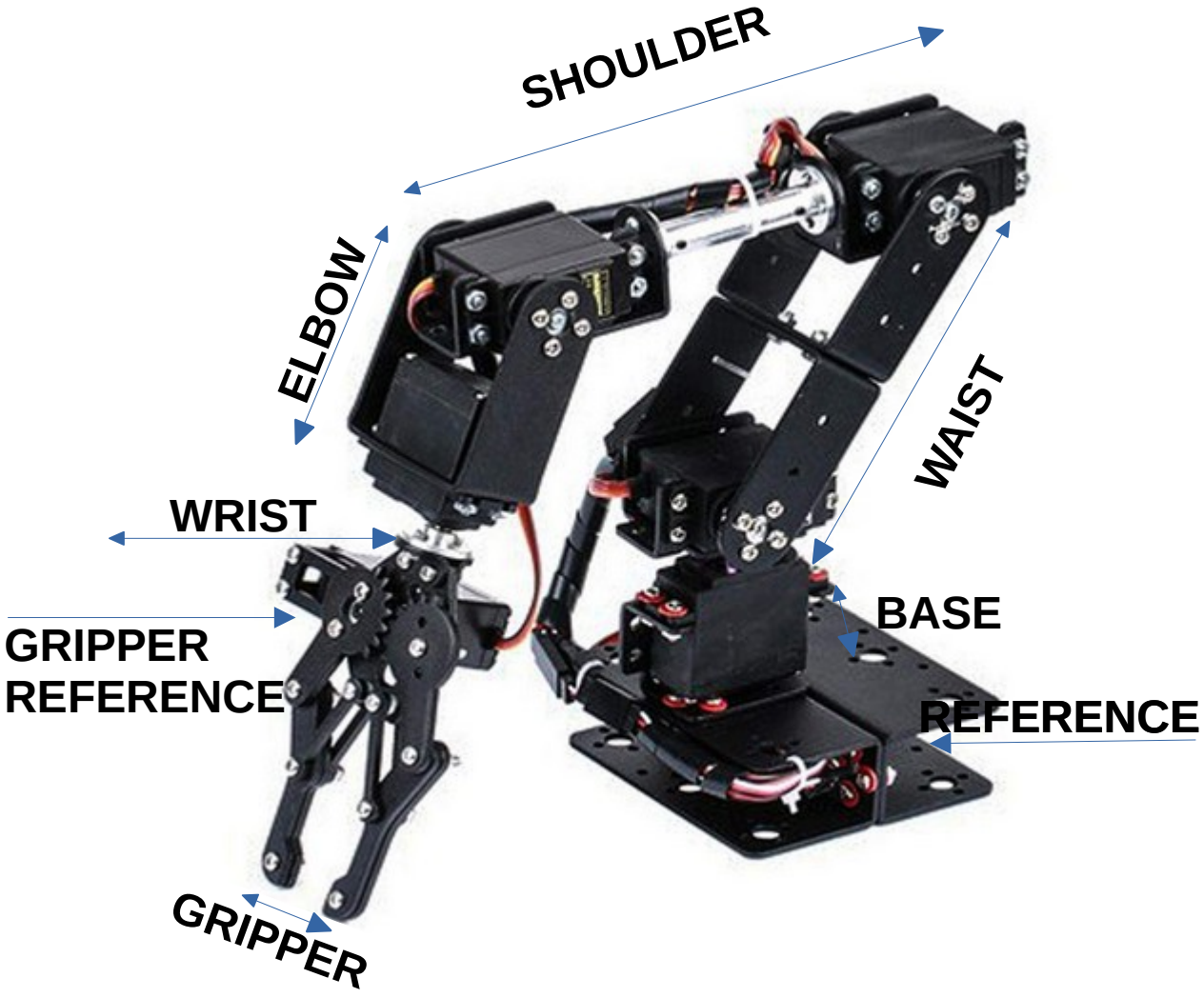
- WAIST: go Forward and Backward forming an angle with the REFERENCE from -180 degrees to 180 degrees (Initially 90 degrees: vertically standing)

- SHOULDER: turn Up and Down forming an angle with the WAIST from 0 to 360 degrees (Initially 90 degrees: perpendicular to WAIST)

- ELBOW: go Forward and Backward forming an angle with the SHOULDER from -180 to 180 degrees (Initially 0 degree: linear to SHOULDER)

- WRIST: turn circularly around by 180 degrees (Initially 0 degree)

- GRIPPER: works by Opening or Closing to "grab things" forming an angle with the GRIPPER REFERENCE from 0 to 90 degrees (Initially 90 degree: totally close)



Commands for HAND ROBOT

The computer use the following commands to communicate with the hand robot. The parameters to the functions are in degrees, but in real implementation could be in cm, ft, steps ... The only way to know the exact values for a move is by testing and trials. One time knowing the exact values should be saved in the program.

The basic commands are:
- **SetBase (angle)**
- **SetWaist (angle)**
- **SetShoulder (angle)**
- **SetElbow (angle)**
- **SetWrist (angle)**
- **SetGripper (angle)**

A general command is used for simplicity:
- **SetArm (base_angle, waist_angle, shoulder_angle, elbow_angle, wrist_angle, gripper_angle)**

This command is implemented from the sixth basic commands respectively

The plastic base as a holder for the cartridge is right under the robot. The robot is in the middle of the table. On its left side, there is white plastic holders, black plastic holders, RFID tags, non dilution bases, dilution bases, small screws, small rings, big rings, pins and metal rings. On the right side there is 1000002 chips, 1000003 chips, 1000005 round plastic, top bases, large screws and labels.

The high level functions are:

Move (ROBOT, PLACE)
Grab (ROBOT, GADGET)
Put (ROBOT, GADGET, PLACE)
Flip (ROBOT, GADGET)
Push(TOOL, GADGET, DISTANCE)
Peel(ROBOT, GADGET)
Release(ROBOT, GADGET)

Move (ROBOT, StartPosition):

Normally this function is achieved by sending all angles to the initial values.

Move (ROBOT, Place):

This function can be achieved by try and test different angles to get the values. After this the values have to be saved and used to put them in the program to go to the specific place

Grab (ROBOT, GADGET)

This function can be achieved by try and test with different angles in SetGripper(angle) to get the right angle

Put (ROBOT, GADGET, PLACE)

This function is achieved by moving down the hand robot to the place and putting the gadget

Flip (ROBOT, GADGET)

This function is achieved by calling the function SetWrist (π)

Push (TOOL, GADGET, DISTANCE)

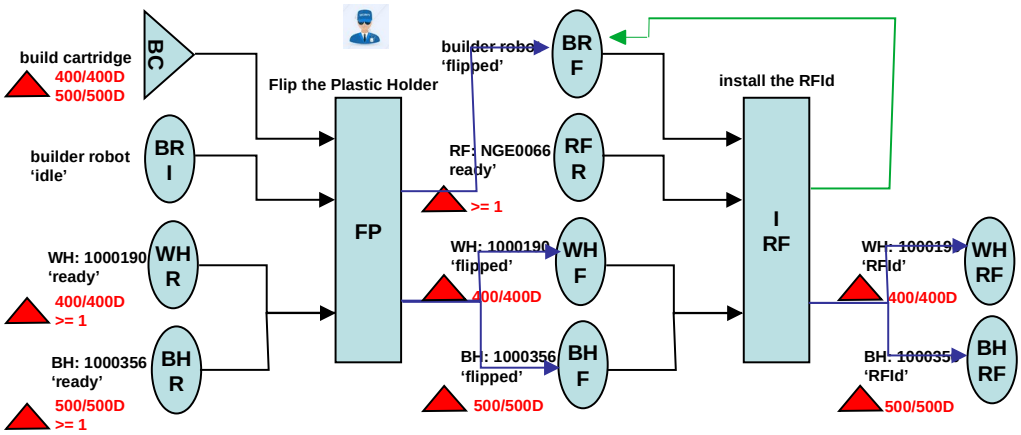
Move the tool in a direction by a distance to push a gadget

Peel (ROBOT, GADGET)

Use the hand robot to peel

Release (ROBOT, GADGET)

This function can be achieved by using SetGripper(angle) to open and release the gadget



BC: Message to build the cartridge with type of cartridge to build:
400 / 400D / 500 / 500D

Transition FP: Flip the Plastic Holder

```
/* The hand robot grab the White or Black Plastic Holder and turn it 180 degrees */
routine FP (accept BC,
                    BRI,
                    WHR guard 400 / 400D to build message in BC,
                              count(WHR) >= 1,
                    BHR  guard 500 / 500D to build message in BC,
                              count(BHR) >= 1;
            return  BRF,
                    WHF guard 400 / 400D to build message in BC,
                    BHF  guard 500 / 500D to build message in BC)
  update BUILDER_ROBOT
    set Status = 'in process';
  case (message in BC is to build 400 / 400D):
    select max(Id) into :IdPlastic_Holder
      from PLASTIC_HOLDERS
        where (Code_Part = '1000190') and
              (Status = 'ready');
    update PLASTIC_HOLDERS
      set Status = 'in process'
        where (Code_Part = '1000190') and
              (Id = :IdPlastic_Holder);
    move(BR,WPHContainer)
      /*  Move the Hand Robot to the container with White Plastic Holders */
    waitfor(ProcessTime)
    grab(BR, WPH)
      /* The Hand Robot grab one White Plastic Holder */
    waitfor(ProcessTime)
    flip(BR,WPH)
      /* The hand Robot turn the wrist 180 degrees */
    waitfor(ProcessTime)
    update PLASTIC_HOLDERS
      set Status = 'flipped'
        where (PARTS.Code = '1000190') and
              (Id = :IdPlastic_Holder);
```

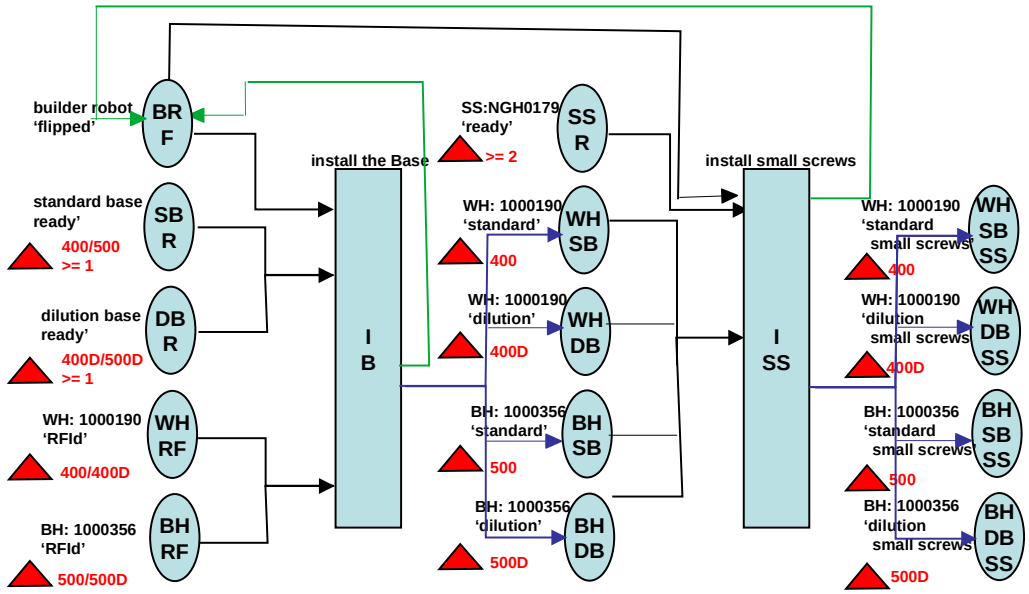
```
   case (message in BC is to build 500 / 500D):
      select max(Id) into :IdPlastic_Holder
        from PLASTIC_HOLDERS
          where (Code_Part = '1000356') and
                (Status = 'ready');
      update PLASTIC_HOLDERS
       set Status = 'in process'
          where (Code_Part = '1000356') and
                (Id = :IdPlastic_Holder);
       move(BR,BPHContainer)
          /*  Move the Hand Robot to the container with Black Plastic Holders */
       waitfor(ProcessTime)
       grab(BR, BPH)
          /* The Hand Robot grab one Black Plastic Holder */
       waitfor(ProcessTime)
       flip(BR,BPH)
          /* The hand Robot turn the wrist 180 degrees */
       waitfor(ProcessTime)
       update PLASTIC_HOLDERS
          set Status = 'flipped'
             where (PARTS.Code = '1000356') and
                   (Id = :IdPlastic_Holder);
   update BUILDER_ROBOT
      set Status = 'flipped';

Place BRF: Builder Robot Flipped
create view BRF as
 select *
    from  BUILDER_ROBOTS
       where (Status = 'flipped');

Place WHF: White Plastic Holder Flipped
create view WHF as
  select *
    from  PLASTIC_HOLDERS
      where (Code_Part = '1000190') and
               (Id = :IdPlastic_Holder) and
               (Status = 'flipped');

Place BHF: Black Plastic Holder Flipped
create view BHF as
  select *
    from  PLASTIC_HOLDERS
      where (Code_Part = '1000356') and
              (Id = :IdPlastic_Holder) and
              (Status = 'flipped');
```

Transition IRF: Insert the RFId Tag

/* The Builder Robot put the plastic holder in the base under it. The computer give a command to the RFId Stack Device to move out one RFId Tag. This is done by pushing the RFId plastic tube holder from the back where a steel tube having a form of reversed "s" penetrate into the open hole of the plastic tube and push the RFId tag as shown in the picture. The Robot Builder peel the RFId Tag and attach it to the Plastic Holder */

```
routine IRF (accept BRF,
                    RFR guard count(RFR) >= 1,
                    WHF guard 400 / 400D to build message in BC,
                    BHF  guard 500 / 500D to build message in BC;
             return  BRF,
                    WHRF guard 400 / 400D to build message in BC,
                    BHRF  guard 500 / 500D to build message in BC)
  update BUILDER_ROBOT
     set Status = 'in process';
  update PLASTIC_HOLDERS
     set Status = 'in process'
        where (Id = :IdPlasticHolder);
  select max(Id) into :IdRFId_Tag
     from RFID_TAGS
       where (Code_Part = 'NGE0066') and
             (Status = 'ready');

  move(BR,Start_Position)
  /*  Move the Hand Robot to the Start Position */
  waitfor(ProcessTime)
  move(BR,PlaticHolderBase)
  /*  Move the Hand Robot to the Plastic Holder Base under it */
  waitfor(ProcessTime)
  release(BR,Plastic_Holder)
  /*  The Hand Robot release the Plastic Holder to the Base */
  waitfor(ProcessTime)
  move(BR,RFIdTagsHolder)
  /*  Move the Hand Robot to the RFId Tags Holder */
  waitfor(ProcessTime)
  push(SteelTube,RFIdTag,halfway)
  /*  The computer give command to the Steel Tube to move
       one RFId Tag forward halfway */
  waitfor(ProcessTime)
  peel(BR,RFIdTag)
  /* The Builder Robot peel the RFId Tag */
  waitfor(ProcessTime)
 grab(BR,RFId_Tag)
  /*  The computer grab the RFId Tag */
  waitfor(ProcessTime)
  move(BR,PlaticHolderBase)
  /*  Move the Hand Robot to the Plastic Holder Base under it */
  waitfor(ProcessTime)
  put(BR,RFId_Tag,PlasticHolder)
  /*  The Hand Robot put the RFId Tag over Plastic Holder */
  waitfor(ProcessTime)
```

```
    release(BR,RFId_Tag)
    /*  The Hand Robot release the RFId_Tag */
    waitfor(ProcessTime)

   update PLASTIC_HOLDERS
     set Status = 'RFId'
       where (Id = :IdPlastic_Holder);
    update RFID_TAGS
     set Code_Plastic_Holder = :IdPlastic_Holder,
          Status = 'plastic holder'
       where Id = :IdRFId_Tag;
   update BUILDER_ROBOT
      set Status = 'flipped';

Place WHRF: White Plastic with RFId
create view WHRF as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'RFId');

Place BHRF: Black Plastic with RFId
create view BHRF as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFIdTag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (PLASTIC_HOLDERS.Status = 'RFId');
```



Transition IB: Install the Base

/* The bases are stacked with the face up for the holes (holes for the rings, Pin). This is done before by the warehouse robot that has a camera to recognize the faces of the bases. The 2 holes for the small screws are toward the wall.
The robot bring the plastic holder in the right place between the base.
An arm push the base toward the plastic holder.
****/***

```
routine IB (accept BRF,
                    SBR, guard 400 / 500      to build message in BC guard count(SBR) >= 1,
                    DBR, guard 400D / 500D to build message in BC guard count(DBR) >= 1
                    WHRF guard 400 / 400D to build message in BC,
                    BHRF  guard 500 / 500D to build message in BC;
            return  BRF,
                     WHSB guard 400   to build message in BC,
                     WHDB guard 400D to build message in BC,
                     BHSB  guard 500D to build message in BC,
                     BHDB  guard 500D to build message in BC)
   update BUILDER_ROBOT
      set Status = 'in process';
    move(BR,Start_Position)
    /*  Move the Hand Robot to the Start Position */
    waitfor(ProcessTime)
    move(BR,PlaticHolderBase)
    /*  Move the Hand Robot to the Plastic Holder Base under it */
    waitfor(ProcessTime)
    grab(BR,Plastic_Holder)
    /*  The Hand Robot grab the Plastic Holder from the Base */
    waitfor(ProcessTime)
```

```
case (message in BC is to build 400):
    select max(Id) into :IdBase
      from BASES
        where (Code_Part = '1000???'); /* 1000??? stand for codestandardbase */
   move(BR,StandardBaseContainer)
    /*  Move the Hand Robot to the container with standard bases */
   waitfor(ProcessTime)
   push(SteelTube,StandardBase,halfway)
   /*  The computer give command to the Steel Tube to move
         the standard base forward halfway */
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
      set Status = 'standard'
         where (Code_Part = '1000190') and
               (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
    select max(Id) into :IdBase
      from BASES
        where (Code_Part = '1000???'); /* 1000??? stand for codedilutionbase */
   move(BR,DilutionBaseContainer)
    /*  Move the Hand Robot to the container with dilution bases */
   waitfor(ProcessTime)
   push(SteelTube,DilutionStandardBase,halfway)
   /*  The computer give command to the Steel Tube to move
         the dilution base forward halfway */
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
      set Status = 'dilution'
         where (Code_Part = '1000190') and
               (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
    select max(Id) into :IdBase
      from BASES
        where (Code_Part = '1000???'); /* 1000??? stand for codestandardbase */
   move(BR,StandardBaseContainer)
    /*  Move the Hand Robot to the container with standard bases */
   waitfor(ProcessTime)
   push(SteelTube,StandardBase,halfway)
   /*  The computer give command to the Steel Tube to move
         the standard base forward halfway */
   waitfor(ProcessTime)
   update PLASTIC_HOLDERS
      set Status = 'standard'
         where (Code_Part = '1000356') and
               (Id = :IdPlastic_Holder);
```

```
case (message in BC is to build 500D):
     select max(Id) into :IdBase
       from BASES
          where (Code_Part = '1000???'); /* 1000??? stand for codedilutionbase
     move(BR,DilutionBaseContainer)
     /*  Move the Hand Robot to the container with dilution bases */
    waitfor(ProcessTime)
    push(SteelTube,DilutionStandardBase,halfway)
    /*  The computer give command to the Steel Tube to move
           the dilution base forward halfway */
    waitfor(ProcessTime)
    update PLASTIC_HOLDERS
       set Status = 'dilution'
          where (Code_Part = '1000356') and
                  (Id = :IdPlastic_Holder);
 move(BR,Start_Position)
 /*  Move the Hand Robot to the Start Position */
 waitfor(ProcessTime)
 move(BR,PlaticHolderBase)
 /*  Move the Hand Robot to the Plastic Holder Base under it */
 waitfor(ProcessTime)
 release(BR,Plastic_Holder)
 /*  The Hand Robot release the Plastic Holder to the Base */
 waitfor(ProcessTime)
update BASES
  set Code_Plastic_Holder = :IdPlastic_Holder,
       Status = 'plastic holder'
     where Id = :IdBase;
update BUILDER_ROBOT
   set Status = 'flipped';
```

Place WHSB: White Plastic Holder with Standard Base inserted

```
create view WHSB as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard');
```

Place WHDB: White Plastic Holder with Dilution Base inserted

```
create view WHDB as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'dilution');
```

Place BHSB: Black Plastic Holder with Standard Base inserted

```
create view BHSB as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard');
```

Place BHDB: Black Plastic Holder with Dilution Base inserted

```
create view BHDB as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFID_TAGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'dilution');
```

```
Transition ISS: Install the Small Screws
/* The Plastic Holder is in the Plastic Base under the builder robot. The Builder Robot
   move to the Small Screws Container, take one screw at a time and put it in the right
   hole. There is two Screw Drivers on a track positioned above the holes for the
   small screws. The track can move via pipes down and up to the screws to screws
   them.
*/
routine ISS (accept BRF,
                        SSR     guard  count(SSR) >= 2,
                        WHSB guard 400    to build message in BC,
                        WHDB guard 400D to build message in BC,
                        BHSB  guard 500    to build message in BC,
                        BHDB  guard 500D to build message in BC;
             return   BRF,
                        WHSBSS guard 400   to build message in BC,
                        WHDBSS guard 400D to build message in BC,
                        BHSBSS  guard 500D to build message in BC,
                        BHDBSS  guard 500D to build message in BC )
   select Id into :IdRightScrew, :IdLeftScrew
         from SCREWS
          order by id desc
          fetch first 2 rows only
           where (Code_Part = 'NGH0179');
   update BUILDER_ROBOT
      set Status = 'in process';
   Move(BR,ScrewsContainer);
    /*  Move the Hand Robot to the Small Screws container */
   waitfor(ProcessTime)
   grab(BR,SmallScrew)
    /*  Get one small screw */
   waitfor(ProcessTime)
   Move(BR,LeftHoleScrew);
    /*  Move the Hand Robot to the left hole for the small screw in the Plastic Holder Base
   */
   put(BR,SmallScrew,LeftSideHole)
    /*  Put the small screw into the left hole */
   waitfor(ProcessTime)
   Move(BR,ScrewsContainer);
    /*  Move the Hand Robot to the Small Screws container */
   waitfor(ProcessTime)
   grab(BR,SmallScrew)
    /*  Get one small screw */
   waitfor(ProcessTime)
   Move(BR,RightHoleScrew);
    /*  Move the Hand Robot to the right hole for the small screw in the Plastic Holder
       base */
   put(BR,SmallScrew,RightSideHole)
    /*  Put the small screw into the right hole */
   waitfor(ProcessTime)
```

```
command(smallscrewdrivers);
 /*  Send a command to the pipes holding the screw drivers to move down and after
     the screw drivers screwing to move up */
waitfor(ProcessTime)
move(BR,Start_Position)
/*  Move the Hand Robot to the Start Position */
waitfor(ProcessTime)
case (message in BC is to build 400):
   update PLASTIC_HOLDERS
      set Status = ‘standard small screws’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
   update PLASTIC_HOLDERS
      set Status = ‘dilution small screws’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = ‘standard small screws’
         where (Code_Part = ‘1000356’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500D):
   update PLASTIC_HOLDERS
      set Status = ‘dilution small screws’
         where (Code_Part = ‘1000356’) and
                  (Id = :IdPlastic_Holder);
update SCREWS
  set Code_Plastic_Holder = :IdPlastic_Holder,
      Status = ‘plastic holder’
    where Id = (:IdRightScrew) or (Id = :IdLeftScrew);
update BUILDER_ROBOT
   set Status = ‘flipped’;
```

Place WHSBSS: White Plastic Holder with Standard Base and small Screws inserted

```
create view WHSBSS as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS
      where (PLASTIC_HOLDERS.Code_Part = ‘1000190’) and
               (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
               (RFID_TAGS.Id = :IdRFId_Tag) and
               (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (BASES.Id = :IdBase) and
               (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               ((SCREWS.Id = :IdRightScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((SCREWS.Id = :IdLeftScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
               (PLASTIC_HOLDERS.Status = ‘standard small screws’);
```

Place WHDBSS: White Plastic Holder with Dilution Base and small Screws inserted

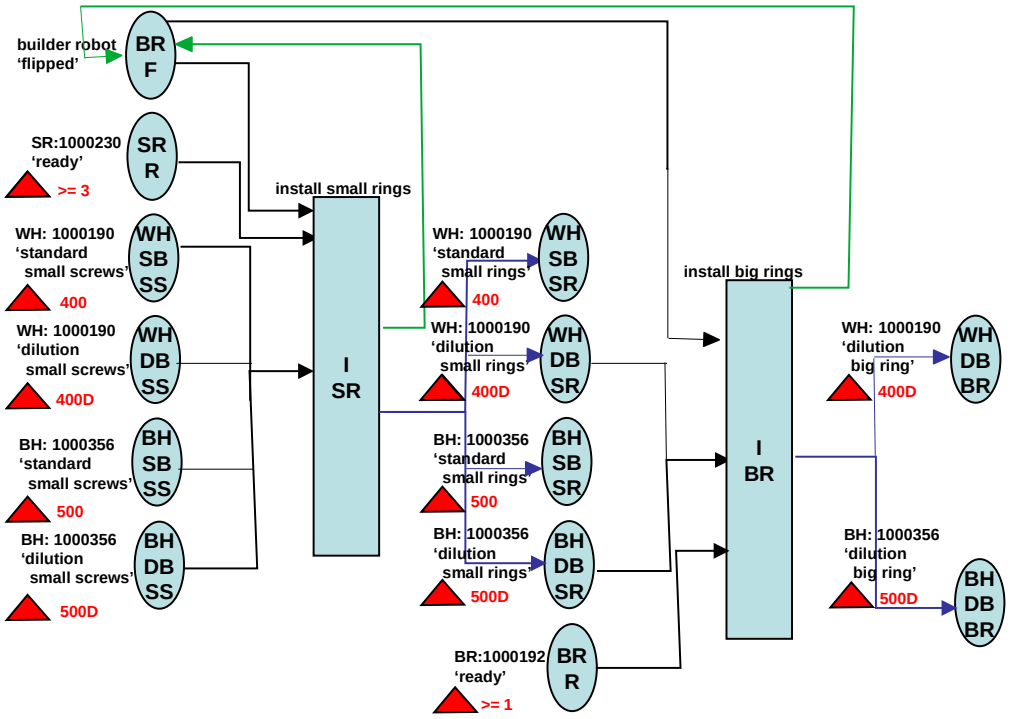
```
create view WHSBSS as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PLASTIC_HOLDERS.Status = 'dilution small screws');
```

Place BHSBSS: Black Plastic Holder with Standard Base and small Screws inserted

```
create view BHSBSS as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PLASTIC_HOLDERS.Status = 'standard small screws');
```

Place BHDBSS: Black Plastic Holder with Dilution Base and small Screws inserted

```
create view BHDBSS as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PLASTIC_HOLDERS.Status = 'dilution small screws');
```



Transition ISR: Install 3 small rings

/* The Plastic Holder is in the Plastic Base under the builder robot. The Builder Robot grab one small ring at a time and put it in the right hole. The rings are on a plastic tube maybe with a gap between them and the distance between one ring and another on the plastic tube is calculated and saved in the computer. Every time a ring is grabbed, the next time the builder robot go to the same place to grab the next ring after adding the saved distance.

***/**

```
routine ISR (accept  BRF,
                     SRR        guard count(SRR) >= 3,
                     WHSBSS guard 400     to build message in BC,
                     WHDBSS guard 400D  to build message in BC,
                     BHSBSS guard 500      to build message in BC,
                     BHDBSS guard 500D   to build message in BC;
             return  BRF,
                    WHSBSR guard 400   to build message in BC,
                    WHDBSR guard 400D to build message in BC,
                    BHSBSR  guard 500    to build message in BC,
                    BHDBSR  guard 500D to build message in BC)
  update BUILDER_ROBOT
     set Status = 'in process';
  update PLASTIC_HOLDERS
    set Status = 'in process'
      where Id = :IdPlastic_Holder;
  select Id into :IdRightRing, :IdLeftRing, :IdPinRing
       from RINGS
         order by id desc
         fetch first 3 rows only
          where (Code_Part = '1000230');
```

```
Move(BR,SmallRingsTube);
 /*  Move the Hand Robot to the Small Rings tube */
waitfor(ProcessTime)
grab(BR,SmallRing)
 /*  Get one small ring */
waitfor(ProcessTime)
Move(BR,LeftHoleSmallRing);
 /*  Move the Hand Robot to the left hole for the small ring in the Base  */
put(BR,SmallRing,LeftSideHole)
 /*  Put the small ring into the left hole */
waitfor(ProcessTime)
Move(BR,SmallRingsTube);
 /*  Move the Hand Robot to the Small Rings tube */
waitfor(ProcessTime)
grab(BR,SmallRing)
 /*  Get one small ring */
waitfor(ProcessTime)
Move(BR,rightHoleSmallRing);
 /*  Move the Hand Robot to the right hole for the small ring in the Base  */
put(BR,SmallRing,RightSideHole)
 /*  Put the small ring into the right hole */
waitfor(ProcessTime)
Move(BR,SmallRingsTube);
 /*  Move the Hand Robot to the Small Rings tube */
waitfor(ProcessTime)
grab(BR,SmallRing)
 /*  Get one small ring */
waitfor(ProcessTime)
Move(BR,MiddleHoleSmallRing);
 /*  Move the Hand Robot to the middle hole for the small ring in the Base  */
put(BR,SmallRing,MiddleSideHole)
 /*  Put the small ring into the middle hole */
waitfor(ProcessTime)
move(BR,Start_Position)
/*  Move the Hand Robot to the Start Position */
waitfor(ProcessTime)
case (message in BC is to build 400):
   update PLASTIC_HOLDERS
      set Status = ‘standard small rings’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
   update PLASTIC_HOLDERS
      set Status = ‘dilution small rings’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = ‘standard small rings’
         where (Code_Part = ‘1000356’) and
                  (Id = :IdPlastic_Holder);
```

```
case (message in BC is to build 500D):
    update PLASTIC_HOLDERS
        set Status = ‘dilution small rings’
            where (Code_Part = ‘1000356’) and
                    (Id = :IdPlastic_Holder);
update RINGS
  set Code_Plastic_Holder = :IdPlastic_Holder,
        Status = ‘plastic holder’
     where Id = (:IdRightRing) or (Id = :IdLeftRing) or (Id = :IdPinRing);
update BUILDER_ROBOT
    set Status = ‘flipped’;
```

```
Place WHSBSR: White Plastic Holder with Standard Base and
                    Small Rings inserted
create view WHSBSR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'standard small rings');

Place WHDBSR: White Plastic Holder with Dilution Base and
                    Small Rings inserted
create view WHDBSR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlasticHolder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'dilution small rings');
```

```
Place BHSBSR: Black Plastic Holder with Standard Base and
                    Small Rings inserted
create view BHSBSR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'standard small rings');

Place BHDBSR: Black Plastic Holder with Dilution Base and
                    Small Rings inserted
create view BHDBSR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'dilution small rings');
```

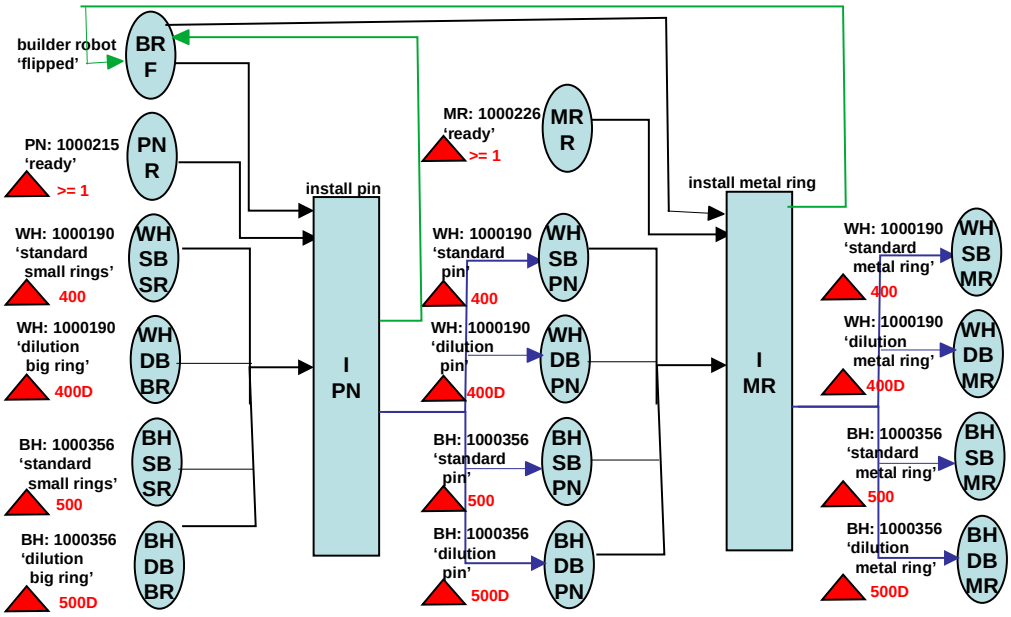
```
Transition IBR: Install big ring
/* The Plastic Holder is in the Plastic Base under the builder robot. The Builder Robot
   grab one large ring and put it in the right hole. The rings are on a plastic tube maybe
   with a gap between them and the distance between one ring and another on the
   plastic tube is calculated and saved in the computer. Every time a ring is grabbed, the
   next time the builder robot go to the same place to grab the next ring after adding the
   saved distance.
*/
routine IBR (accept BRF         guard count(BRF) >= 1,
                                WHDBSR guard 400D  to build message in BC,
                                BHDBSR guard 500D   to build message in BC,
                                BRR;
                    return  BRF,
                                WHDBBR guard 400D to build message in BC,
                                BHDBBR  guard 500D to build message in BC)
   update BUILDER_ROBOT
      set Status = 'in process';
   update PLASTIC_HOLDERS
     set Status = 'in process'
        where Id = :IdPlastic_Holder;
   select Max(Id) into :IdBigRing
           where (Code_Part = '1000192');
   Move(BR,BigRingsTube);
    /*  Move the Hand Robot to the Big Rings tube */
   waitfor(ProcessTime)
   grab(BR,BigRing)
    /*  Get one big ring */
   waitfor(ProcessTime)
   Move(BR,HoleBigRing);
    /*  Move the Hand Robot to the hole for the big ring in the Base  */
   put(BR,BigRing,Hole)
    /*  Put the big ring into the hole */
   waitfor(ProcessTime)
   move(BR,Start_Position)
   /*  Move the Hand Robot to the Start Position */
   waitfor(ProcessTime)
   case (message in BC is to build 400D):
      update PLASTIC_HOLDERS
         set Status = 'dilution big ring'
            where (Code_Part = '1000190') and
                    (Id = :IdPlastic_Holder);;
   case (message in BC is to build 500D):
      update PLASTIC_HOLDERS
         set Status = 'dilution big ring'
            where (PARTS.Code = '1000356');
   update RINGS
     set Code_Plastic_Holder = :IdPlastic_Holder,
         Status = 'plastic holder'
       where Id = (:IdBigRing);
   update BUILDER_ROBOT
      set Status = 'flipped';
```

Place WHDBSR: White Plastic Holder with Dilution Base and Big Ring inserted

```
create view WHSBBR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution big ring');
```

Place BHDBBR: Black Plastic Holder with Dilution Base and Big Ring inserted

```
create view BHDBBR as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFIdTag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution big ring');
```

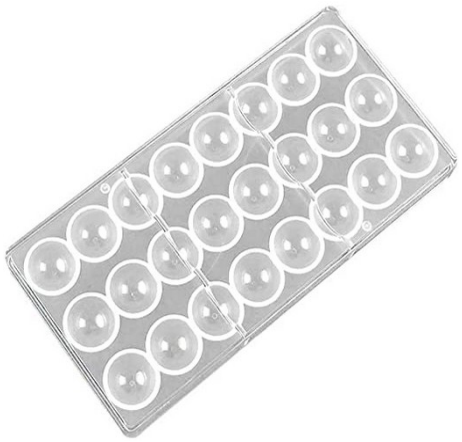


Transition IPN: Install the Pin

/* T*he PINS, pointed to the top, places on a tray with holes to put them in place.* The Plastic Holder is in the Plastic Base under the builder robot. The builder robot grab one PIN at a time and put it in the right hole. Because the PINS are all at the same distance from each other horizontally and at the same distance vertically it is easy to locate the next PIN to grab.

***/**

```
routine IPN (accept BRF,
                    PNR        guard count(PNR) >= 1,
                    WHSBSR guard 400     to build message in BC,
                    WHDBBR guard 400D  to build message in BC,
                    BHSBSR guard 500     to build message in BC,
                    BHDBBR guard 500D   to build message in BC;
             return  BRF,
                   WHSBPN guard 400   to build message in BC,
                   WHDBPN guard 400D to build message in BC,
                   BHSBPN  guard 500   to build message in BC,
                   BHDBPN  guard 500D to build message in BC)
  update BUILDER_ROBOT
     set Status = 'in process';
  update PLASTIC_HOLDERS
    set Status = 'in process'
       where Id = :IdPlastic_Holder;
  select Max(Id) into :IdPin
          where (Code_Part = '1000215');
  Move(BR,PINsTray);
   /*  Move the Hand Robot to the PINs tray */
  waitfor(ProcessTime)
  grab(BR,PIN)
   /*  Get one big ring */
  waitfor(ProcessTime)
  Move(BR,PINHole);
   /*  Move the Hand Robot to the hole for the PIN */
```



```
put(BR,PIN,Hole)
 /*  Put the PIN into the hole */
waitfor(ProcessTime)
move(BR,Start_Position)
/*  Move the Hand Robot to the Start Position */
waitfor(ProcessTime)
case (message in BC is to build 400):
   update PLASTIC_HOLDERS
      set Status = ‘standard pin’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);

case (message in BC is to build 400D):
   update PLASTIC_HOLDERS
      set Status = ‘dilution pin’
         where (Code_Part = ‘1000190’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = ‘standard pin’
         where (Code_Part = ‘1000356’) and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = ‘dilution pin’
         where (Code_Part = ‘1000356’) and
                  (Id = :IdPlastic_Holder);
update PINS
  set Code_Plastic_Holder = :IdPlasticHolder,
      Status = ‘plastic holder’
    where Id = (:IdPin);
update BUILDER_ROBOT
   set Status = ‘flipped’;
```

```
Place WHSBPN: White Plastic Holder with Standard Base and
                        Pin inserted
create view WHSBPN as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS, PINS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
               (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
               (RFID_TAGS.Id = :IdRFId_Tag) and
               (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (BASES.Id = :IdBase) and
               (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               ((SCREWS.Id = :IdRightScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((SCREWS.Id = :IdLeftScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
               ((RINGS.Id = :IdRightRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((RINGS.Id = :IdLeftRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((RINGS.Id = :IdPinRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
               (PINS.Id = :IdPin) and
               (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (PLASTIC_HOLDERS.Status = 'standard pin');

Place WHDBPN: White Plastic Holder with Dilution Base and
                        Pin inserted
create view WHDBPN as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS, PINS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              ((RINGS.Id = :IdRightRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdLeftRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdPinRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (RINGS.Id = :IdBigRing) and
              (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PINS.Id = :IdPin) and
              (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'dilution pin');
```

Place BHSBPN: Black Plastic Holder with Standard Base and Pin inserted

```
create view BHSBPN as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS, PINS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              ((RINGS.Id = :IdRightRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdLeftRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdPinRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PINS.Id = :IdPin) and
              (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard pin');
```

Place BHDBPN: Black Plastic Holder with Dilution Base and Pin inserted

```
create view BHDBPN as
  select *
    from  PLASTIC_HOLDERS, RFID_TAGS, BASES, SCREWS, RINGS, PINS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
             (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
             (RFID_TAGS.Id = :IdRFId_Tag) and
             (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
             (BASES.Id = :IdBase) and
             (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
             ((SCREWS.Id = :IdRightScrew) and
              (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
             ((SCREWS.Id = :IdLeftScrew) and
              (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
             ((RINGS.Id = :IdRightRing) and
              (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
             ((RINGS.Id = :IdLeftRing) and
              (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
             ((RINGS.Id = :IdPinRing) and
              (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
             (RINGS.Id = :IdBigRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
             (PINS.Id = :IdPin) and
             (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
             (PLASTIC_HOLDERS.Status = 'dilution pin');
```

```
Transition IMR: Install the Metal Ring
/* The Plastic Holder is in the Plastic Base under the builder robot. The Builder Robot
   grab one metal ring and put it in the right hole. The rings are on a plastic tube maybe
    with a gap between them and the distance between one ring and another on the
    plastic tube is calculated and saved in the computer. Every time a ring is grabbed, the
    next time the builder robot go to the same place to grab the next ring after adding the
    saved distance.
*/
routine IMR (accept BRF,
                        MRR        guard count(MRR) >= 1,
                        WHSBPN guard 400     to build message in BC,
                        WHDBPN guard 400D  to build message in BC,
                        BHSBPN guard 500      to build message in BC,
                        BHDBPN guard 500D   to build message in BC;
             return  BRF,
                       WHSBMR guard 400   to build message in BC,
                       WHDBMR guard 400D to build message in BC,
                       BHSBMR  guard 500   to build message in BC,
                       BHDBMR  guard 500D to build message in BC)
   update BUILDER_ROBOT
      set Status = 'in process';
   update PLASTIC_HOLDERS
     set Status = 'in process'
       where Id = :IdPlastic_Holder;
   select Max(Id) into :IdMetal_Ring
      from METAL_RINGS
          where (Code_Part = '1000226');
   Move(BR,MetalRingsTube);
    /*  Move the Hand Robot to the Metal Rings tube */
   waitfor(ProcessTime)
   grab(BR,MetalRing)
    /*  Get one metal ring */
   waitfor(ProcessTime)
   Move(BR,PINMetalRing);
    /*  Move the Hand Robot to the hole for the metal ring above the PIN  */
   put(BR,MetalRing,PIN)
    /*  Put the metal ring on top of the PIN */
   waitfor(ProcessTime)
   move(BR,Start_Position)
   /*  Move the Hand Robot to the Start Position */
   waitfor(ProcessTime)
   case (message in BC is to build 400):
      update PLASTIC_HOLDERS
         set Status = 'standard metal ring'
            where (Code_Part = '1000190') and
                    (Id = :IdPlastic_Holder);
   case (message in BC is to build 400D):
      update PLASTIC_HOLDERS
         set Status = 'dilution metal ring'
            where (Code_Part = '1000190') and
                    (Id = :IdPlastic_Holder);
```

```
case (message in BC is to build 500):
    update PLASTIC_HOLDERS
        set Status = ‘standard metal ring’
            where (Code_Part = ‘1000356’) and
                    (Id = :IdPlastic_Holder);
case (message in BC is to build 500D):
    update PLASTIC_HOLDERS
        set Status = ‘dilution metal ring’
            where (Code_Part = ‘1000356’) and
                    (Id = :IdPlastic_Holder);
update METAL_RINGS
  set Code_Plastic_Holder = :IdPlasticHolder,
        Status = ‘plastic holder’
     where Id = (:IdMetalRing);
update BUILDER_ROBOT
    set Status = ‘flipped’;
```

Place WHSBMR: White Plastic Holder with Standard Base and Metal Ring inserted

```
create view WHSBMR as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              ((RINGS.Id = :IdRightRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdLeftRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdPinRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PINS.Id = :IdPin) and
              (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (METAL_RINGS.Id = :IdMetal_Ring) and
              (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard metal ring');
```

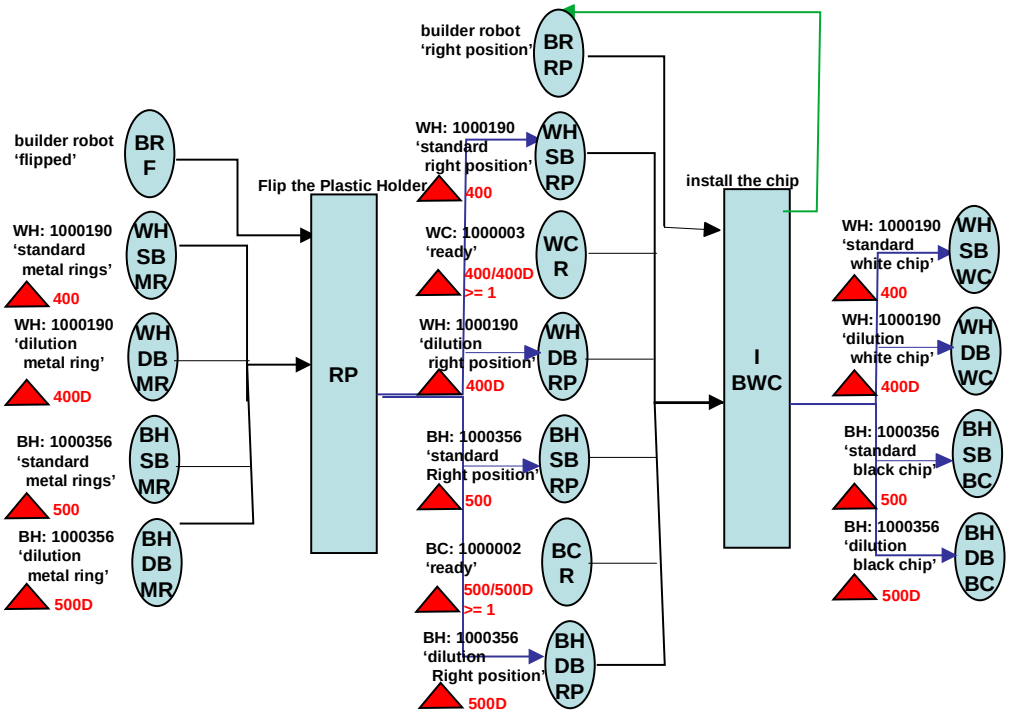
Place WHDBMR: White Plastic Holder with Dilution Base and Metal Ring inserted

```
create view WHDBMR as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution metal ring');
```

```
Place BHSBMR: Black Plastic Holder with Standard Base and
                    Metal Ring inserted
create view BHSBMR as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard metal ring');
```

Place BHDBMR: Black Plastic Holder with Dilution Base and
Metal Ring inserted

```
create view BHDBMR as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution metal ring');
```



Transition RP: Flip the plastic Holder

/* The hand robot grab the Plastic Holder and turn it 180 degrees */

```
routine IMR (accept BRF,
                    WHSBMR guard 400     to build message in BC,
                    WHDBMR guard 400D  to build message in BC,
                    BHSBMR guard 500      to build message in BC,
                    BHDBMR guard 500D   to build message in BC;
             return  BRRP,
                   WHSBRP guard 400   to build message in BC,
                   WHDBRP guard 400D to build message in BC,
                   BHSBRP  guard 500   to build message in BC,
                   BHDBRP  guard 500D to build message in BC)
  update BUILDER_ROBOT
     set Status = 'in process';
  update PLASTIC_HOLDERS
    set Status = 'in process'
       where Id = :IdPlastic_Holder;
  move(BR,PlasticHolderBase)
     /* The Builder Robot move to the Plastic Holder Base under it */
  wait_for(delaytime)
  grab(BR,PlasticHolder)
     /* The builder Robot grab the Plastic Holder out from the Plastic Holder Base */
  wait_for(delaytime)
  flip(BR,PlasticHolder)
     /* The hand Robot turn 180 degrees */
  wait_for(delaytime)
  release(BR,PlasticHolder)
     /* The builder Robot release the Plastic Holder into the Plastic Holder Base */
  wait_for(delaytime)
```

```
update PLASTIC_HOLDERS
    set Status = ‘right position’
      where Id = :IdPlastic_Holder;
update BUILDER_ROBOT
   set Status = ‘right position’;
```

Place WHSBRP: White Plastic Holder with Standard Base in Right Position

```
create view WHSBRP as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Rings) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard right position');
```

Place WHDBRP: White Plastic Holder with Dilution Base in Right Position

```
create view WHDBRP as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution right position');
```

Place BHSBRP: Black Plastic Holder with Standard Base in Right Position

```
create view BHSBRP as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFIdT_ag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              ((RINGS.Id = :IdRightRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdLeftRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdPinRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PINS.Id = :IdPin) and
              (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (METAL_RINGS.Id = :IdMetal_Ring) and
              (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard right position');
```

```
Place BHDBRP: Black Plastic Holder with Dilution Base in
                        Right Position
create view BHDBRP as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
               (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
               (RFID_TAGS.Id = :IdRFId_Tag) and
               (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (BASES.Id = :IdBase) and
               (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               ((SCREWS.Id = :IdRightScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((SCREWS.Id = :IdLeftScrew) and
                (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
               ((RINGS.Id = :IdRightRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((RINGS.Id = :IdLeftRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
               ((RINGS.Id = :IdPinRing) and
                (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
               (RINGS.Id = :IdBigRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (PINS.Id = :IdPin) and
               (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (METAL_RINGS.Id = :IdMetal_Ring) and
               (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
               (PLASTIC_HOLDERS.Status = 'dilution right position');

Place BRRP: Builder Robot Flipped
create view BRRP as
  select *
    from  ROBOTS
       where (ROBOTS.Kind = 'builder') and
                (ROBOTS.Status = 'right position');
```

Transition IBWC: Install the Chip

/* The Plastic Holder is in the Plastic Base under the builder robot. ***The computer give a command to the Chips Stack Device to move out one Chip. This is done by pushing the Chip plastic tube holder from the back where a steel tube having a form of reversed "s" penetrate into the open hole of the plastic tube and push the chip. The Robot Builder grab the Chip and put it in its place on the Plastic Holder.***
The warehouse u*se an image processing to recognize the square of digits in the chip, or a sensor to recognize the edge of the chip with a sharp side and a square side; in both cases, the chip will be positioned in the right place in the plastic holder tube* with the digits faced up. If there is no image processor, a needle scan the edge of the chip, if the sensor of the movement found a square shape followed by a sharp point shape, it mean the chip is face up, if not will be turned up-down. The warehouse robot has to recognize the chips number (1000002 /1000003) from the scanner of the box and put them in the right tube.
***/**

```
routine IBWC (accept BRRP,
                          WHSBRP  guard 400          to build message in BC,
                           WCR       guard 400 / 400D to build message in BC,
                                          count(WCR) >= 1,
                           WHDBRP guard 400D         to build message in BC,
                           BHSBRP  guard 500          to build message in BC,
                           BCR       guard 500 / 500D  to build message in BC,
                                          count(BCR) >= 1,
                           BHDBRP  guard 500D          to build message in BC;
                 return  BRRP,
                          WHSBWC guard 400   to build message in BC,
                          WHDBWC guard 400D to build message in BC,
                          BHSBBC   guard 500   to build message in BC,
                          BHDBBC   guard 500D to build message in BC)
   update BUILDER_ROBOT
      set Status = 'in process';
   update PLASTIC_HOLDERS
     set Status = 'in process'
       where Id = :IdPlastic_Holder;

   case (message in BC is to build 400):
      select Max(Id) into :IdChip
         from CHIPS
           where (Code_Part = '1000003');
      move(BR,WhiteChipsTube)
      /*  Move the Hand Robot to the 1000003 Chips TubeHolder */
      waitfor(ProcessTime)
      push(SteelTube,Chip,halfway)
      /*  The computer give command to the Steel Tube to move one 1000003 Chip
         forward halfway */
      waitfor(ProcessTime)
     grab(BR,Chip)
      /*  The computer grab the 1000003 Chip */
      waitfor(ProcessTime)
      move(BR,PlaticHolderBase)
      /*  Move the Hand Robot to the Plastic Holder Base under it */
      waitfor(ProcessTime)
```

```
    put(BR,Chip,PlasticHolder)
    /*  The Hand Robot put the 1000003 Chip in its place */
    waitfor(ProcessTime)
    update PLASTIC_HOLDERS
       set Status = 'standard white chip'
          where (Code_Part = '1000190') and
                (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
    select Max(Id) into :IdChip
       from CHIPS
         where (Code_Part = '1000003');
    move(BR,WhiteChipsTube)
    /*  Move the Hand Robot to the 1000003 Chips TubeHolder */
    waitfor(ProcessTime)
    push(SteelTube,Chip,halfway)
    /*  The computer give command to the Steel Tube to move one 1000003 Chip
        forward halfway */
    waitfor(ProcessTime)
   grab(BR,Chip)
    /*  The computer grab the 1000003 Chip */
    waitfor(ProcessTime)
    move(BR,PlaticHolderBase)
    /*  Move the Hand Robot to the Plastic Holder Base under it */
    waitfor(ProcessTime)
    put(BR,Chip,PlasticHolder)
    /*  The Hand Robot put the 1000003 Chip in its place */
    waitfor(ProcessTime)
    update PLASTIC_HOLDERS
       set Status = 'dilution white chip'
          where (Code_Part = '1000190') and
                (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
    select Max(Id) into :IdChip
       from CHIPS
         where (Code_Part = '1000002');
    move(BR,WhiteChipsTube)
    /*  Move the Hand Robot to the 1000002 Chips TubeHolder */
    waitfor(ProcessTime)
    push(SteelTube,Chip,halfway)
    /*  The computer give command to the Steel Tube to move one 1000002 Chip
       forward halfway */
    waitfor(ProcessTime)
   grab(BR,Chip)
    /*  The computer grab the 1000002 Chip */
    waitfor(ProcessTime)
    move(BR,PlaticHolderBase)
    /*  Move the Hand Robot to the Plastic Holder Base under it */
    waitfor(ProcessTime)
    put(BR,Chip,PlasticHolder)
    /*  The Hand Robot put the 1000002 Chip in its place */
    waitfor(ProcessTime)
```

```
    update PLASTIC_HOLDERS
       set Status = ‘standard black chip’
          where (Code_Part = ‘1000356’) and
                   (Id = :IdPlastic_Holder);

case (message in BC is to build 500D):
    select Max(Id) into :IdChip
       from CHIPS
         where (Code_Part = ‘1000002’);
    move(BR,WhiteChipsTube)
    /*  Move the Hand Robot to the 1000002 Chips TubeHolder */
    waitfor(ProcessTime)
    push(SteelTube,Chip,halfway)
    /*  The computer give command to the Steel Tube to move one 1000002 Chip
 forward halfway */
    waitfor(ProcessTime)
   grab(BR,Chip)
    /*  The computer grab the 1000002 Chip */
    waitfor(ProcessTime)
    move(BR,PlaticHolderBase)
    /*  Move the Hand Robot to the Plastic Holder Base under it */
    waitfor(ProcessTime)
    put(BR,Chip,PlasticHolder)
    /*  The Hand Robot put the 1000002 Chip in its place */
    waitfor(ProcessTime)
    update PLASTIC_HOLDERS
       set Status = ‘dilution black chip’
          where (Code_Part = ‘1000356’) and
                   (Id = :IdPlastic_Holder);
move(BR,Start_Position)
/*  Move the Hand Robot to the Start Position */
waitfor(ProcessTime)
update CHIPS
  set Code_Plastic_Holder = :IdPlastic_Holder,
       Status = ‘plastic holder’
     where Id = (:IdChip);
update BUILDER_ROBOT
    set Status = ‘flipped’;
```

Place WHSBWC:
White Plastic Holder with Standard Base and
Chip inserted

```
create view WHSBWC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
              (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
              (RFID_TAGS.Id = :IdRFId_Tag) and
              (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (BASES.Id = :IdBase) and
              (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              ((SCREWS.Id = :IdRightScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((SCREWS.Id = :IdLeftScrew) and
               (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              ((RINGS.Id = :IdRightRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdLeftRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
              ((RINGS.Id = :IdPinRing) and
               (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
              (PINS.Id = :IdPin) and
              (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (METAL_RINGS.Id = :IdMetal_Ring) and
              (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (CHIPS.Id = :IdChip) and
              (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
              (PLASTIC_HOLDERS.Status = 'standard white chip');
```

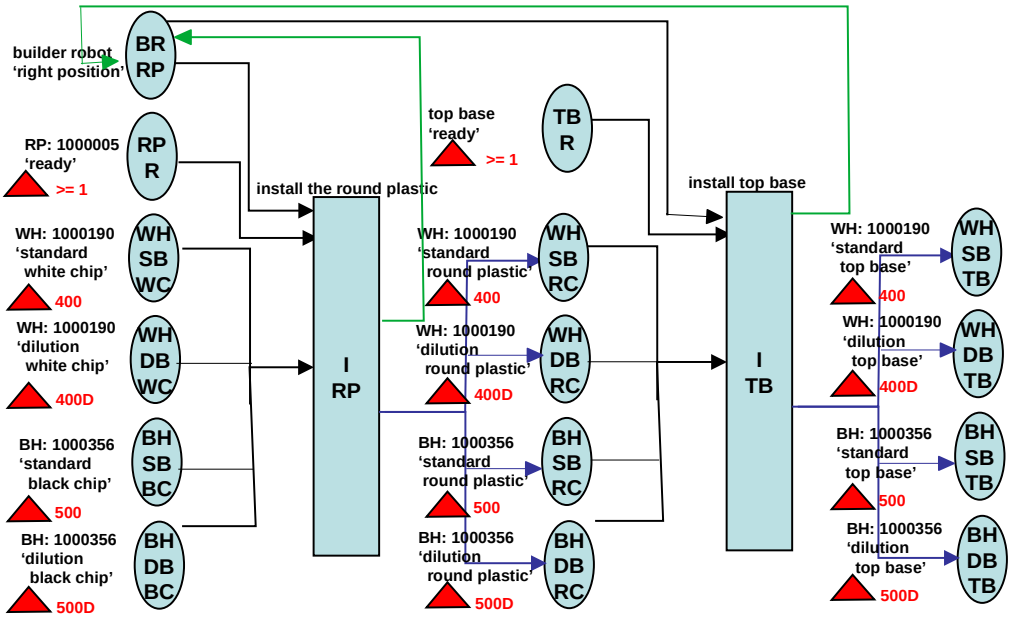
Place WHDBWC: White Plastic Holder with Dilution Base and Chip Ring inserted

```
create view WHDBWC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution white chip');
```

```
Place BHSBBC: Black Plastic Holder with Standard Base and
                    Chip Ring inserted
create view BHSBBC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard black chip');
```

Place BHDBBC: Black Plastic Holder with Dilution Base and
Chip Ring inserted

```
create view BHDBBC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution black chip');
```



Transition IRP: Install the Round Plastic Protector

/* The Plastic Holder is in the Plastic Base under the builder robot. *The computer give a command to the Round Plastic Protector Tape Device to move one Round Plastic Protector halfway, pass in a lame to peel it. The Device is attached to the wall and move the tape one Round Plastic Protector at the time. The Builder Robot grab it and the device move it totally out. The builder Robot put it on top of the chip.*

***/**

```
routine IRP (accept BRRP,
                    RPR          guard count(RPR) >= 1,
                    WHSBWC  guard 400   to build message in BC,
                    WHDBWC guard 400D to build message in BC,
                    BHSBBC  guard 500    to build message in BC,
                    BHDBBC  guard 500D to build message in BC;
             return  BRRP,
                   WHSBRC guard 400     to build message in BC,
                   WHDBRC guard 400D  to build message in BC,
                   BHSBRC   guard 500    to build message in BC,
                   BHDBRC   guard 500D to build message in BC)
  update BUILDER_ROBOT
     set Status = 'in process';
  update PLASTIC_HOLDERS
    set Status = 'in process'
       where Id = :IdPlastic_Holder;
  push(Device,Round_Plastic,halfway)
  /*  The device push the Round Plastic Protector while peeling it halfway */
  waitfor(ProcessTime)
  grab(BR,Round_Plastic)
  /*  The Builder Robot hold the Round Plastic Protector */
  waitfor(ProcessTime)
  push(Device,Round_Plastic,halfway)
  /*  The device push the Round Plastic Protector while peeling it totally while hold by
      the Builder Robot */
  waitfor(ProcessTime)
```

```
move(BR,PlaticHolderBase)
/*  Move the Hand Robot to the Plastic Holder Base under it */
waitfor(ProcessTime)
put(BR,Round_Plastic,Chip)
/*  The Hand Robot put the Round Plastic Protector on top of the chip */
waitfor(ProcessTime)
move(BR,Start_Position)
/*  Move the Hand Robot to the Start Position */
waitfor(ProcessTime)
select Max(Id) into :IdRound_Plastic
   from ROUND_PASTIC
     where (Code_Part = '1000005');
case (message in BC is to build 400):
   update PLASTIC_HOLDERS
      set Status = 'standard round plastic'
         where (Code_Part = '1000190') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
   update PLASTIC_HOLDERS
      set Status = 'dilution round plastic'
         where (Code_Part = '1000190') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = 'standard round plastic'
         where (Code_Part = '1000356') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500D):
   update PLASTIC_HOLDERS
      set Status = 'dilution round plastic'
         where (Code_Part = '1000356') and
                  (Id = :IdPlastic_Holder);
update ROUND_PLASTIC
  set Code_Plastic_Holder = :IdPlastic_Holder,
      Status = 'plastic holder'
    where Id = (:IdRound_Plastic);
update BUILDER_ROBOT
   set Status = 'flipped';
```

Place WHSBRC: White Plastic Holder with Standard Base and
Round Plastic Protector inserted

```
create view WHSBRC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard round plastic');
```

```
Place WHDBRC: White Plastic Holder with Dilution Base and
                        Round Plastic Protector inserted
create view WHDBRC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution round plastic');
```

```
Place BHSBRC: Black Plastic Holder with Standard Base and
                        Round Plastic Protector inserted
create view BHSBRC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard black round plastic');
```

Place BHDBRC: Black Plastic Holder with Dilution Base and
Round Plastic Protector inserted

```
create view BHDBRC as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution black round plastic');
```

Transition ITB: Install Top Base

/* Use the hole for the number as a reference. The camera can look over this hole and turn accordingly the base till it see number 2 or number 3. Another way is when the warehouse bring the Top Bases is to insert them in a cylinder with a large plastic pin for the hole for number as a reference. So, when they take it away to use it, it come on a unique way, so it will be filled in its place. Anyways, there is many ways to achieve it The builder Robot take one Top Base and put it on the Plastic Holder */

```
routine ITB (accept BRRP,
                          TBR          guard count(TBR) >= 1,
                          WHSBRC  guard 400   to build message in BC,
                          WHDBRC guard 400D to build message in BC,
                          BHSBRC  guard 500    to build message in BC,
                          BHDBRC  guard 500D to build message in BC;
                 return  BRRP,
                         WHSBTB guard 400     to build message in BC,
                         WHDBTB guard 400D  to build message in BC,
                         BHSBTB   guard 500    to build message in BC,
                         BHDBTB   guard 500D to build message in BC)
   update BUILDER_ROBOT
      set Status = 'in process';
   update PLASTIC_HOLDERS
     set Status = 'in process'
       where Id = :IdPlastic_Holder;

   push(SteelTube,TopBase,halfway)
   /*  The Steel Tube push the Top Base halfway */
   waitfor(ProcessTime)
   grab(BR,Round_Plastic)
   /*  The Builder Robot grab the Top Base and bring it out */
   waitfor(ProcessTime)
   move(BR,PlaticHolderBase)
   /*  Move the Hand Robot to the Plastic Holder Base under it */
   waitfor(ProcessTime)
   put(BR,TopBase,PlasticHolder)
   /*  The Hand Robot put the Top Base on top of the Base */
   waitfor(ProcessTime)
   move(BR,Start_Position)
   /*  Move the Hand Robot to the Start Position */
   waitfor(ProcessTime)
   select Max(Id) into :IdTopBase
      from Bases
       where (Code_Part = 'codetopbase');
   case (message in BC is to build 400):
      update PLASTIC_HOLDERS
         set Status = 'standard top base'
            where (Code_Part = '1000190') and
                    (Id = :IdPlastic_Holder);
   case (message in BC is to build 400D):
      update PLASTIC_HOLDERS
         set Status = 'dilution top base'
            where (Code_Part = '1000190') and
                    (Id = :IdPlastic_Holder);
```

```
case (message in BC is to build 500):
    update PLASTIC_HOLDERS
        set Status = ‘standard top base’
            where (Code_Part = ‘1000356’) and
                    (Id = :IdPlastic_Holder);
case (message in BC is to build 500D):
    update PLASTIC_HOLDERS
        set Status = ‘dilution top base’
            where (Code_Part = ‘1000356’) and
                    (Id = :IdPlastic_Holder);    update Bases
  set Code_Plastic_Holder = :IdPlastic_Holder,
        Status = ‘plastic holder’
     where Id = (:IdTopBase);
update BUILDER_ROBOT
    set Status = ‘flipped’;
```

Place WHSBTB: White Plastic Holder with Standard Base and Top Base inserted

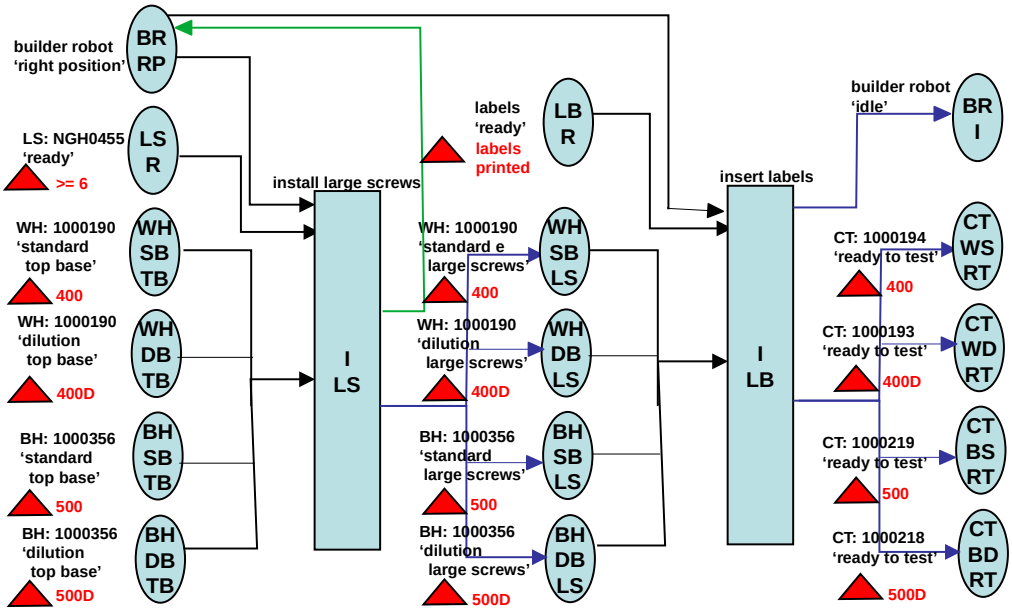
```
create view WHSBTB as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard top base');
```

```
Place WHDBTB: White Plastic Holder with Dilution Base and
                        Top Base inserted
create view WHDBTB as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution top base');
```

```
Place BHSBTB: Black Plastic Holder with Standard Base and
                        Top Base inserted
create view BHSBTB as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlasticHolder) and
            (RFID_TAGS.Id = :IdRFIdTag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'standard top base');
```

Place BHDBTB: Black Plastic Holder with Dilution Base and
Top Base inserted

```
create view BHDBTB as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PLASTIC_HOLDERS.Status = 'dilution top base');
```



Transition ILS: Install Large Screws

```
/* The Plastic Holder is in the Plastic Base under the builder robot. The Builder Robot
   move to the Large Screws Container, take one screw at a time and put it in the right
   hole. There is six Screw Drivers on a circular track positioned above the holes for the
   large screws. The track can move via pipes down and up to the screws to screws
   them.
*/
routine ILS (accept BRRP,
                        LSR        guard count(LSR) >= 6,
                        WHSBTB guard 400   to build message in BC,
                        WHDBTB guard 400D to build message in BC,
                        BHSBTB  guard 500    to build message in BC,
                        BHDBTB  guard 500D to build message in BC;
             return  BRRP,
                       WHSBLS  guard 400     to build message in BC,
                       WHDBLS  guard 400D  to build message in BC,
                       BHSBLS   guard 500    to build message in BC,
                       BHDBLS   guard 500D to build message in BC)

  /* The builder robot move the Plastic Holder with the top base in the right position
     under the screw driver for each screw to be inserted. A command from the
     computer make the screw driver go down and insert the screw. */
   select Id into :IdFirstScrew, :IdSecondScrew, :IdThirdScrew, :IdFourthScrew,
                        :IdFifthScrew, :IdSixthScrew
          from SCREWS
            order by id desc
            fetch first 6 rows only
             where (Code_Part = 'NGH0455');
   update BUILDER_ROBOT
      set Status = 'in process';
   update PLASTIC_HOLDERS
     set Status = 'in process'
       where Id = :IdPlastic_Holder;
```

```
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,FirstHoleScrew);
 /*  Move the Hand Robot to the First hole for the large screw in the Plastic Holder
    Base */
put(BR,LargeScrew,FirstHole)
 /*  Put the large screw into the first hole */
waitfor(ProcessTime)
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,FourthHoleScrew);
 /*  Move the Hand Robot to the fourth hole for the large screw in the Plastic Holder
      Base */
put(BR,LargeScrew,FourthHole)
 /*  Put the large screw into the fourth hole */
waitfor(ProcessTime)
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,SecondHoleScrew);
 /*  Move the Hand Robot to the second hole for the large screw in the Plastic Holder
    Base */
put(BR,LargeScrew,second Hole)
 /*  Put the large screw into the second hole */
waitfor(ProcessTime)
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,FifthHoleScrew);
 /*  Move the Hand Robot to the fifth hole for the large screw in the Plastic Holder
    Base */
put(BR,LargeScrew,FifthHole)
 /*  Put the large screw into the fifth hole */
waitfor(ProcessTime)
```

```
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,ThirdHoleScrew);
 /*  Move the Hand Robot to the third hole for the large screw in the Plastic Holder
    Base */
put(BR,LargeScrew,ThirdHole)
 /*  Put the large screw into the third hole */
waitfor(ProcessTime)
Move(BR,ScrewsContainer);
 /*  Move the Hand Robot to the Large Screws container */
waitfor(ProcessTime)
grab(BR,LargeScrew)
 /*  Get one large screw */
waitfor(ProcessTime)
Move(BR,SixthHoleScrew);
 /*  Move the Hand Robot to the sixth hole for the large screw in the Plastic Holder
    Base */
put(BR,LargeScrew,SixthHole)
 /*  Put the large screw into the sixth hole */
waitfor(ProcessTime)
case (message in BC is to build 400):
   update PLASTIC_HOLDERS
      set Status = 'standard large screw'
         where (Code_Part = '1000190') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 400D):
   update PLASTIC_HOLDERS
      set Status = 'dilution large screw'
         where (Code_Part = '1000190') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500):
   update PLASTIC_HOLDERS
      set Status = 'standard large screw'
         where (Code_Part = '1000356') and
                  (Id = :IdPlastic_Holder);
case (message in BC is to build 500D):
   update PLASTIC_HOLDERS
      set Status = 'dilution large screw'
         where (Code_Part = '1000356') and
                  (Id = :IdPlastic_Holder);
update Screws
  set Code_Plastic_Holder = :IdPlastic_Holder,
      Status = 'plastic holder'
    where (Id = :IdFirstScrew) or (Id = :IdSecondScrew) or (Id = :IdThirdScrew) or
          (Id = :IdFourthScrew) or (Id = :IdFifthScrew) or (Id = :IdSixthScrew);
update BUILDER_ROBOT
   set Status = 'right position';
```

```
Place WHSBLS: White Plastic Holder with Standard Base and
                    Large Screws inserted
create view WHSBLS as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdFirstScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSecondScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdthirdScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdFourthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdFifthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSixthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'standard large screws');
```

```
Place WHDBLS: White Plastic Holder with Dilution Base and
                        Large screws inserted
create view WHDBLS as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000190') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdFirstScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSecondScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdthirdScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdFourthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdFifthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSixthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'dilution large screws');
```

Place BHSBLS: Black Plastic Holder with Standard Base and Large Screws inserted

```
create view BHSBLS as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRoundPlastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdFirstScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSecondScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdthirdScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdFourthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdFifthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSixthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'standard large screws');
```

Place BHDBLS: Black Plastic Holder with Dilution Base and Large Screws inserted

```
create view BHDBLS as
  select *
    from  PLASTIC_HOLDERS, BASES, SCREWS, RINGS, PINS, METAL_RINGS, CHIPS,
          ROUND_PLASTIC
      where (PLASTIC_HOLDERS.Code_Part = '1000356') and
            (PLASTIC_HOLDERS.Id = :IdPlastic_Holder) and
            (RFID_TAGS.Id = :IdRFId_Tag) and
            (RFId.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdRightScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdLeftScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((RINGS.Id = :IdRightRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdLeftRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((RINGS.Id = :IdPinRing) and
             (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (RINGS.Id = :IdBigRing) and
            (RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (PINS.Id = :IdPin) and
            (PINS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (METAL_RINGS.Id = :IdMetal_Ring) and
            (METAL_RINGS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (CHIPS.Id = :IdChip) and
            (CHIPS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (ROUND_PLASTIC.Id = :IdRound_Plastic) and
            (ROUND_PLASTIC.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            (BASES.Id = :IdTopBase) and
            (BASES.Code_Plastic_Holder = PLASTIC_HOLDERS.Id) and
            ((SCREWS.Id = :IdFirstScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSecondScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdthirdScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdFourthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            ((SCREWS.Id = :IdFifthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) or
            ((SCREWS.Id = :IdSixthScrew) and
             (SCREWS.Code_Plastic_Holder = PLASTIC_HOLDERS.Id)) and
            (PLASTIC_HOLDERS.Status = 'dilution large screws');
```

```
Transition ILB: Install Lables
/* There is many ways to make the printing of the labels an automatic process: bring
   the warehouse robot to the printer, change the rolls for labels and ink. Make the
   computer simulate the login and access to printer software and writing the lot code
   and printing automatically by sending the right APIs to the mouse and
   keyboard or request a printer from the manufacturer with APIs to call for printing.
   One time printing is ready, the robot bring them to the table and roll them over a
   special device on the wall. The device roll one label at a time and a clamp take it
   and insert it in the right position on the cartridge
*/
routine ILB (accept BRRP,
                        LBR        guard labels printed,
                        WHSBLS guard 400   to build message in BC,
                        WHDBLS guard 400D to build message in BC,
                        BHSBLS  guard 500    to build message in BC,
                        BHDBLS  guard 500D to build message in BC;
                return  BRRP,
                        WHSBRT  guard 400     to build message in BC,
                        WHDBRT  guard 400D  to build message in BC,
                        BHSBRT   guard 500    to build message in BC,
                        BHDBRT   guard 500D to build message in BC)

   update BUILDER_ROBOT
      set Status = 'in process';
   /* Use manually process to print the labels and insert them */
   delete from PLASTIC_HOLDERS
     where (Id = :IdPlastic_Holder;
   delete from RFID_TAGS
     where (Id = :IdRFId_Tag);
   delete from BASES
        where (Id = :IdBase);
   delete from SCREWS
     where (Id = :IdRightScrew');
   delete from SCREWS
     where (Id = :IdLeftScrew);
   delete from RINGS
      where (Id = :IdRightRing);
   delete from RINGS
      where (Id = :IsLeftRing)
   delete from RINGS
      where (Id = :IdPinRing)
   delete from RINGS
      where (Id = :IdBigRing)
  delete from PINS
     where (Id = :IdPin)
  delete from METAL_RINGS
    where (Id = :IdMetal_Ring)
  delete from CHIPS
    where (Id = :IdChip)
  delete from ROUND_PLASTIC
    where (Id = :IdRound_Plastic)
  delete from BASES
    where (Id = :IdTopBase)
```

```
delete from SCREWS
  where (Id = :IdFirstScrew');
delete from SCREWS
  where (Id = :IdSecondScrew');
delete from SCREWS
  where (Id = :IdThirdScrew');
delete from SCREWS
  where (Id = :IdFourthScrew');
delete from SCREWS
  where (Id = :IdFifthScrew');
delete from SCREWS
  where (Id = :IdSixthScrew');

case (message in BC is to build 400):
   insert into CARTRIDGES (Code,       Status          ) values
                          ('1000194', 'ready to test');
select Max(Id) into :IdCartridge
   from CARTRIDGES
     where (Code = '1000194');
case (message in BC is to build 400D):
   insert into CARTRIDGES (Code,       Status          ) values
                          ('1000193', 'ready to test');
select Max(Id) into :IdCartridge
   from CARTRIDGES
     where (Code = '1000193');
case (message in BC is to build 500):
   insert into CARTRIDGES (Code,       Status          ) values
                          ('1000219', 'ready to test');
select Max(Id) into :IdCartridge
   from CARTRIDGES
     where (Code = '1000219');
case (message in BC is to build 500D):
   insert into CARTRIDGES (Code,       Status          ) values
                          ('1000218', 'ready to test');
select Max(Id) into :IdCartridge
   from CARTRIDGES
     where (Code = '1000218');
```

Place CTWSRT: 400 ready to test

```
create view CTWSRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
              (Id = :IdCartridge) and
              (Status = ‘ready to test’);
```

Place CTWDRT: 400D ready to test

```
create view CTWDRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
              (Id = :IdCartridge) and
              (Status = ‘ready to test’);
```

Place CTBSRT: 500 ready to test

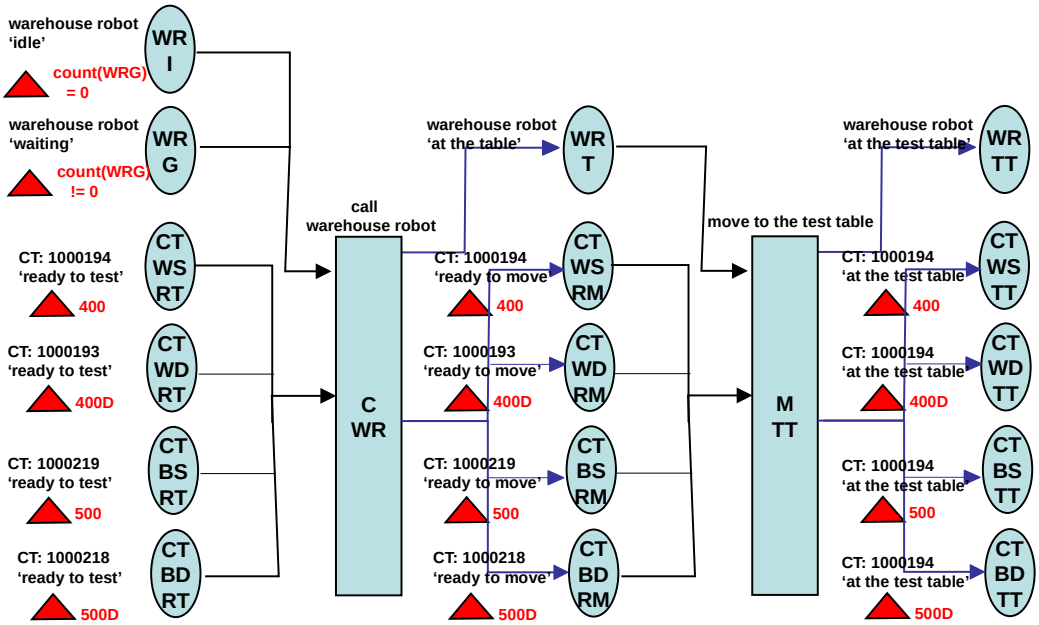
```
create view CTBSRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
              (Id = :IdCartridge) and
              (Status = ‘ready to test’);
```

Place CTBDRT: 500D ready to test

```
create view CTBDRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
              (Id = :IdCartridge) and
              (Status = ‘ready to test’);
```

Testing & Packaging

Alan El-Sabbagh WHATSAPP: +971551574255



Place WRG: Warehouse Robot with status 'waiting'

/* The warehouse robot will be in this situation when the cartridge is on the tester to be tested */

```
create view WRB as
 select *
    from  WAREHOUSE_ROBOTS
      where Status = 'waiting';
```

Transition CWR: Bring the warehouse robot to the builder table to move the cartridges to the final phase

```
routine CWR (accept WRI       guard count(WRG) =  0,
                    WRG       guard count(WRG) != 0,
                    CTWSRT guard 400   to build message in BC,
                    CTWDRT guard 400D to build message in BC,
                    CTBSRT  guard 500    to build message in BC,
                    CTBDRT  guard 500D to build message in BC;
             return    WRT,
                    CTWSRM guard 400   to build message in BC,
                    CTWDRM guard 400D to build message in BC,
                    CTBSRM  guard 500    to build message in BC,
                    CTBDRM  guard 500D to build message in BC)
```

```
  /* The warehouse robot should move to the builder table to collect the cartridge built
     for further process. */
   move_robot(WR,table_location)
   waitfor(ProcessTime)

   select COUNT(*) as count_rows
      From WRG;
```

```
if (count_rows == 0)
   update WAREHOUSE_ROBOTS
      set status = ‘at the table’
         where Status = ‘idle’
  else
   update WAREHOUSE_ROBOTS
      set status = ‘at the table’
         where Status = ‘waiting’;

case (message in BC is to build 400):
   select MAX(Id) into :IdCartridge
      from CARTRIDGES
          where (Code = ‘1000194’) and
                    (Status = ‘ready to test’);
   update CARTRIDGES
      set Status = ‘ready to move’
         where (Id = :IdCartridge);
case (message in BC is to build 400D):
   select MAX(Id) into :IdCartridge
      from CARTRIDGES
          where (Code = ‘1000193’) and
                    (Status = ‘ready to test’);
case (message in BC is to build 500):
   select MAX(Id) into :IdCartridge
      from CARTRIDGES
          where (Code = ‘1000219’) and
                    (Status = ‘ready to test’);
   update CARTRIDGES
      set Status = ‘ready to move’
         where (Id = :IdCartridge);
case (message in BC is to build 500D):
   select MAX(Id) into :IdCartridge
      from CARTRIDGES
          where (Code = ‘1000218’) and
                    (Status = ‘ready to test’);
   update CARTRIDGES
      set Status = ‘at the table’
         where (Id = :IdCartridge);
```

Place CTWSRT: 400 ready to test

```
create view CTWSRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
                (Id = :IdCartridge) and
                (Status = ‘ready to move’);
```

Place CTWDRT: 400D ready to test

```
create view CTWDBRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
                (Id = :IdCartridge) and
                (Status = ‘ready to move’);
```

Place CTBSRT: 500 ready to test

```
create view CTBSRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
                (Id = :IdCartridge) and
                (Status = ‘ready to move’);
```

Place CTBDRT: 500D ready to test

```
create view CTBDRT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
                (Id = :IdCartridge) and
                (Status = ‘ready to move’);
```

```
Transition MTT: Bring the warehouse robot  to the test table
routine MTT (accept WRT,
                        CTWSRM guard 400   to build message in BC,
                        CTWDRM guard 400D to build message in BC,
                        CTBSRM  guard 500    to build message in BC,
                        CTBDRM  guard 500D to build message in BC;
             return    WRT,
                        CTWSTT guard 400   to build message in BC,
                        CTWDTT guard 400D to build message in BC,
                        CTBSTT  guard 500    to build message in BC,
                        CTBDTT  guard 500D to build message in BC)

  /* The warehouse robot take the cartridge built to the test table. */
   grabcartridge(WR,cartridge)
   waitfor(ProcessTime)
   move_robot(WR,test_table)
   waitfor(ProcessTime)
   puttable(WR,cartridge)
   waitfor(ProcessTime)

   update WAREHOUSE_ROBOTS
      set status = 'at the test table'
         where Status = 'at the table';

   case (message in BC is to build 400):
      update CARTRIDGES
         set Status = 'at the test table'
            where (Code = '1000194') and
                     (Id = :IdCartridge) and
                     (Status = 'ready to move');
   case (message in BC is to build 400D):
      update CARTRIDGES
         set Status = 'at the test table'
            where (Code = '1000193') and
                     (Id = :IdCartridge) and
                     (Status = 'ready to move');
   case (message in BC is to build 500):
      update CARTRIDGES
         set Status = 'at the test table'
            where (Code = '1000219') and
                     (Id = :IdCartridge) and
                     (Status = 'ready to move');
   case (message in BC is to build 500D):
      update CARTRIDGES
         set Status = 'at the test table'
            where (Code = '1000218') and
                     (Id = :IdCartridge) and
                     (Status = 'ready to move');
```

Place CTWSTT: 400 at the test table

```
create view CTWSTT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
              (Id = :IdCartridge) and
              (Status = ‘at the test table’);
```

Place CTWDTT: 400D at the test table

```
create view CTWDBTT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
              (Id = :IdCartridge) and
              (Status = ‘at the test table’);
```

Place CTBSTT: 500 at the test table

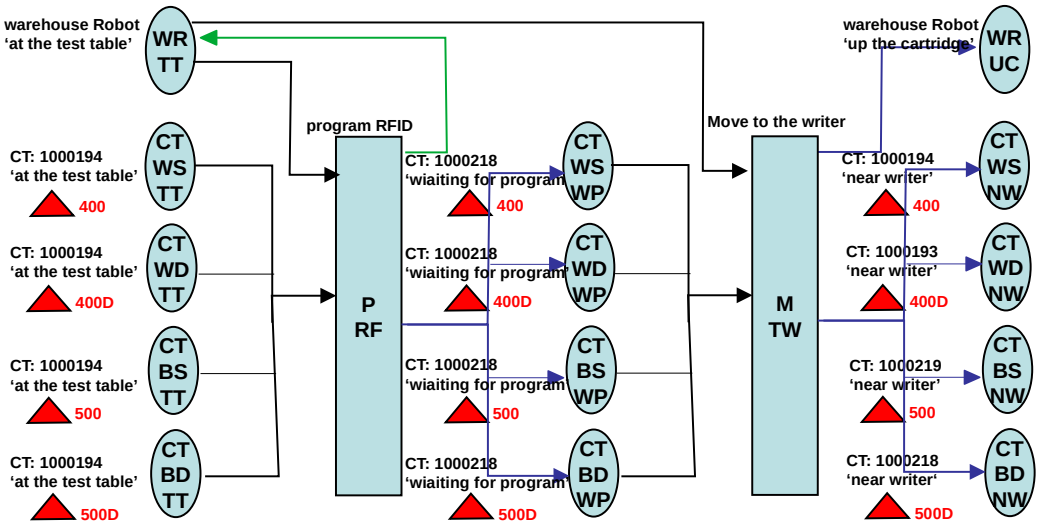
```
create view CTBSTT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
              (Id = :IdCartridge) and
              (Status = ‘at the test table’);
```

Place CTBDTT: 500D at the test table

```
create view CTBDTT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
              (Id = :IdCartridge) and
              (Status = ‘at the test table’);
```

Place WRTT: Warehouse Robot at the test table

```
create view WRTT as
    select *
       from WAREHOUSE_ROBOTS
         where Status = ‘at the test table’;
```



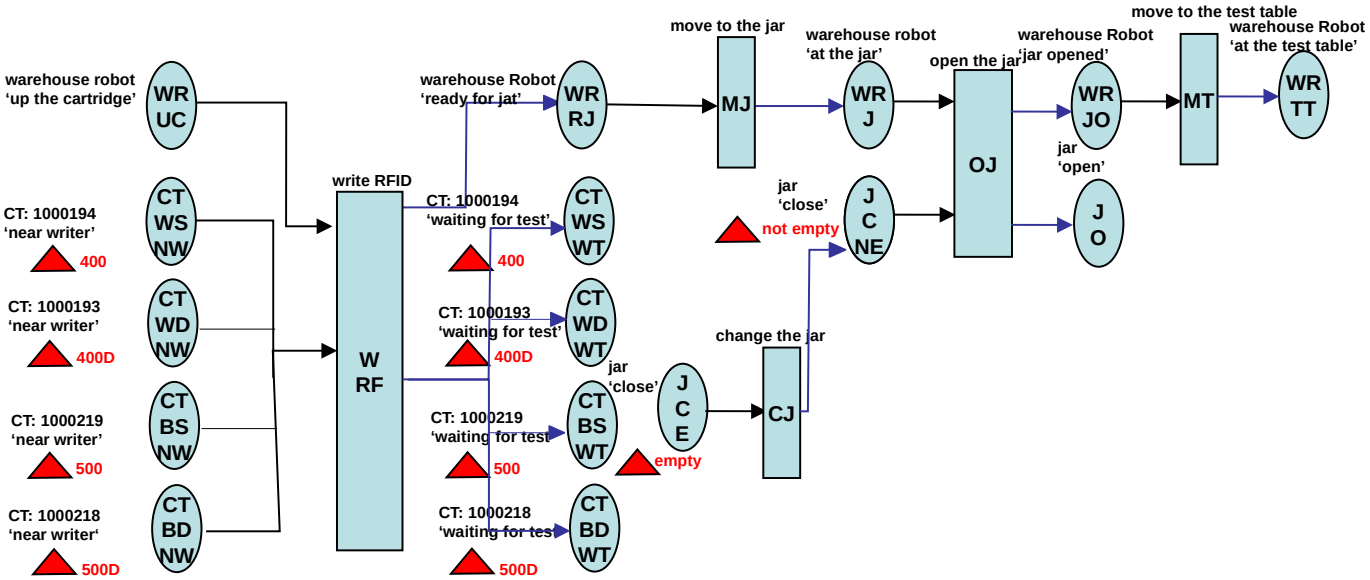
Transition PRF: Prepare the program to write the RFID tag

```
routine PRF (accept WRTT,
                    CTWSTT guard 400   to build message in BC,
                    CTWDTT guard 400D to build message in BC,
                    CTBSTT  guard 500    to build message in BC,
                    CTBDTT  guard 500D to build message in BC;
             return    WRTT,
                    CTWSWP guard 400   to build message in BC,
                    CTWDWP guard 400D to build message in BC,
                    CTBSWP  guard 500    to build message in BC,
                    CTBDWP  guard 500D to build message in BC)

 /* Use of an API of the writer program to input the data of the cartridge or use APIs for
       keyboard and mouse to enter the data. */
 input_data(cartridge_info)
 waitfor(ProcessTime)

 case (message in BC is to build 400):
    update CARTRIDGES
       set Status = 'waiting for program'
          where (Code = '1000194') and
                (Id = :IdCartridge) and
                (Status = 'at the test table');
 case (message in BC is to build 400D):
    update CARTRIDGES
       set Status = 'waiting for program'
          where (Code = '1000193') and
                (Id = :IdCartridge) and
                (Status = 'at the test table');
```

```
   case (message in BC is to build 500):
      update CARTRIDGES
         set Status = ‘waiting for program’
            where (Code = ‘1000219’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test table’);
   case (message in BC is to build 500D):
      update CARTRIDGES
         set Status = ‘waiting for program’
            where (Code = ‘1000218’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test table’);

Place CTWSWP: 400 waiting for program
create view CTWSWP as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
            (Id = :IdCartridge) and
            (Status = ‘waiting for program’);

Place CTWDWP: 400D waiting for program
create view CTWDBWP as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
            (Id = :IdCartridge) and
            (Status = ‘waiting for program’);

Place CTBSWP: 500 waiting for program
create view CTBSWP as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
            (Id = :IdCartridge) and
            (Status = ‘waiting for program’);

Place CTBDWP: 500D waiting for program
create view CTBDWP as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
            (Id = :IdCartridge) and
            (Status = ‘waiting for program’);
```

```
Transition MTW: Move the cartridge up near to the writer
routine MTW (accept WRTT,
                    CTWSWP guard 400   to build message in BC,
                    CTWDWP guard 400D to build message in BC,
                    CTBSWP  guard 500    to build message in BC,
                    CTBDWP  guard 500D to build message in BC;
             return    WRUC,
                    CTWSNW guard 400   to build message in BC,
                    CTWDNW guard 400D to build message in BC,
                    CTBSNW  guard 500    to build message in BC,
                    CTBDNW  guard 500D to build message in BC)

  /* The warehouse robot take the cartridge up near to the writer. */
   grabcartridge(WR,cartridge)
   waitfor(ProcessTime)
   move_up(WR,cartridge)
   waitfor(ProcessTime)

   update WAREHOUSE_ROBOTS
      set status = 'up the cartridge'
         where Status = 'at the test table';

   case (message in BC is to build 400):
      update CARTRIDGES
         set Status = 'near writer'
            where (Code = '1000194') and
                  (Id = :IdCartridge) and
                  (Status = 'waiting for program');
   case (message in BC is to build 400D):
      update CARTRIDGES
         set Status = 'near writer'
            where (Code = '1000193') and
                  (Id = :IdCartridge) and
                  (Status = 'waiting for program');
   case (message in BC is to build 500):
      update CARTRIDGES
         set Status = 'near writer'
            where (Code = '1000219') and
                  (Id = :IdCartridge) and
                  (Status = 'waiting for program');
   case (message in BC is to build 500D):
      update CARTRIDGES
         set Status = 'near writer'
            where (Code = '1000218') and
                  (Id = :IdCartridge) and
                  (Status = 'waiting for program');
```

```
Place CTWSNW: 400 ready to test
create view CTWSNWT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
                (Id = :IdCartridge) and
                (Status = ‘near writer’);

Place CTWDNW: 400D ready to test
create view CTWDBNW as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
                (Id = :IdCartridge) and
                (Status = ‘near writer’);

Place CTBSNW: 500 ready to test
create view CTBSNW as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
                (Id = :IdCartridge) and
                (Status = ‘near writer’);

Place CTBDNW: 500D ready to test
create view CTBDNW as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
                (Id = :IdCartridge) and
                (Status = ‘near writer’);

Place WRUC: up the cartridge
/* The warehouse robot  brings up the cartridge to the writer */
create view WRUC as
 select *
     from  WAREHOUSE_ROBOTS
       where Status = ‘up the cartridge’;
```



Transition WRF: Write on the RFID tag of the cartridge the data from the writer routine WRF (accept WRUC,

CTWSNW guard 400 to build message in BC,
CTWDNW guard 400D to build message in BC,
CTBSNW guard 500 to build message in BC,
CTBDNW guard 500D to build message in BC;
return WRRJ,
CTWSWT guard 400 to build message in BC,
CTWDWT guard 400D to build message in BC,
CTBSWT guard 500 to build message in BC,
CTBDWT guard 500D to build message in BC)

```
/* Wait for successful message, if override needed get pressed. */
 command(write_cartridge)
 waitfor(ProcessTime)
 If not check_message('successful') then press(override)
 waitfor(ProcessTime)
 move_down(WR,cartridge)
 waitfor(ProcessTime)
 put_table(WR,cartridge)
 waitfor(ProcessTime)

 update WAREHOUSE_ROBOTS
    set status = 'ready for jar'
       where Status = 'up the cartridge';

 case (message in BC is to build 400):
    update CARTRIDGES
       set Status = 'waiting for test'
          where (Code = '1000194') and
                (Id = :IdCartridge) and
                (Status = 'near writer');
```

```
    case (message in BC is to build 400D):
       update CARTRIDGES
          set Status = ‘waiting for test’
             where (Code = ‘1000193’) and
                      (Id = :IdCartridge) and
                      (Status = ‘near writer’);
    case (message in BC is to build 500):
       update CARTRIDGES
          set Status = ‘waiting for test’
             where (Code = ‘1000219’) and
                      (Id = :IdCartridge) and
                      (Status = ‘near writer’);
    case (message in BC is to build 500D):
       update CARTRIDGES
          set Status = ‘waiting for test’
             where (Code = ‘1000218’) and
                      (Id = :IdCartridge) and
                      (Status = ‘near writer’);

Place CTWSWT: 400 waiting for test
create view CTWSWT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
               (Id = :IdCartridge) and
               (Status = ‘waiting for test’);

Place CTWDWT: 400D waiting for test
create view CTWDBWT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
               (Id = :IdCartridge) and
               (Status = ‘waiting for test’);

Place CTBSWT: 500 waiting for test
create view CTBSWT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
               (Id = :IdCartridge) and
               (Status = ‘waiting for test’);

Place CTBDWT: 500D waiting for test
create view CTBDWT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
               (Id = :IdCartridge) and
               (Status = ‘waiting for test’);
```

Place WRRJ: ready for jar
/* *The warehouse robot* ready to open the gas jar */
create view WRRJ as
select *
from WAREHOUSE_ROBOTS
where Status = ‘ready for jar’;

Transition MJ: move to the jar
routine MJ (accept WRRJ;
return WRJ)
/* The warehouse robot move to the jar to open it. */
command(open_jar)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
set status = ‘at the jar’
where Status = ‘ready for jar’;

Place WRJ: at the jar
/* *The warehouse robot* brings up the cartridge to the writer */
create view WRUC as
select *
from WAREHOUSE_ROBOTS
where Status = ‘at the jar’;

Place JCE: Air Jar empty
/* The jar can be detected empty by making calculation about the consuming of every cartridge, by checking the indicator on the jar, by checking the error of Air Supply on the tester or manually by declaring a transition that change the status */
create view JCE as
select *
from JARS
where Volume = ‘empty’;

Transition CJ: change the empty jar
routine CJ (accept JCE;
return JCNE)
/* The computer sends a series of commands to the warehouse robot to change the jar. The command of Changejar is a high level command consisting of many basic commands to make the change. *The system send an email adverting that the jar is empty*. */
changejar(WR,jar)
waitfor(ProcessTime)
sendemail(‘email.com’,’jar empty’)

Place JCNE: Air Jar not empty
/* The jar can be detected empty by making calculation about the consuming of every cartridge, by checking the indicator on the jar, by checking the error of Air Supply on the tester or manually by declaring a transition that change the status */
create view JCE as
select *
from JARS
where Volume = ‘not empty’;

Transition OJ: open to the jar
routine OJ (accept WRJ,
JCNE;
return WRJO,
JO)
/* The warehouse robot open the jar. If any check on the pressure instrument should be done to recognize the emptiness of the jar, more Places / Transitions should be written. */
command(open_jar)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
set status = 'jar opened'
where Status = 'at the jar';

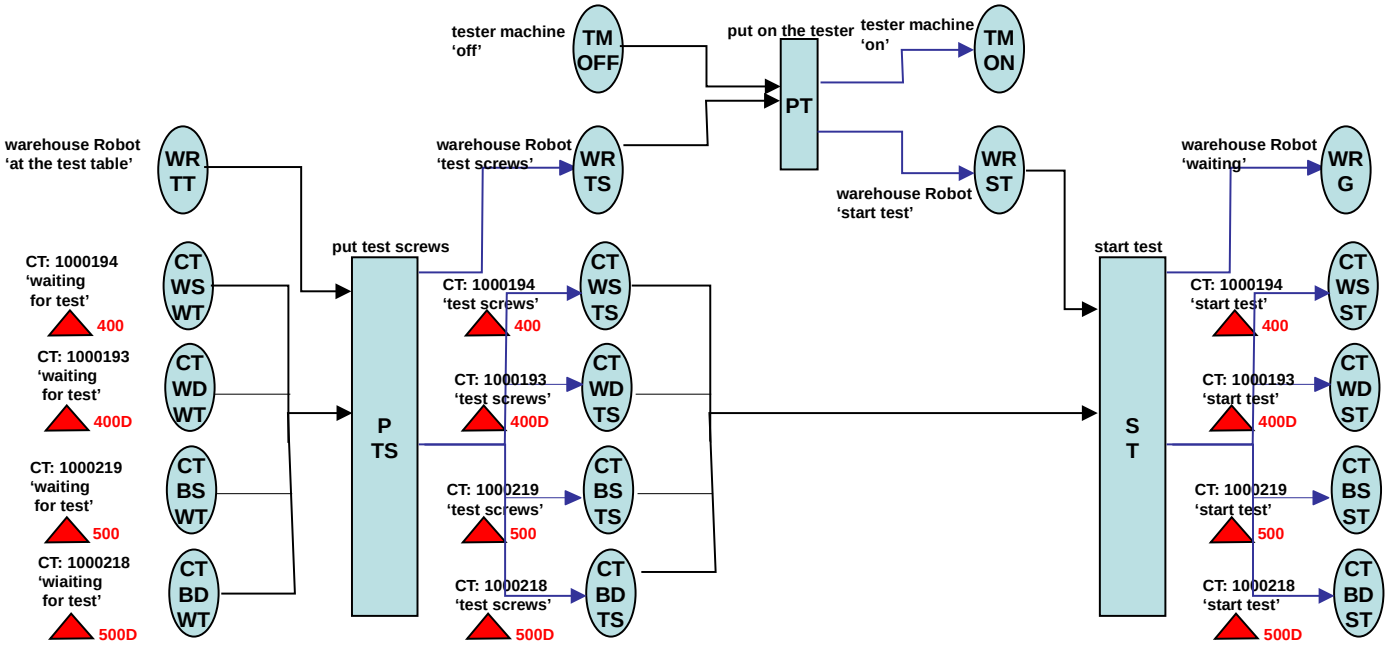
update JARS
set Status = 'open'
where Status = 'close';

Place WRJO: Jar opened
create view WRJO as
select *
from WAREHOUSE_ROBOTS
where Status = 'jar opened';

Place JO: Jar open
create view JO as
select *
from JARS
where Status = 'open';

Transition MT: Move to the test table
routine MT (accept WRJO;
return WRTT)
/* The warehouse robot return to the test table. */

update WAREHOUSE_ROBOTS
set status = 'at the test table'
where Status = 'jar opened';



Transition PTS: Put test screws
routine PTS (accept WRTT,
CTWSWT guard 400 to build message in BC,
CTWDWT guard 400D to build message in BC,
CTBSWT guard 500 to build message in BC,
CTBDWT guard 500D to build message in BC;
return WRTS,
CTWSTS guard 400 to build message in BC,
CTWDTS guard 400D to build message in BC,
CTBSTS guard 500 to build message in BC,
CTBDTS guard 500D to build message in BC)

/* The warehouse robot take the round plate connected to the air tube and place it on the cartridge after moving it 180 degrees. The screws get inserted and the screw driver moving in four directions to insert the fourth screws. The transition itself could be expanded in transitions */ **places for every step down.**

```
grabcartridge(WR,cartridge)
/* Hold the cartridge */
waitfor(ProcessTime)
flip(WR,cartridge)
/* The Warehouse Robot turn 180 degrees */
waitfor(ProcessTime)
puttube(WR,cartridge)
/* The warehouse robot put the tube on the cartridge */
waitfor(ProcessTime)
insert(WR,screws)
/* insert the 4 screws into the holes using a clamp */
waitfor(ProcessTime)
Move(Wr,firstholescrew);
waitfor(ProcessTime)
command(largescrewdriver);
waitfor(ProcessTime)
```

```
Move(Wr,thirdholescrew);
waitfor(ProcessTime)
command(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,secondholescrew);
waitfor(ProcessTime)
command(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,fourthholescrew);
waitfor(ProcessTime)
command(largescrewdriver);
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
   set status = ‘test screws’
      where Status = ‘at the test table’;

case (message in BC is to build 400):
   update CARTRIDGES
      set Status = ‘test screws’
         where (Code = ‘1000194’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test’);
case (message in BC is to build 400D):
   update CARTRIDGES
      set Status = ‘test screws’
         where (Code = ‘1000193’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test’);
case (message in BC is to build 500):
   update CARTRIDGES
      set Status = ‘test screws’
         where (Code = ‘1000219’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test’);
case (message in BC is to build 500D):
   update CARTRIDGES
      set Status = ‘test screws’
         where (Code = ‘1000218’) and
                  (Id = :IdCartridge) and
                  (Status = ‘at the test’);
```

```
Place CTWSTS: 400 waiting for test
create view CTWSTS as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
                (Id = :IdCartridge) and
                (Status = ‘test screws’);

Place CTWDTS: 400D waiting for test
create view CTWDBTS as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
                (Id = :IdCartridge) and
                (Status = ‘test screws’);

Place CTBSTS: 500 waiting for test
create view CTBSTS as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
                (Id = :IdCartridge) and
                (Status = ‘test screws’);

Place CTBDTS: 500D waiting for test
create view CTBDTS as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
                (Id = :IdCartridge) and
                (Status = ‘test screws’);

Place WRTS: Test screws
create view WRTS as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘test screws’;
```

Place TMOFF: Test screws
create view TMOFF as
select *
from TESTER_MACHINE
where Status = ‘off’;

Transition PT: Put on the tester machine
routine PT (accept WRTS,
TMOFF;
return WRTS,
TMON)

/* The warehouse robot put the tester machine on, because the button is on the back there is an extension to make it turn on from the side. */

turnon(WR,tester_machine)
/* Hold the cartridge */
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
set status = ‘start test’
where Status = ‘test screws’;

update TESTER_MACHINE
set status = ‘on’
where Status = ‘off’;

Place TMON: Tester machine set to on
create view TMON as
select *
from TESTER_MACHINE
where Status = ‘on’;

Place WRTS: start test
create view WRTS as
select *
from WAREHOUSE_ROBOTS
where Status = ‘start test’;

```
Transition ST: Put test screws
routine ST (accept WRST,
                    CTWSTS guard 400   to build message in BC,
                    CTWDTS guard 400D to build message in BC,
                    CTBSTS  guard 500    to build message in BC,
                    CTBDTS  guard 500D to build message in BC;
              return WRG,
                    CTWSST guard 400   to build message in BC,
                    CTWDST guard 400D to build message in BC,
                    CTBSST  guard 500    to build message in BC,
                    CTBDST  guard 500D to build message in BC)

  /* The warehouse robot press the start button to start the test and go in waiting status. */

  pressstart(WR,testermachine)
     /* Press the start button */
  waitfor(ProcessTime)

  update WAREHOUSE_ROBOTS
     set status = 'waiting'
        where Status = 'start test';

  case (message in BC is to build 400):
     update CARTRIDGES
        set Status = 'start test'
           where (Code = '1000194') and
                    (Id = :IdCartridge) and
                    (Status = 'test screws');
  case (message in BC is to build 400D):
     update CARTRIDGES
        set Status = 'start test'
           where (Code = '1000193') and
                    (Id = :IdCartridge) and
                    (Status = 'test screws');
  case (message in BC is to build 400):
     update CARTRIDGES
        set Status = 'start test'
           where (Code = '1000219') and
                    (Id = :IdCartridge) and
                    (Status = 'test screws');
  case (message in BC is to build 500D):
     update CARTRIDGES
        set Status = 'start test'
           where (Code = '1000218') and
                    (Id = :IdCartridge) and
                    (Status = 'test screws');
```

Place CTWSST: 400 waiting for test

```
create view CTWSST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
                (Id = :IdCartridge) and
                (Status = ‘start test’);
```

Place CTWDST: 400D waiting for test

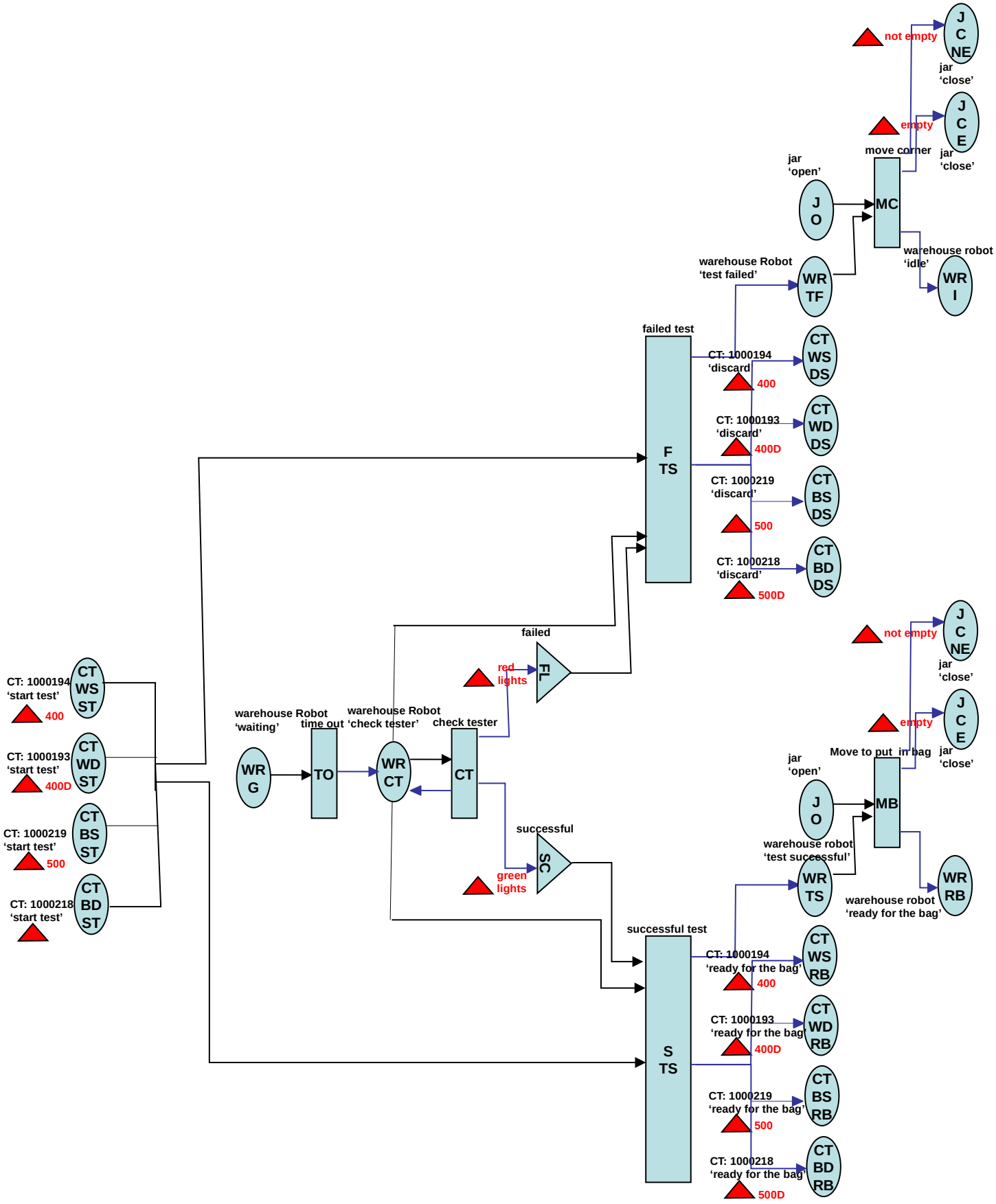
```
create view CTWDBST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
                (Id = :IdCartridge) and
                (Status = ‘start test’);
```

Place CTBSST: 500 waiting for test

```
create view CTBSST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
                (Id = :IdCartridge) and
                (Status = ‘start test’);
```

Place CTBDST: 500D waiting for test

```
create view CTBDST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
                (Id = :IdCartridge) and
                (Status = ‘start test’);
```


CT: 1000194 'start test'
CT WS ST
400
CT: 1000193 'start test'
CT WD ST
400D
CT: 1000219 'start test'
CT BS ST
500
CT: 1000218 'start test'
CT BD ST
warehouse Robot 'waiting'
WR G
time out
TO
warehouse Robot 'check tester'
WR CT
check tester
CT
red lights
FL
failed
green lights
SC
successful
F TS
failed test
CT: 1000194 'discard'
400
CT WS DS
CT: 1000193 'discard'
400D
CT WD DS
CT: 1000219 'discard'
500
CT BS DS
CT: 1000218 'discard'
500D
CT BD DS
S TS
successful test
CT: 1000194 'ready for the bag'
400
CT WS RB
CT: 1000193 'ready for the bag'
400D
CT WD RB
CT: 1000219 'ready for the bag'
500
CT BS RB
CT: 1000218 'ready for the bag'
500D
CT BD RB
warehouse Robot 'test failed'
WR TF
jar 'open'
J O
MC
move corner
not empty
J C NE
jar 'close'
empty
J C E
jar 'close'
warehouse robot 'idle'
WR I
jar 'open'
J O
MB
Move to put in bag
not empty
J C NE
jar 'close'
empty
J C E
jar 'close'
warehouse robot 'test successful'
WR TS
warehouse robot 'ready for the bag'
WR RB

Transition TO: Timer out
routine TO (accept WRG;
return WRCT)

/* The computer after sending the command to the warehouse robot to start the machine, a timer event start, when it is elapsed, the warehouse robot move to a state to check the tester machine. */

```
update WAREHOUSE_ROBOTS
   set status = 'check tester'
      where Status = 'waiting';
```

Place WRCT: start test
create view WRTS as

```
select *
   from WAREHOUSE_ROBOTS
      where Status = 'check tester';
```

Transition CT: check tester machine
routine CT (accept WRCT;
return WRCT,
FL guard red lights,
SC guard green lights)

/* The computer check the lights on the tester machine or read the message to see 'successful' word to send either message of failure or successful. */

```
if successful(WR,tester_Machine) then
  generate_message(SC)
 else
  generate_message(FL)
```

Message FL: cartridge test failed

Message SC: cartridge test successful

Transition FTS: Failed test cartridge
routine FTS (accept WRCT,
FS,
CTWSST guard 400 to build message in BC,
CTWDST guard 400D to build message in BC,
CTBSST guard 500 to build message in BC,
CTBDST guard 500D to build message in BC;
return WRTF,
CTWSDS guard 400 to build message in BC,
CTWDDS guard 400D to build message in BC,
CTBSDS guard 500 to build message in BC,
CTBDDS guard 500D to build message in BC)

/* The cartridge fail to pass the test. The warehouse robot unscrew the test screws and discard the cartridge in the discard bin for later check. */

Move(Wr,firstscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,thirdscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,secondscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,fourthscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
gettube(WR,cartridge)
/* The warehouse robot put the tube on the cartridge */
waitfor(ProcessTime)
flip(WR,cartridge)
/* The Warehouse Robot turn 180 degrees */
waitfor(ProcessTime)
discardinbin(WR,cartridge)
waitfor(ProcessTime)

```
   update WAREHOUSE_ROBOTS
      set status = 'test failed'
         where Status = 'check tester';

   case (message in BC is to build 400):
      update CARTRIDGES
         set Status = 'discard'
            where (Code = '1000194') and
                  (Id = :IdCartridge) and
                  (Status = 'start test');
   case (message in BC is to build 400D):
      update CARTRIDGES
         set Status = 'discard'
            where (Code = '1000193') and
                  (Id = :IdCartridge) and
                  (Status = 'start test');
   case (message in BC is to build 500):
      update CARTRIDGES
         set Status = 'discard'
            where (Code = '1000219') and
                  (Id = :IdCartridge) and
                  (Status = 'start test');
   case (message in BC is to build 500D):
      update CARTRIDGES
         set Status = 'discard'
            where (Code = '1000218') and
                  (Id = :IdCartridge) and
                  (Status = 'start test');

Place CTWSDS: 400 discard
create view CTWSST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = '1000194') and
               (Id = :IdCartridge) and
               (Status = 'discard');

Place CTWDDS: 400D discard
create view CTWDBST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = '1000193') and
               (Id = :IdCartridge) and
               (Status = 'discard');
```

```
Place CTBSDS: 500 discard
create view CTBSST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
              (Id = :IdCartridge) and
              (Status = ‘discard’);

Place CTBDDS: 500D discard
create view CTBDST as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
              (Id = :IdCartridge) and
              (Status = ‘discard’);

Place WRTF: test failed
create view WRTS as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘test failed’;

Transition MC: move corner
routine MC (accept WRTF,
                          JO;
                return WRI,
                        JCNE guard Volume = ‘not empty’,
                        JCE    guard Volume = ‘empty’)

   /* The computer after sending the command to the warehouse robot to start the
       machine, a timer event start, when it is elapsed, the warehouse robot move to a
     state  to check the tester machine. */

    update WAREHOUSE_ROBOTS
       set status = ‘idle’
          where Status = ‘test failed’;
    update JARS
      set Status = ‘close’
         where Status = ‘open’;
    if is_empty(Jar) then
       update JARS
         set Volume = ‘empty’
            where Volume = ‘not empty’;
```

Transition STS: Successful test cartridge

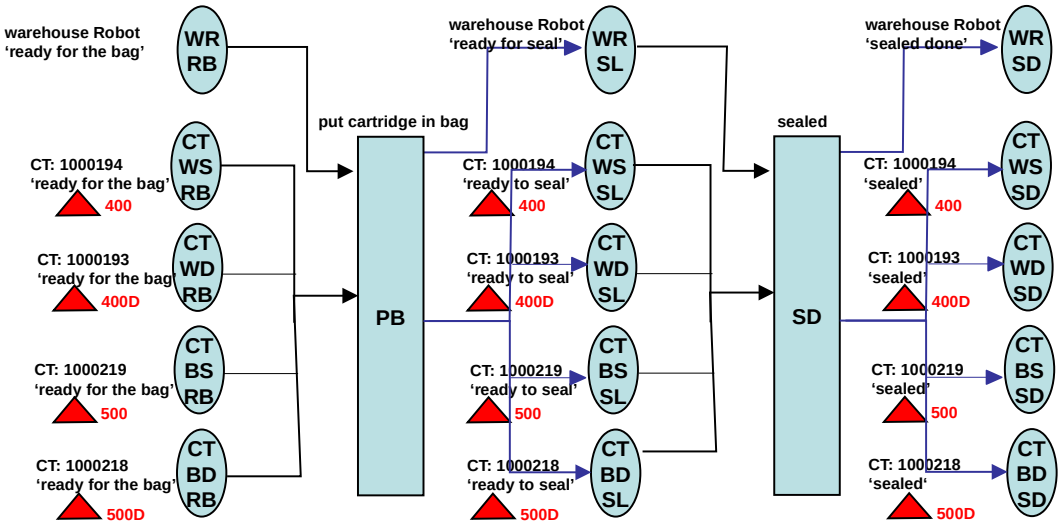
```
routine STS (accept WRCT,
                 FS,
                 CTWSST guard 400   to build message in BC,
                 CTWDST guard 400D to build message in BC,
                 CTBSST  guard 500    to build message in BC,
                 CTBDST  guard 500D to build message in BC;
             return   WRTS,
                 CTWSRB guard 400   to build message in BC,
                 CTWDRB guard 400D to build message in BC,
                 CTBSRB  guard 500    to build message in BC,
                 CTBDRB  guard 500D to build message in BC)
```

/* The cartridge pass the test. The warehouse robot unscrew the test screws and prepare the cartridge to put it in bag. */

```
Move(Wr,firstscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,thirdscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,secondscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
Move(Wr,fourthscrew);
waitfor(ProcessTime)
unscrew(largescrewdriver);
waitfor(ProcessTime)
gettube(WR,cartridge)
/* The warehouse robot put the tube on the cartridge */
waitfor(ProcessTime)
flip(WR,cartridge)
/* The Warehouse Robot turn 180 degrees */
waitfor(ProcessTime)
```

```
   update WAREHOUSE_ROBOTS
      set status = ‘test successful’
         where Status = ‘check tester’;

   case (message in BC is to build 400):
      update CARTRIDGES
         set Status = ‘ready for the bag’
            where (Code = ‘1000194’) and
                  (Id = :IdCartridge) and
                  (Status = ‘start test’);
   case (message in BC is to build 400D):
      update CARTRIDGES
         set Status = ‘ready for the bag’
            where (Code = ‘1000193’) and
                  (Id = :IdCartridge) and
                  (Status = ‘start test’);
   case (message in BC is to build 500):
      update CARTRIDGES
         set Status = ‘ready for the bag’
            where (Code = ‘1000219’) and
                  (Id = :IdCartridge) and
                  (Status = ‘start test’);
   case (message in BC is to build 500D):
      update CARTRIDGES
         set Status = ‘ready for the bag’
            where (Code = ‘1000218’) and
                  (Id = :IdCartridge) and
                  (Status = ‘start test’);

Place CTWSRB: 400 ready for the bag
create view CTWSRB as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for the bag’);

Place CTWDRB: 400D ready for the bag
create view CTWDBRB as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for the bag’);
```

```
Place CTBSRB: 500 ready for the bag
create view CTBSRB as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
              (Id = :IdCartridge) and
              (Status = ‘ready for the bag’);

Place CTBDRB: 500D ready for the bag
create view CTBDRB as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
              (Id = :IdCartridge) and
              (Status = ‘ready for the bag’);

Place WRTS: test successful
create view WRTS as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘test successful’;

Transition MB: move to put in bag
routine MB (accept WRTS,
                          JO;
                return WRRB,
                        JCNE guard Volume = ‘not empty’,
                        JCE    guard Volume = ‘empty’)

   /* The computer after sending the command to the warehouse robot to start the
       machine, a timer event start, when it is elapsed, the warehouse robot move to a
       state  to check the tester machine. */

    update WAREHOUSE_ROBOTS
       set status = ‘ready for the bag’
          where Status = ‘test successful’;
    update JARS
      set Status = ‘close’
         where Status = ‘open’;
    if is_empty(Jar) then
       update JARS
         set Volume = ‘empty’
            where Volume = ‘not empty’;
```



Transition PB: Put cartridge in bag

```
routine PB (accept WRRB,
                    CTWSRB guard 400   to build message in BC,
                    CTWDRB guard 400D to build message in BC,
                    CTBSRB  guard 500    to build message in BC,
                    CTBDRB  guard 500D to build message in BC;
              return  WRSL,
                    CTWSSL guard 400   to build message in BC,
                    CTWDSL guard 400D to build message in BC,
                    CTBSSL  guard 500    to build message in BC,
                    CTBDSL  guard 500D to build message in BC)

   /* The Warehouse Robot put the cartridge inside a bag.  */

   putinbag(WR,cartridge)
   waitfor(ProcessTime)

   update WAREHOUSE_ROBOTS
      set status = 'ready for seal'
         where Status = 'ready for the bag';

   case (message in BC is to build 400):
      update CARTRIDGES
         set Status = 'ready for seal'
            where (Code = '1000194') and
                  (Id = :IdCartridge) and
                  (Status = 'ready for the bag');
   case (message in BC is to build 400D):
      update CARTRIDGES
         set Status = 'ready for seal'
            where (Code = '1000193') and
                  (Id = :IdCartridge) and
                  (Status = 'ready for the bag');
```

```
    case (message in BC is to build 500):
       update CARTRIDGES
          set Status = ‘ready for seal’
             where (Code = ‘1000219’) and
                    (Id = :IdCartridge) and
                    (Status = ‘ready for the bag’);
    case (message in BC is to build 500D):
       update CARTRIDGES
          set Status = ‘ready for seal’
             where (Code = ‘1000218’) and
                    (Id = :IdCartridge) and
                    (Status = ‘ready for the bag’);

Place CTWSSL: 400 ready to seal
create view CTWSSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for seal’);

Place CTWDSL: 400D ready to seal
create view CTWDBSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for seal’);

Place CTBSSL: 500 ready to seal
create view CTBSSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for seal’);

Place CTBDSL: 500D ready to seal
create view CTBDSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
               (Id = :IdCartridge) and
               (Status = ‘ready for seal’);

Place WRSL: warehouse robot ready for seal
create view WRSL as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘ready for seal’;
```

Transition SD: sealed
routine SD (accept WRSL,
CTWSSL guard 400 to build message in BC,
CTWDSL guard 400D to build message in BC,
CTBSSL guard 500 to build message in BC,
CTBDSL guard 500D to build message in BC;
return WRSD,
CTWSSD guard 400 to build message in BC,
CTWDSD guard 400D to build message in BC,
CTBSSD guard 500 to build message in BC,
CTBDSD guard 500D to build message in BC)

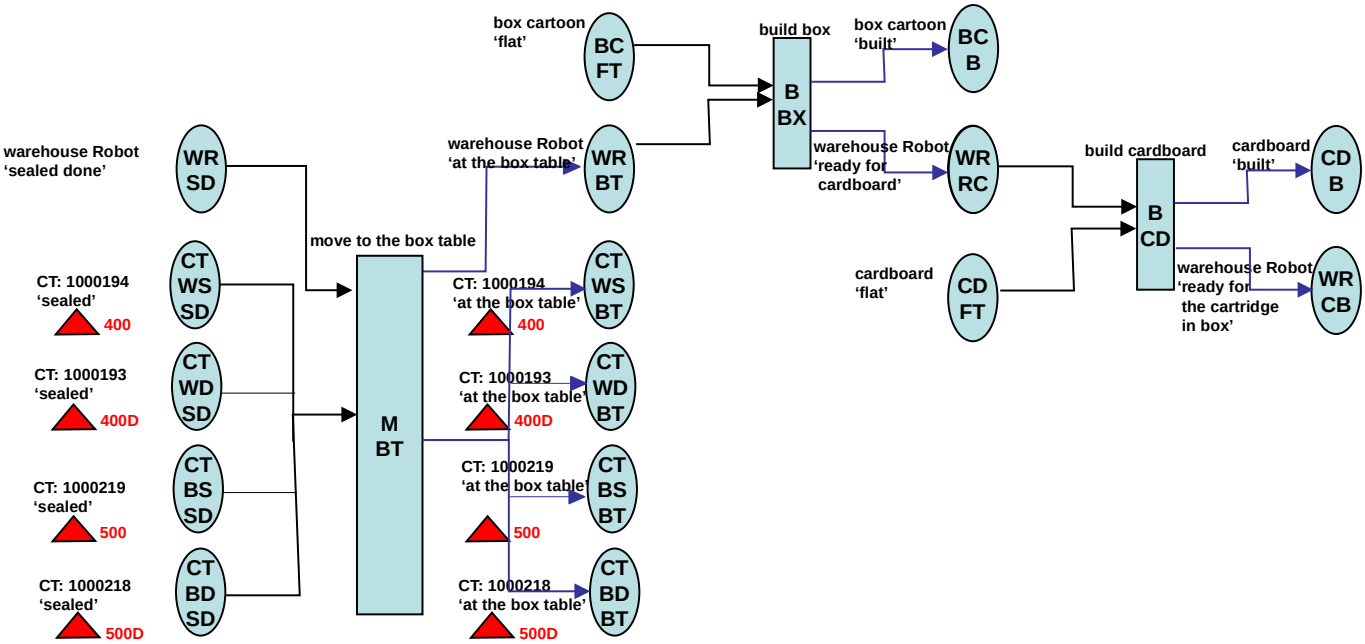
/* The machine for the sealed is under the bag machine. One the Warehouse Robot put the cartridge inside a bag, it press the sealed machine to seal the bag. */

sealbag(WR,cartridge)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
set status = 'sealed done'
where Status = 'ready for the bag';

case (message in BC is to build 400):
update CARTRIDGES
set Status = 'sealed'
where (Code = '1000194') and
(Id = :IdCartridge) and
(Status = 'ready for seal');
case (message in BC is to build 400D):
update CARTRIDGES
set Status = 'sealed'
where (Code = '1000193') and
(Id = :IdCartridge) and
(Status = 'ready for seal');
case (message in BC is to build 500):
update CARTRIDGES
set Status = 'sealed'
where (Code = '1000219') and
(Id = :IdCartridge) and
(Status = 'ready for seal');
case (message in BC is to build 500D):
update CARTRIDGES
set Status = 'sealed'
where (Code = '1000218') and
(Id = :IdCartridge) and
(Status = 'ready for seal');

```
Place CTWSSD: 400 sealed
create view CTWSSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
                (Id = :IdCartridge) and
                (Status = ‘sealed’);

Place CTWDSD: 400D sealed
create view CTWDBSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
                (Id = :IdCartridge) and
                (Status = ‘sealed’);

Place CTBSSD: 500 sealed
create view CTBSSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
                (Id = :IdCartridge) and
                (Status = ‘sealed’);

Place CTBDSD: 500D sealed
create view CTBDSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
                (Id = :IdCartridge) and
                (Status = ‘sealed’);

Place WRSD: warehouse robot sealed done
create view WRSL as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘sealed done’;
```



Transition MBT: Move to the box table
routine MBT (accept WRSD,
CTWSSD guard 400 to build message in BC,
CTWDSD guard 400D to build message in BC,
CTBSSD guard 500 to build message in BC,
CTBDSD guard 500D to build message in BC;
return WRBT,
CTWSBT guard 400 to build message in BC,
CTWDBT guard 400D to build message in BC,
CTBSBT guard 500 to build message in BC,
CTBDBT guard 500D to build message in BC)

/* The Warehouse Robot bring the bag of cartridge to the box table. */

```
grab(WR,cartridge_bag)
waitfor(ProcessTime)
move_robot(WR,box_table)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
   set status = 'at the box table'
      where Status = 'sealed done';

case (message in BC is to build 400):
   update CARTRIDGES
      set Status = 'at the box table'
         where (Code = '1000194') and
               (Id = :IdCartridge) and
               (Status = 'sealed');
case (message in BC is to build 400D):
   update CARTRIDGES
      set Status = 'at the box table'
         where (Code = '1000193') and
               (Id = :IdCartridge) and
               (Status = 'sealed');
```

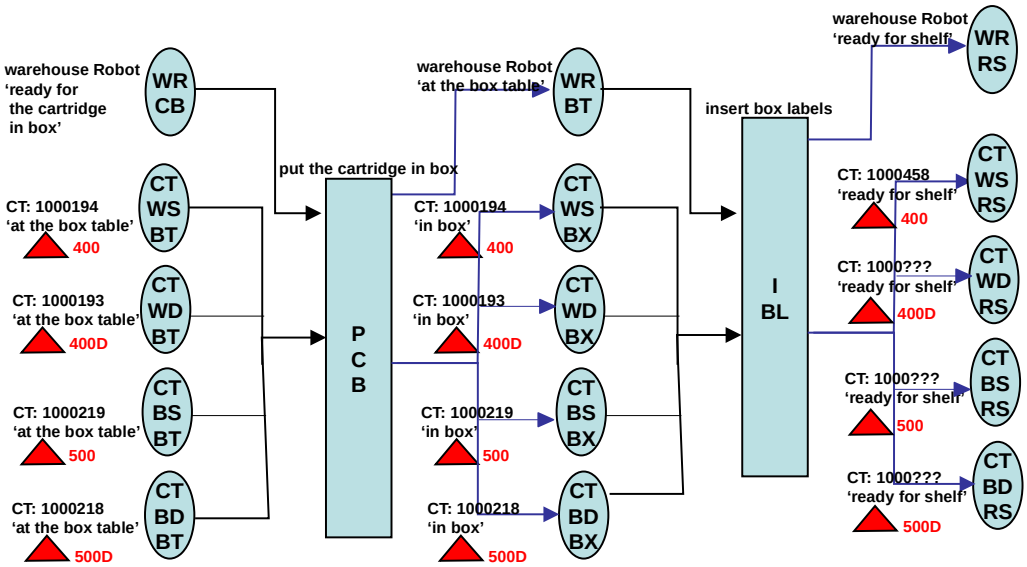
```
    case (message in BC is to build 500):
       update CARTRIDGES
          set Status = ‘at the box table’
             where (Code = ‘1000219’) and
                      (Id = :IdCartridge) and
                      (Status = ‘sealed’);
    case (message in BC is to build 500D):
       update CARTRIDGES
          set Status = ‘at the box table’
             where (Code = ‘1000218’) and
                      (Id = :IdCartridge) and
                      (Status = ‘sealed’);

Place CTWSBT: 400 at the box table
create view CTWSBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTWDBT: 400D at the box table
create view CTWDBBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTBSBT: 500 at the box table
create view CTBSBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTBDBT: 500D at the box table
create view CTBDBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place WRBT: warehouse robot at the box table
create view WRSL as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘at the box table’;
```

Transition BBX: Build the box for the cartridge

```
routine BBX (accept WRBT,
                   BCFT;
             return  WRRC,
                   BCB)

   /* The Warehouse Robot use the box cartoon flat to build the box. The Right back is
        folded 90 degrees followed by the left back side. Later the back side of the flat
        box cartoon is bring up 90 degrees. The right middle side of the flat box is
        turned 180 degrees followed by the left middle side. */

   fold(WR,rightback_box)
   waitfor(ProcessTime)
   fold(WR,leftback_box)
   waitfor(ProcessTime)
   fold(WR,back_box)
   waitfor(ProcessTime)
   fold(WR,right_box)
   waitfor(ProcessTime)
   fold(WR,leftt_box)
   waitfor(ProcessTime)

   update WAREHOUSE_ROBOTS
      set status = 'ready for cardboard'
         where Status = 'at the box table';

    update BOX_CARTOON
      set status = 'built'
         where Status = 'flat';
```

Transition BCD: Build the cardboard for the box

```
routine BCD (accept WRRC,
                    CDFT;
             return WRCB,
                    CDB)

   /* The Warehouse Robot use the cardboard to build for the box. The Right edge is
        folded 90 degrees down followed by the left edge. Later the back side of the flat
        cardboard is bring up 90 degrees followed by the front side.  */

   fold(WR,right_edge)
   waitfor(ProcessTime)
   fold(WR,left_edge)
   waitfor(ProcessTime)
   fold(WR,back_edge)
   waitfor(ProcessTime)
   fold(WR,front_edge)
   waitfor(ProcessTime)

   update WAREHOUSE_ROBOTS
      set status = 'ready for cardboard'
         where Status = 'at the box table';

    update BOX_CARTOON
      set status = 'built'
         where Status = 'flat';
```



Transition PCB: Put the cartridge in box
routine PCB (accept WRCB,
CTWSBT guard 400 to build message in BC,
CTWRBT guard 400D to build message in BC,
CTBRBT guard 500 to build message in BC,
CTBRBT guard 500D to build message in BC;
return WRBT,
CTWSBX guard 400 to build message in BC,
CTWDBX guard 400D to build message in BC,
CTBSBX guard 500 to build message in BC,
CTBDBX guard 500D to build message in BC)

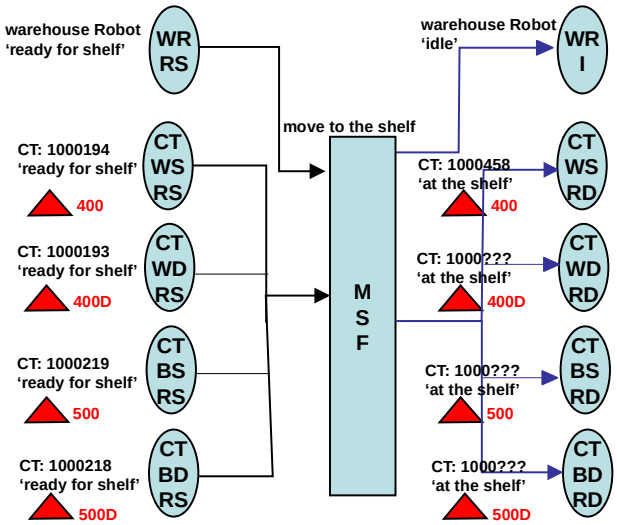
/* The Warehouse Robot put the cartridge in the box, close the front part of the box by folding it the far away part 90 degrees later the nearest part . */

```
putinbox(WR,bag_cartridge)
waitfor(ProcessTime)
fold(WR,far_part)
waitfor(ProcessTime)
fold(WR,near_part)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
   set status = 'at the box table'
      where Status = 'ready for the cartridge in box';

case (message in BC is to build 400):
   update CARTRIDGES
      set Status = 'in box'
         where (Code = '1000194') and
               (Id = :IdCartridge) and
               (Status = 'at the box table');
```

```
    case (message in BC is to build 400D):
       update CARTRIDGES
          set Status = ‘in box’
             where (Code = ‘1000193’) and
                      (Id = :IdCartridge) and
                      (Status = ‘at the box table’);
    case (message in BC is to build 500):
       update CARTRIDGES
          set Status = ‘in box’
             where (Code = ‘1000219’) and
                      (Id = :IdCartridge) and
                      (Status = ‘at the box table’);
    case (message in BC is to build 500D):
       update CARTRIDGES
          set Status = ‘in box’
             where (Code = ‘1000218’) and
                      (Id = :IdCartridge) and
                      (Status = ‘at the box table’);

Place CTWSSL: 400 in box
create view CTWSSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000194’) and
               (Id = :IdCartridge) and
               (Status = ‘in box’);

Place CTWDSL: 400D in box
create view CTWDBSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000193’) and
               (Id = :IdCartridge) and
               (Status = ‘in box’);

Place CTBSSL: 500 in box
create view CTBSSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000219’) and
               (Id = :IdCartridge) and
               (Status = ‘in box’);

Place CTBDSL: 500D in box
create view CTBDSL as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000218’) and
               (Id = :IdCartridge) and
               (Status = ‘in box’);
```

```
Transition IBL: Insert Box Labels
routine IBL (accept WRBT,
                    CTWSBX guard 400   to build message in BC,
                    CTWDBX guard 400D to build message in BC,
                    CTBSBX  guard 500    to build message in BC,
                    CTBDBX  guard 500D to build message in BC;
             return  WRRS,
                     CTWSRS guard 400   to build message in BC,
                     CTWDRS guard 400D to build message in BC,
                     CTBSRS  guard 500    to build message in BC,
                     CTBDRS  guard 500D to build message in BC)

  /* There is many ways to make the printing of the labels an automatic process: bring
      the warehouse robot to the printer, change the rolls for labels and ink. Make the
       computer simulate the login and access to printer software and writing the lot
      code and printing automatically by sending the right APIs to the mouse and
      keyboard or request a printer from the manufacturer with APIs to call for
      printing. One time printing is ready, the robot bring them to the table and roll
      them over a special device on the wall. The device roll one label at a time and a
      clamp take it and insert it in the right position on the bottom – left of the top box
  */

  Insertlabels(WR,box)
  waitfor(ProcessTime)

  update WAREHOUSE_ROBOTS
     set status = 'ready for shelf'
        where Status = 'at the box table';

  case (message in BC is to build 400):
     update CARTRIDGES
        set Code = '1000458', Status = 'ready for shelf'
           where (Code = '1000194') and
                    (Id = :IdCartridge) and
                    (Status = 'in box');
  case (message in BC is to build 400D):
     update CARTRIDGES
        set Code = '1000???', Status = 'ready for shelf'
           where (Code = '1000193') and
                    (Id = :IdCartridge) and
                    (Status = 'in box');
  case (message in BC is to build 500):
     update CARTRIDGES
        set Code = '1000???', Status = 'ready for shelf'
           where (Code = '1000219') and
                    (Id = :IdCartridge) and
                    (Status = 'in box');
  case (message in BC is to build 500D):
     update CARTRIDGES
        set Code = '1000???', Status = 'ready for shelf'
           where (Code = '1000218') and
                    (Id = :IdCartridge) and
                    (Status = 'in box');
```

```
Place CTWSRS: 400 ready for shelf
create view CTWSSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000458’) and
                (Id = :IdCartridge) and
                (Status = ‘ready for shelf’);

Place CTWDRS: 400D ready for shelf
create view CTWDBSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
                (Id = :IdCartridge) and
                (Status = ‘ready for shelf’);

Place CTBSRS: 500 ready for shelf
create view CTBSSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
                (Id = :IdCartridge) and
                (Status = ‘ready for shelf’);

Place CTBDRS: 500D ready for shelf
create view CTBDSD as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
                (Id = :IdCartridge) and
                (Status = ‘ready for shelf’);

Place WRRS: warehouse robot sealed done
create view WRSL as
    select *
       from WAREHOUSE_ROBOTS
          where Status = ‘ready for shelf’;
```



Transition MSF: Bring the cartridge to the shelf
routine MSF (accept WRRS,
CTWSRS guard 400 to build message in BC,
CTWDRS guard 400D to build message in BC,
CTBSRS guard 500 to build message in BC,
CTBDRS guard 500D to build message in BC;
return WRI,
CTWSRD guard 400 to build message in BC,
CTWDRD guard 400D to build message in BC,
CTBSRD guard 500 to build message in BC,
CTBDRD guard 500D to build message in BC)

/* The Warehouse Robot bring the box of cartridge to the shelf. */

```
grab(WR,cartridge_box)
waitfor(ProcessTime)
move_robot(WR,shelf)
waitfor(ProcessTime)
put(WR,cartridge_box)
waitfor(ProcessTime)
move_robot(WR,corner)
waitfor(ProcessTime)

update WAREHOUSE_ROBOTS
   set status = 'idle'
      where Status = 'ready for shelf';

case (message in BC is to build 400):
   update CARTRIDGES
      set Status = 'at the shelf'
         where (Code = '1000458') and
               (Id = :IdCartridge) and
               (Status = 'ready for shelf');
```

```
    case (message in BC is to build 400D):
       update CARTRIDGES
          set Status = ‘at the shelf’
             where (Code = ‘1000???’) and
                      (Id = :IdCartridge) and
                      (Status = ‘ready for shelf’);
    case (message in BC is to build 500):
       update CARTRIDGES
          set Status = ‘at the shelf’
             where (Code = ‘1000???’) and
                      (Id = :IdCartridge) and
                      (Status = ‘ready for shelf’);
    case (message in BC is to build 500D):
       update CARTRIDGES
          set Status = ‘at the shelf’
             where (Code = ‘1000???’) and
                      (Id = :IdCartridge) and
                      (Status = ‘ready for shelf’);

Place CTWSRD: 400 at the shelf
create view CTWSBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000458’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTWDRD: 400D at the shelf
create view CTWDBBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTBSRD: 500 at the shelf
create view CTBSBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);

Place CTBDRD: 500D at the shelf
create view CTBDBT as
  select *
    from  CARTRIDGES
      where (cartridges.Code = ‘1000???’) and
               (Id = :IdCartridge) and
               (Status = ‘at the box table’);
```