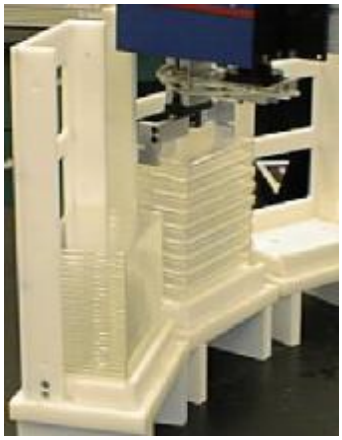
# Help sheet 21 – stacks and piles

## Using stacks or piles.

This is done as above by creating a Cartesian ROW. The dimensions of the stack might not be so well known.

Normally you would have some sort of holder to restrain the items in X and Y, as on the right. However if you are placing objects on a pile with no requirement to pick them up again then a loose pile will work.



Suppose you have a stack that must hold up to 20 plastic microplates like those on the right.

1 First stack 20 plates in the holder.
2 Create a route 20 lines high – reserve at least 21 in case you want an approach position.
3 Using Jog take the robot to the top plate and learn line 20
4 Move robot away and remove all but the bottom plate.
5 Using Jog teach line 1.
6 Click interpolate.
7 If you want an approach position, for example to carefully lower a plate on to the one below it then create the approach as above, e.g.
0 0 100 MOVE for 10mm
then click **set approach**.

You can then unload the stack very simply like this.

```
: FILLSTACK
STACK           ( STACK is the name of the route
ALIGN
20 1 DO
    GETPLATE    ( whatever that may entail
    I INTO      ( go to the tray, next position
    UNGRIP      ( put plate in the tray
    UP
LOOP
;
```

Or empty a stack like this:

```
: GETSTACK
STACK           ( STACK is the name of the route
ALIGN
20 1 DO
    I INTO      ( go to the tray
    GRIP
    UP
    PUTPLATE    ( whatever that may entail
LOOP
;
```

# Help sheet 21 – stacks and piles

Of course you may have more than one stack. In that case you would need a variable for each stack, to keep count.

```
USER STK1
USER STK2
: GET1
STAK1     ( name of route for stack 1 - don't forget only the first 5
characters are recognized. STACK1 and STACK2 would not work.
STK1 @    ( get the count for stack1 from STK1
INTO      ( go to the position noted in STK1
GRIP
UP
-1 STK1 +! ( decrement the value in STK1
;
```

then PUTPLATE or whatever is next.

```
: GET2
STAK2     ( name of route for stack 1 - don't forget only the first 5
characters are recognized. STACK1 and STACK2 would not work.
STK2 @    ( get the count for stack1 from STK1
INTO      ( go to the position noted in STK1
GRIP
UP
-1 STK2 +! ( decrement the value in STK2
;
: INIT
CR ." fill stacks with plates. Press enter. " KEY DROP
20 STK1 ! 20 STK2 !
;
```

Or to put a plate back:

```
: PUT1
STAK1     ( name of route for stack 1 - don't forget only the first 5
characters are recognized. STACK1 and STACK2 would not work.
STK1 @    ( get the count for stack1 from STK1
INTO      ( go to the position noted in STK1
UNGRIP
UP
1 STK1 +! ( decrement the value in STK1
;
```

Make sure you don't over-fill the stack

```
: PUT1
STAK1     ( name of route for stack 1 - don't forget only the first 5
characters are recognized. STACK1 and STACK2 would not work.
STK1 @    ( get the count for stack1 from STK1
DUP 19 > IF 101 ABORT THEN
INTO      ( go to the position noted in STK1
UNGRIP
UP
1 STK1 +! ( decrement the value in STK1
;
```

If your program or user tries to put in more than 20 the system will abort. Enter ERR ? and you will see 101 (or whatever code you allocate) which tells you why the system stopped.