**Data**

Private class MyKit{

Kit kit

MyKitStatus MKS;

…

}

enum MyKitStatus{NotDone, Done}

Private class Arm{

Part part

ArmStatus AS

…

}

enum ArmStatus{Empty, Full}

List<PartType> KitConfig

List<MyKit> MyKits //Only 2

Map<Nest,List<Part> parts> GoodParts

List<Arm> Arms //Only 4

//Shared data with KitRobot, //Stand

List<Kit> KitsOnStand

List<Nest> nests

Stand stand

//Prevent collisions (not for v0)

//Shared with PartsRobot.

Semaphore AccessKit

**Messages**

//From FCS

HereIsKitConfiguration(List<PartType> config){

KitConfig = config;

}

//From Camera

HereAreGoodParts(Map<Nest, List<Part> parts){

//May overwrite current keys in //GoodParts but that’s ok

For ∀ key in parts

GoodParts.put(key, parts.get(key));

}

//From Kit Robot

UseThisKit(Kit k){

MyKit mk = new MyKit(k, KitConfig);

MyKits.add(mk);

}**Scheduler**

If(MyKits.size > 0){

If Ǝ mk in MyKits ϵ mk.MKS = Done

Call RequestInspection(mk)

If Ǝ mk in MyKits ϵ mk.MKS = NotDone and

for ∀ a in Arms, a.AS = Full

Call PlaceParts()

}

If(GoodParts.size > 0 and Ǝ a in Arms ϵ a.AS = Empty

Call PickUpPart(a)

**Actions**

PickUpPart(Arm a){

Part pickUpPart;

Int nestIndex

//If no kits available, try

//and pick up a different part

//in each arm while waiting for a kit

if(MyKits.size < 1){

if(Arms.isEmpty(){ //Pick up first part

nestIndex = first nest in GoodParts;

List<part> available = GoodParts.get(nestIndex);

pickUpPart = available.get(0);

} else {

If Ǝ p in GoodParts ϵ for ∀ a in Arms, Arms.part != p

nestIndex = GoodParts.indexOf(p)

pickUpPart = p;

else {

//We have a duplicate, just pick something //unless 2 arms already have that part

if Ǝ p in GoodParts ϵ for exactly 1 a in Arms, a.part = p

nestIndex = GoodParts.indexOf(p)

pickUpPart = available.get(0);

}

}

} else { //There are kits on the stand

for (mk : MyKits)

If Ǝ p in GoodParts ϵ mk.kit.isNeeded(p) = true

nestIndex = GoodParts.indexOf(p);

pickUpPart = p

// else don’t pick up anything as

//the kits don’t need anything

//from the current batch of parts

Nests.get(nestIndex).TakingPart(pickUpPart);

DoPickUpPart(part);

Nests.get(nestIndex).DonePickingUpParts();

}

PlaceParts(){

//O(n­2)…bad but it works

For(mk: MyKits){

For (a:Arms){

Part = a.part;

If(mk.isNeeded(a.part){

Mk.kit.parts.put(Part);

Mk.kit.partsExpected.remove(Part);

a.part = null;

DoPlacePart(Part, mk.kit);

}

} //end inner for

CheckMyKit(mk);

} // end outer for

}

CheckMyKit(MyKit mk){

If(mk.kit.partsExpected.size = 0)

//no more parts expected

mk.KS = Done;

}

RequestInspection(MyKit mk){

Stand.KitIsDone(mk.kit);

MyKits.remove(mk);

}

DoPickUpPart(part p){ //Animation}

DoPlacePart(Part p, Kit k){

//Animation}