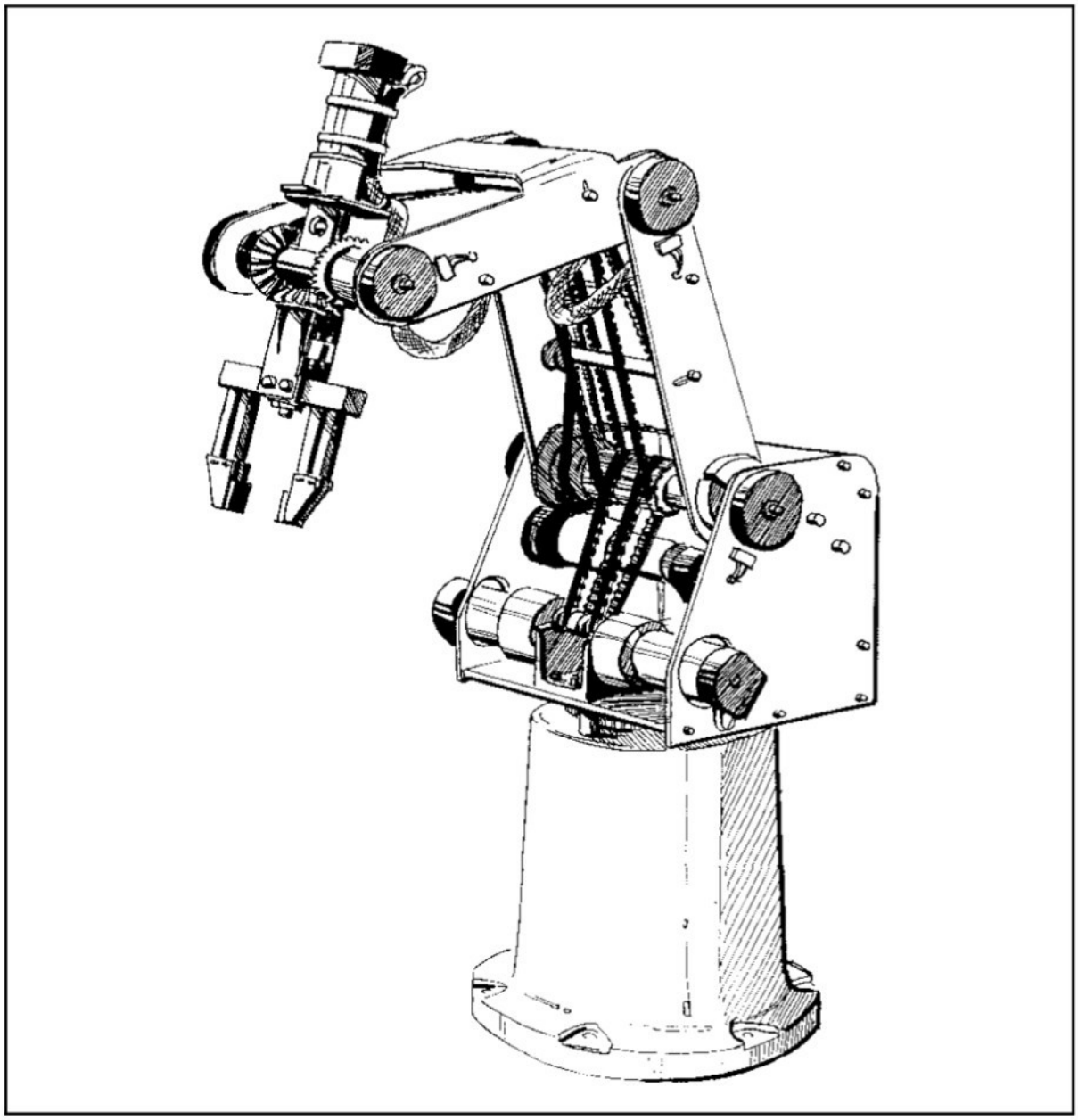
PRÁCTICA 2 ACL:



Alumna: Yanierkis Justina Armario Aguirre.
Grado: GITI.

ÍNDICE

Ejercicio 1: Paradas de emergencia y recuperación. Interrupciones externas. TRIGGER……2

Apartado A……2

Apartado B……6

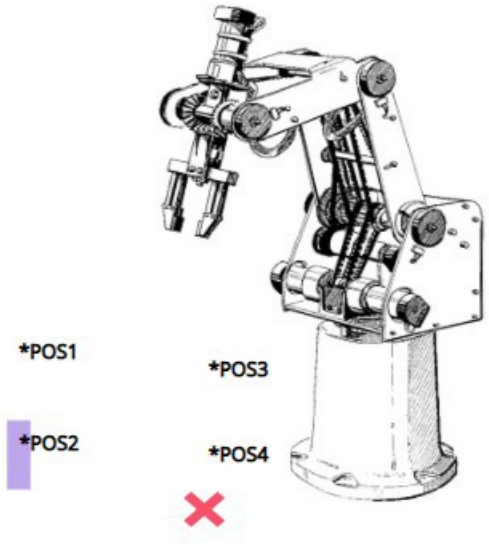
Ejercicio 2: Sincronización de dos procesos – PEND/POST……..11

Ejercicio 3: Sincronización de múltiples procesos – QPEND/QPOST……13

Ejercicio 1: Paradas de emergencia y recuperación. Interrupciones externas. TRIGGER.

Apartado A. Modificar el programa del ejercicio 1 de la práctica 1 de Scorbot para que en el momento en que se conecte un interruptor externo, se detenga el robot y vaya inmediatamente a una posición de descanso definida previamente.

En este apartado tomaremos uno de los programas realizados en la práctica anterior, como la finalidad de este programa es la de ver que funciona el uso de *TRIGGER* para detener un programa en curso mediante un interruptor hemos optado por la situación más sencilla. Mover una pieza de una posición inicial a otra final.



El objetivo es que si durante el transcurso de dicho programa se conecta un interruptor externo, el programa se pare y vaya a una posición de descanso definida previamente. Nuestra posición de descanso será la del HOME. Existe un problema si paramos o suspendemos un programa que se encuentra ejecutando una orden de movimiento y es que realizará esa orden, nosotros buscamos que se detenga por completo, por lo que deberemos borrar el buffer con el comando *CLRBUF* justo después de parar o suspender un programa.

Para una mejor comprensión de los programas utilizamos mostraremos un pseudocódigo de cada uno de ellos:

Programa EJ1:

1) Partimos de la posición del HOME.
2) Nos movemos a la posición donde se encuentra la pieza y la cogemos reduciendo la velocidad en la posición de aproximación.
3) Volvemos, con la pieza, a la posición de HOME reduciendo la velocidad hasta llegar a la posición de aproximación y luego poniéndola a su valor normal hasta que llega al HOME.
4) Colocamos la pieza en la posición final y la soltamos reduciendo la velocidad cuando sea conveniente.

5) El brazo robótico volverá a su posición de inicio (en el HOME) reduciendo la velocidad en el tramo citado anteriormente.

Programa PARO:

Se mantiene en espera y cuando el interruptor se ponga a ‘1’ correrá el programa PAU.

Programa PAU:

1. Para el programa EJ1.
2. Borra el Buffer.
3. Va a una posición de descanso.

Programa PRINC:

Corre simultáneamente los programas EJ1 y PARO.

El código es el siguiente:

Hay que tener en cuenta que antes de la realización de los programas hay que definir y darles valores a nuestras posiciones. Nos moveremos manualmente con nuestro brazo robótico y una vez estemos en la posición deseada la definimos y grabamos.

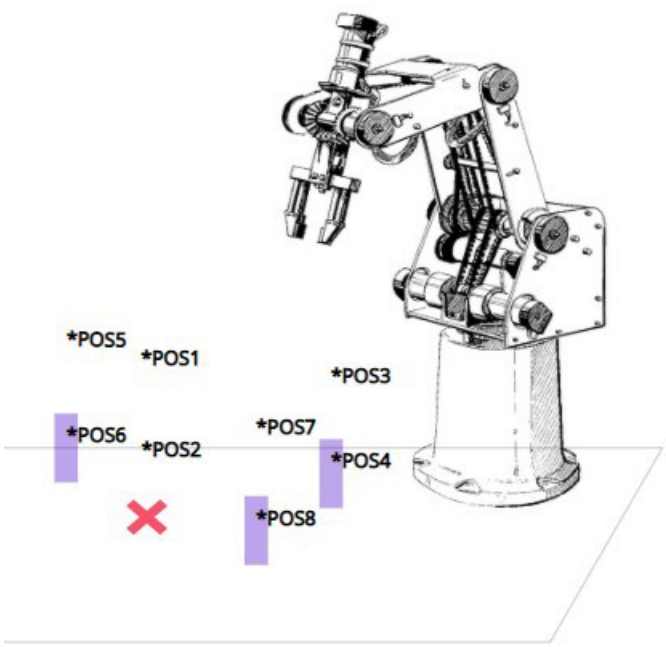
```
DEFP POS1
HERE POS1
DEFP POS2
HERE POS2
DEFP POS3
HERE POS3
DEFP POS4
HERE POS4
DEFP POS5          //Esta será nuestra posición de reposo.
HERE POS5

EDIT EJ1                    //Creamos el programa
*****************************************************************************************
SPEED 25
MOVED 0
MOVED POS1          //Se mueve a la pos de aproximación
OPEN                //Es aquí donde abre la garra
SPEED 10            //Reducimos la velocidad
MOVED POS2
CLOSE               //Cerramos y cogemos la pieza
MOVED POS1          //Volvemos a la pos de aprox
SPEED 25            //Volvemos a la velocidad habitual fuera de la zona de peligro
MOVED 0
```

```
MOVED POS3
SPEED 10
MOVED POS4
OPEN                    //Depositamos la pieza
MOVED POS3
CLOSE
SPEED 25
MOVED 0
EXIT
************************************************************************************************
```

Para ejecutar el programa habría que utilizar el comando *RUN EJ2*.

```
          PROGRAM   PAU
          **********************

STOP     EJ1
CLRBUF                  //Así nos aseguramos que se detiene por completo
SPEED     25
MOVED POS5          //La posición de descanso
END



          PROGRAM   PARO
          **********************

TRIGGER   PAU BY  IN 1 1   //Se mantiene a la espera de que el interruptor se ponga a
                                    ‘ 1’
END

          PROGRAM   PRINC
          **********************

RUN      EJ1
RUN      PARO
END
```

Apartado B. Realizar un programa que haga lo siguiente:
• El robot estará inicialmente en una posición de descanso.
• Cuando se conecte un interruptor externo, el SCORBOT empezará a ejecutar el programa que apila tres piezas realizado en la práctica 1.
• En el instante en que se desconecte el interruptor, el robot se parará inmediatamente. Si tenía alguna pieza cogida la dejará en una posición intermedia y luego irá a la posición de descanso.
• Si se vuelve a conectar el interruptor, el robot continuará ejecutando el programa anterior (si hay alguna pieza en la posición intermedia, la cogerá primero).



En este código ocurre lo mencionado anteriormente, al suspender el programa hay que ejecutar el comando *CLRBUF* para borrar el Buffer. Además de esto, como vamos a suspender el programa y luego reanudarlo, será necesario saber si el programa tenía una pieza en la garra o no, del mismo modo, también sera conveniente saber si la garra se encontraba abierta o cerrada.

Para una mejor comprensión de los programas utilizamos mostraremos un pseudocódigo de cada uno de ellos:

Programa E2E:

Se coloca en una pos de descanso y espera a que el interruptor se ponga a ‘1’ para ejecutar el programa de apilar las piezas.

Programa START:

Estará a la espera de que el interruptor se desconecte para ejecutar el programa QUITO.

(1) Partimos de una posición inicial HOME.

(2) Cogemos la primera pieza y volvemos a la posición HOME.
(3) Colocamos la primera pieza en su posición final, actualizamos la posición de aprox para que ahora esté 10cm por encima y volvemos a la pos HOME.
(4) Cogemos la segunda pieza y realizamos un proceso análogo al paso anterior.
(5) Cogemos la tercera pieza y la colocamos en la pos final, quedando las tres piezas alineadas verticalmente.
(6) Volvemos a la pos del HOME y finalizamos el programa.

Durante este programa se indicarán los estados de la garra y si esta tiene pieza o no.

Programa QUITO:

1. Suspendo el programa START.
2. Borro el buffer.
3. Si tengo una pieza la coloco el una posición segura.
4. Cierro la garra y nos movemos a la pos HOME.
5. Esperamos a que se active el interruptor para ejecutar el programa CONT.

Programa CONT:

1. Ver si estaba en el estado con pieza, si es así recogerla de la posición segura.
2. Ver si la garra estaba abierta o cerrada y abrirla o cerrarla.
3. Continuar el programa START por el punto en el que se suspendió.
4. Mantenerse a la espera de que el interruptor se ponga a cero para volver a ejecutar el programa QUITO.

El código de los programas será el siguiente:

Tenemos que tener en cuenta que las posiciones desde POS1 a POS5 ya están definidas así que sólo definimos y grabamos las restantes.

```
GLOBAL GARR   //Definimos la variable para señalar el estado de la garra
GLOBAL PIEZA   //Lo mismo para saber si tiene o no una pieza la garra
DEFP AUX1        //Definimos variables auxiliares de posición
DEFP AUX2
DEFP P5           //Definimos y grabamos nuestra pos segura
HERE P5
DEFP P6
HERE P6
DEFP POS6      //Def y creamos las pos y las pos de aprox de las piezas
HERE POS6
DEFP POS7
HERE POS7
DEFP POS8
HERE POS8
```

```
        PROGRAM   E2E
          **********************

   SPEED     25
   MOVED     0          //Nos movemos a una posición de descanso
   WAIT      IN[1] = 1    //Esperamos a que se conecte el interruptor
   RUN       START      //Empezamos nuestro programa
   END



           PROGRAM   START
           **********************

TRIGGER   QUITO BY  IN 1 0   //Realizará el programa QUITO cuando el interruptor se
                                desconecte
SPEED     25
MOVED     0
CLOSE
SET       GARR = 0             //Se encuentra cerrada
SET       PIEZA = 0            //No tiene pieza
MOVED     POS3
SPEED     10
OPEN
SET       GARR = 1           //La garra está abierta
MOVED     POS4
CLOSE
SET       GARR = 0
SET       PIEZA = 1          //Cogemos la primera pieza
MOVED     POS3
SPEED     25
MOVED     0
MOVED     POS1
SETP      AUX1=POS1  //Usamos esto para que el SHITFC no machaque a POS1
SETP      AUX2=POS2
SPEED     10
MOVED     POS2
OPEN
SET       GARR = 1
SET       PIEZA = 0          //Colocamos la primera pieza en la pos final
MOVED     POS1
SPEED     25
CLOSE
SET       GARR = 0
MOVED     0
SHIFTC    AUX1 BY Z  440       //Actualizamos las posiciones de aprox y colocación de la
                                 siguiente pieza
SHIFTC    AUX2 BY Z  440
SPEED     25
MOVED     POS7
```

```
OPEN
SET      GARR = 1
SPEED    10
MOVED    POS8
CLOSE
SET      GARR = 0
SET      PIEZA = 1                  //Cogemos la segunda pieza
MOVED    POS7
SPEED    25
MOVED    0
MOVED    AUX1
SPEED    10
MOVED    AUX2
OPEN
SET      GARR = 1
SET      PIEZA = 0              //Colocamos en la pos final la segunda pieza
MOVED    AUX1
CLOSE
SET      GARR = 0
SPEED    25
MOVED    0
SHIFTC   AUX1 BY Z  440      //Actualizamos
SHIFTC   AUX2 BY Z  440
MOVED    POS5
OPEN
SET      GARR = 1
SPEED    10
MOVED    POS6
CLOSE
SET      GARR = 0
SET      PIEZA = 1          //Cogemos la tercera pieza
MOVED    POS5
SPEED    25
MOVED    0
MOVED    AUX1
SPEED    10
MOVED    AUX2
OPEN
SET      GARR = 1
SET      PIEZA = 0         //Colocamos la tercera pieza
MOVED    AUX1
CLOSE
SET      GARR = 0
SPEED    25
MOVED    0
END
```

Nota: No hace falta resetear la posición de AUX1 o AUX2 al fnal del programa ya que al empezar el programa de nuevo, le daremos los valores de POS1 y POS2.

```
        PROGRAM   QUITO
          **********************

SUSPEND   START   //Suspendo el programa de apilar piezas
CLRBUF            //Borro el buffer
IF        PIEZA = 1      //Si tiene una pieza la llevo a una pos segura
  SPEED     25
  MOVED     0
  MOVED     P5
  SPEED     10
  MOVED     P6
  OPEN
  MOVED     P5
ENDIF
CLOSE             //Cierro la garra y me voy a la pos HOME
SPEED     25
MOVED     0
TRIGGER   CONT BY  IN 1 1   //A la espera de que se conecte el interruptor para
                                continuar el programa de apilar piezas
END

          PROGRAM   CONT
          **********************

SPEED     25
IF        PIEZA = 1           //Si el estado era con pieza la recojo de la pos segura
  MOVED     0
  MOVED     P5
  SPEED     10
  OPEN
  MOVED     P6
  CLOSE
  MOVED     P5
  SPEED     25
  MOVED     0
ENDIF
IF        GARR = 1            //si la garra estaba en estado abierto, la abro
  OPEN
ENDIF
IF        GARR = 0            //Si estaba en estado cerrado, la cierro
  CLOSE
ENDIF
CONTINUE  START
TRIGGER   QUITO BY  IN 1 0
END
```

Podríamos implementar el programa también añadiendo el estado LENTO, cuando esté a ‘1’ significará que el programa está en SPEED 10 y cuando valga ‘0’ que está en SPEED 25, así evitaríamos posibles problemas de que vaya a una pos muy rápido o lento de lo que debería.

Ejercicio 2: Sincronización de dos procesos – PEND/POST.

Realizar las siguientes operaciones:
• Definir dos posiciones (POS1 y POS2) separadas unos 10 cm.
• Escribir dos programas que se ejecutarán de forma simultánea. El primero de ellos vigilará la entrada de un interruptor externo, calculará el tiempo (T) que transcurre desde que se conecta el interruptor hasta que se desconecta, y transmitirá el tiempo T al segundo programa.
• El segundo programa esperará hasta que reciba el tiempo T del primer programa e imprimirá T por pantalla. Seguidamente comprobará que T es mayor que un valor (por ejemplo 3 segundos) y ordenará que el robot se mueva desde POS1 hasta POS2 en un tiempo T, y regrese a POS1 también en un tiempo T.

Para una mejor comprensión de los programas utilizamos mostraremos un pseudocódigo de cada uno de ellos:

PROGRAMA UN1:

Es el programa que se encarga de medir el tiempo.

1. Espera a que se active el interruptor y se le asigna un valor en tiempo (a través de un contador interno que tiene el programa).
2. Espera a que se desconecte dicho interruptor y se le asigna un valor en tiempo a otra variable.
3. Como queremos saber el tiempo transcurrido entre esas dos acciones, restamos el tiempo de apagado al tiempo de encendido, y le asignamos ese valor a otra variable (RESUL).
4. El valor de RESUL es el que le asamos a nuestra variable global TIEMP que la usará en el segundo programa.
5. Finalizamos el programa.

PROGRAMA DO1:

En este programa supondremos que si el tiempo es menos de 3 segundos es un error y no hará nada.

1. Definimos la variable TIEM y le asignamos un valor inicial de 0.
2. Mostramos el tiempo por pantalla.
3. Cuando se pase el valor de la variable global calculada desde el programa UN1 a nuestra variable TIEM y comprobemos que esta es mayor que 3s podremos continuar.

4. Si esta condición se cumple, primero nos movemos con una velocidad rápida a la POS1.
5. Desde la POS1 se mueve a la POS2 en el tiempo señalado.
6. Esperamos un poco para que se aprecie más el movimiento.
7. Volvemos a movernos, pero esta vez desde la POS2 a la POS1 en el mismo tiempo calculado.
8. Finalizamos el programa.

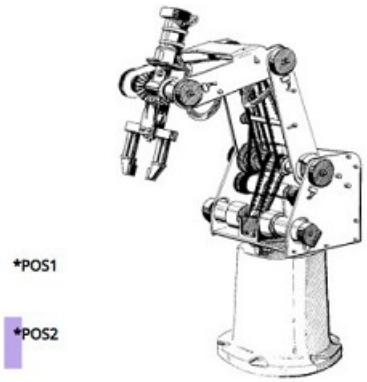
PROGRAMA E333:

Este es el programa principal del ejercicio, es el encargado de ejecutar los dos programas anteriores.

1. Vamos a una velocidad rápida a una posición de reposo.
2. Ejecutamos el programa UN1.
3. Ejecutamos el programa DO1.
4. Finalizamos el programa.

El código de los programas será el siguiente:

Utilizaremos las posiciones POS1 y POS2 utilizadas anteriormente por lo que no será necesario volverlas a definir. También tenemos que tener en cuenta que para el uso de los comandos *PEND* y *POST* hay que trabajar con una variable definida global, en este caso usaremos una variable llamada TIEMP.




```
GLOBAL TIEMP   //Designamos nuestra variable global

        PROGRAM   UN1
          **********************
   DEFINE CLOC1            //Variable interna que indica el momento inicial
   DEFINE CLOC2            //Variable interna que indica el momento final
   DEFINE RESUL            //Será donde se calcula el tiempo transcurrido entre las dos
                             acciones
   WAIT     IN[1] = 1        //Esperamos a que se encienda el interruptor
   SET      CLOC1 = TIME   //Cuando eso ocurra le damos una valor del tiempo a esta
                             variable
   WAIT     IN[1] = 0                  //Hacemos lo mismo cuando se desconecte
   SET      CLOC2= TIME
   SET      RESUL=CLOC2 – CLOC1        //Calculamos el tiempo transcurrido
   POST     RESUL TO TIEMP          //Pasamos el valor a la variable global TIEMP
   END
```

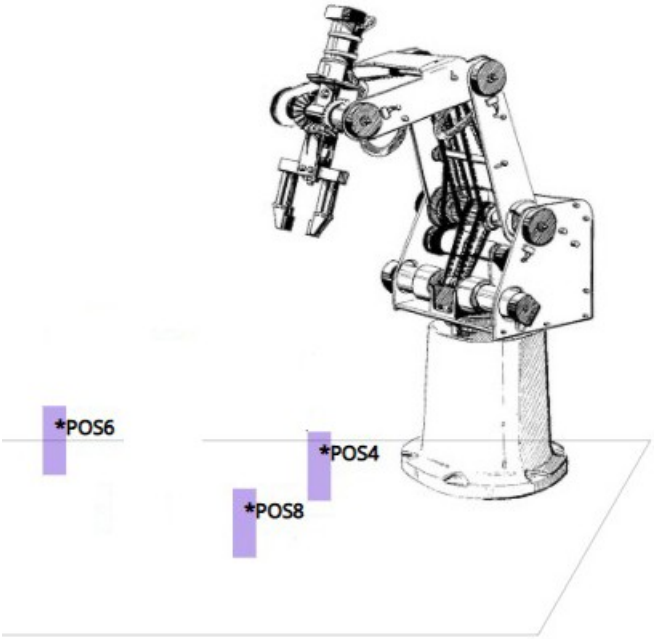
```
        PROGRAM   DO1
          **********************
   DEFINE TIEM      //Definimos la variable con la que trabajaremos dentro del programa
   SET TIEM=0                    //Le asignamos un valor inicial
   PEND      TIEM FROM TIEMP     //Cuando se ejecute el programa UN1 y traslada el
valor de RESUL a la variable global TIEMP es cuando esta se enviará aquí, a la variable
TIEM
   PRINTLN   "EL TIEMPO ES:"  TIEM  //Mostramos por pantalla el valor de TIEM
   IF       TIEM > 300   //Hay que comprobar que sea mayor de 3s
    SPEED     25
    MOVED     POS1      //Se colocará inicialmente en la POS1
    MOVED     POS2 TIEM   //Desde la POS1 irá a la POS2 en el tiempo señalado
    DELAY     300       //Esperamos 3s para que sea más apreciable el movto
    MOVED     POS1 TIEM   //Desde la POS2 nos movemos a la POS1 en el tiempo
                          señalado.
   ENDIF                  //Terminamos la condición
   END

          PROGRAM   E333
          **********************

   SPEED     25
   MOVE      0            //Nos colocamos en una posición de reposo
   RUN       UN1          //Ejecutamos el programa para calcular el tiempo
   RUN       DO1          //Ejecutamos el programa que realiza el movimiento
   END                    //Finalizamos el programa
```

Ejercicio 3: Sincronización de múltiples procesos - QPEND/QPOST.

Se tienen 3 piezas colocadas en lugares conocidos. Hacer 4 programas que se ejecutarán de forma simultánea:

- El primero de ellos, esperará a que le envíen valores a la cola (1, 2 ó 3), y cuando reciba alguno,cogerá la pieza correspondiente, la levantará unos 20 cm. y la volverá a colocar.
- El segundo programa vigilará el interruptor externo. Cuando se conecte el interruptor, enviará un “1” a la cola para que el robot coja la pieza número 1.
- El tercer programa pedirá por pantalla que se le den números del 1 al 3, y los irá enviando a la cola.
- El cuarto programa imprimirá en la pantalla el contenido de la cola cada 5 segundos.

Reutilizaremos la situación del ejercicio 1, por lo que tampoco será necesario volver a definir las posiciones.



Los comandos *QPEND* y *QPOST* actúan igual que los comandos *PEND* y *POST*, la diferencia es que estos trabajan con valores de cola. Igual que en el ejercicio anterior, deberemos definir una variable global para la cola.

La idea de estos programa es que se realicen continuamente, es decir, no solo una vez y luego parar, ya que buscamos rellenar la cola y realiza las acciones pertinentes. Para ello, utilizaremos los comandos *LABEL* y *GOTO* para, una vez finalizado un programa, que salte a la primera línea y así crear una especie de bucle.

Además de estos comandos también utilizaremos la orden *SHIFTC* ya que uno de los programas nos piden coger una de las piezas, y levantarla 20cm para luego volver a colocarla en su posición original.

En el tercer programa leeremos por pantalla el número que nos indique el usuario. Para ello usaremos el comando *READ* que también permite mostrar por pantalla lo que le indiquemos mientras esté entre comillas. Esto nos ahorrará el uso de un *PRINT*.

Para una mejor comprensión de los programas utilizamos mostraremos un pseudocódigo de cada uno de ellos:

PROGRAMA PRI:

Programa que recibido un número por cola, va a la posición correspondiente, sube y baja la pieza.

1. El programa esperará a que le envíen a través de la variable global un número del 1 al 3.
2. Cuando reciba el número se irá a la pieza correspondiente y la subirá 20cm, luego la volverá a bajar 20cm.
3. Volverá al principio del programa.

PROGRAMA SEGU:

1. Espera a que el interruptor externo se conecte.
2. Escribe un 1 en la variable global de la cola.
3. Espera a que se desconecte el interruptor.
4. A continuación vuelve a la primera línea del programa.

PROGRAMA TERC:

1. El programa pide por pantalla que se le indique un número del 1 al 3.
2. Guarda lo que el usuario escribe en una variable interna.
3. Si esa variable es el 1,2 o 3 se le asignará ese valor a la variable global.
4. Vuelve a la primera línea del programa.

PROGRAMA CUART:

1. Recorre la variable global de la cola.
2. Muestra por pantalla la cola.
3. Espera 5 segundos.
4. Vuelve a la primera línea del programa.

PROGRAMA CEROO:

Se trata del programa principal de este ejercicio, encargado de ejecutar, de forma simultánea los 4 programas anteriores. Ante de eso y para evitar posibles colisiones, colocaremos el brazo robótico en una posición de descanso, en HOME.

El código de los programas será el siguiente:

```
DIMG COLA[10]    //Ponemos nuestra cola como variable global para trabajar con ella
                   desde diferentes programas.
DEFP P1AU          //Definimos una posición auxiliar pero no le grabamos nada


        PROGRAM   PRI
          **********************
DEFINE I           //Definimos un contado
DEFINE NUM      //Definimos una variable para que guarde el número asignado

FOR I=1 TO 10    //Recorremos el vector
SET COLA[I]=0    //Lo inicializamos a 0
ENDFOR

LABEL 0          //Marcamos aquí el retorno de código

QPEND NUM TO COLA  //Esperamos que se envíe un nuevo valor a cola, este tiene que
                     estar entre 1 y 3.
IF NUM=1                  //Si es 1 significa que queremos coger la pieza 1
SETP P1AU=POS6       //Le asignamos el mismo valor de posición que POS6, que es
                     dónde se encuentra la pieza 1.
ENDIF

IF NUM=2                //Hacemos lo mismo para las otras posiciones
SETP P1AU=POS8
ENDIF

IF NUM=3
SETP P1AU=POS4
ENDIF

SHIFTC P1AU BY Z 100      //Para movernos a una posición de aproximación
SPEED 25                      //Indicamos velocidad rápida
```

```
MOVED POS1A            //Nos movemos ahora a la pos de aproximación
OPEN                                //Abrimos la garra
SPEED 10                //Ponemos velocidad lenta para evitar colisiones
SHIFTC P1AU BY Z -100 //Sobrescribimos P1AU para que ahora esté en la pos inicial
MOVED P1AU                 //Nos movemos hacia la pieza seleccionada previamente
CLOSE                    //Cogemos la pieza seleccionada previamente
SPEED 25                  //Nos volvemos a poner en velocidad rápida
SHIFTC P1AU BY Z 200    //Sobrescribimos la posición para que la marque a 20cm por
                           encima
MOVED P1AU              //Nos movemos con la pieza hasta esta posición
SHIFTC P1AU BY Z -200   //Volvemos a sobrescribir la posición auxiliar para que esté en
                          su posición inicial y poder colocar la pieza, aquí lo haremos
                          directamente, sin posición de aproximación ya que el ejercicio
                          pide que levantes la pieza 20cm y la vuelvas a bajar, sin
                           interrupciones ni cambio de velocidades
MOVED P1AU              //Nos movemos a la posición inicial de la pieza a una velocidad
                           rápida
OPEN                    //Abrimos la garra para soltar la pieza
SPEED 10                //Reducimos la velocidad
SHIFTC P1AU BY Z 100    //Sobrescribimos la posición auxiliar para que marque la
                          posición de aproximación
MOVED P1AU              //Nos movemos a la posición de aproximación
SPEED 25                //Aumentamos la velocidad
CLOSE                   //Cerramos la garra
MOVED 0                 //Nos movemos a una posición de descanso, en este caso
                           hemos optado por la posición HOME
GOTO 0                  //Saltamos a la línea de LABEL 0 dentro de este mismo
                          programa
   END
```

Nota: Al final del programa no será necesario resetear la posición de nuestra variable auxiliar POS1AU ya que cuando se produce el salto en LABEL 0, dependiendo del número siguiente de la cola, se le asignará el valor de POS6, POS8 o POS4.
También podríamos haber evitado el uso de tantos SHIFTC ya que de ejercicios anteriores tenemos grabadas la posición de aproximación de las 3 piezas (POS5, POS7 y POS3), solo tendríamos que añadirlo en los IF del principio y asignarle esos valores a otra variable auxiliar (por ejemplo, P2AU).

```
        PROGRAM   SEGU
          **********************
LABEL 1                 //Aquí marcamos nuestro retorno de código
WAIT IN[1]=1            //Esperamos que se encienda el interruptor externo
QPOST 1 TO COLA  //Cuando esto ocurra escribimos un 1 en la variable global COLA
WAIT IN[1]=0            //Volvemos a esperar a que se desconecte
GOTO 1                  //Saltamos a la línea del código dónde está LABEL 1

  END
```

Nota: Para la realización de la escritura del ‘1’ en la cola también se podría utilizar un *TRIGGER* pero es más eficiente el uso de un *WAIT*.

```
        PROGRAM   TERC
          **********************
DEFINE ESCR            //Definimos una variable interna

LABEL 2                //Aquí marcamos el retorno de código

READ “Introduzca el número de la pieza que desea (1, 2 y 3)” ESCR    //Utilizamos este
                              comando para mostrar por pantalla lo que queremos que se nos
                                devuelva, este valor se guardará en la variable ESCR
IF ESCR>0                  //Si dicha variable es mayor que cero y menor que cero
                             procederá al envío del valor de esta a la variable global COLA
   ANDIF ESCR<4             si no se cumple no hará nada, saltará arriba y esperará
QPOST ESCR TO COLA          a que se introduzca un número que lo cumpla
ENDIF

GOTO 2                     //Saltamos a la línea de LABEL 2

   END
```

```
        PROGRAM   CUART
          **********************
DEFINE CONT            //Definimos la variable interna de contador

LABEL 3                  //Aquí marcamos el retorno de código

PRINTLN “Mostramos la cola”      //Esto lo mostraremos por pantalla

FOR CONT=1 TO 10                 //Recorremos la cola

  IF COLA[CONT] <>0              //Vemos si el valor es distinto de cero
      PRINT COLA[CONT]           //Si lo es, mostraremos por pantalla sus componentes
  ENDIF                         //Salimos del IF

ENDFOR                           //Salimos del FOR

DELAY 500                       //Esperamos 5 segundos

GOTO 3                         //Saltamos a la línea de código LABEL 3

   END
```

```
        PROGRAM   CUART
          **********************
SPEED 25        //Indicamos la velocidad rápida
MOVED 0         //Nos movemos a una posición de descanso, la del HOME
CLOSE       //Cerramos la garra por si se hubiese quedad abierta de programas anteriores

RUN PRI         //Ejecutamos simultáneamente los programas
RUN SEGU
RUN TERC
RUN CUART

    END
```