El proyecto final de la asignatura
Diseño con Microcontroladores.

Equipo de robots-brazos controlados por el microcontrolador Renesas M16C

Diseñado por Vladislav Rykov

Índice:

1. Descripción del proyecto
2. Componentes
    1. El microcontrolador Renesas M16C
    2. Motores paso-a-paso
    3. Transistores
3. Cableado
4. Código
5. Posibles ampliaciones
    1. Control de alimentación
    2. Cola de eventos
    3. Control de zonas solapadas

1. Descripción del proyecto.

La intención del trabajo actual es diseñar y simular el funcionamiento de dos robots-brazos que actúan sobre una cinta transportadora concurrentemente recogiendo los productos que se mueven por ella y los redirigiéndolos a los contenedores de producción.

El objetivo principal es aplicar los conocimientos adquiridos en la asignatura a un caso práctico y de esa forma resolver un problema del mundo real usando un sistema de computación.

El núcleo del sistema es el microcontrolador Renesas M16C que controla ambos robots-brazos mediante sus puertos configurados como las salidas. Las patillas de varios puertos del microcontrolador están conectadas con los motores paso-a-paso controlando las articulaciones del robot. Los sensores indican que hay un producto el cual hay que recoger y el primer robot libre empieza el movimiento.

En el proyecto se controla solo los robots y los sensores, sin proveer el control de otros elementos del ambiente de producción como la cinta transportadora en sí, puesto que estos elementos tienen que disponer de sus módulos de control.

El escenario principal es el siguiente:

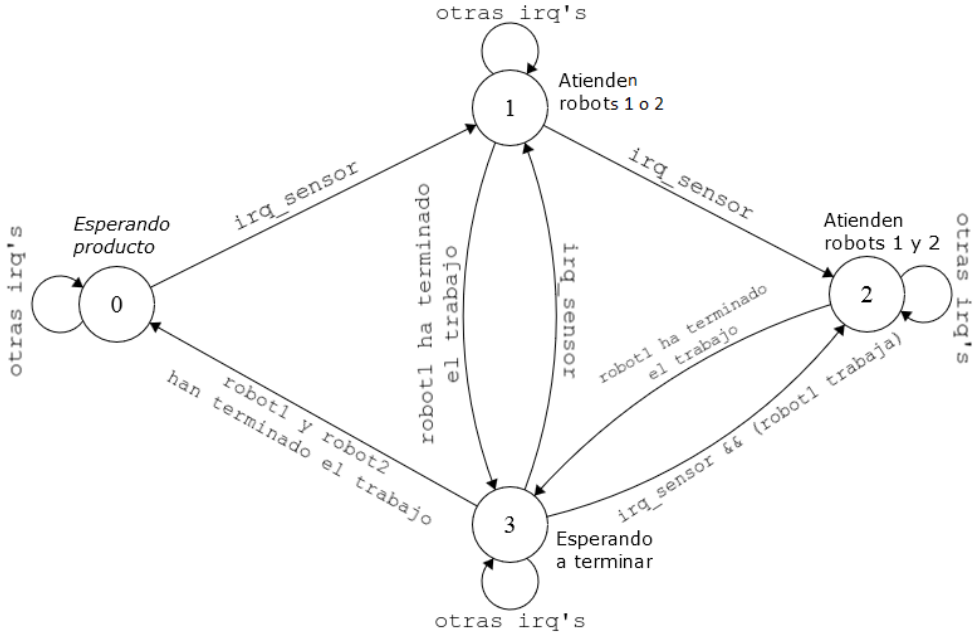
- los robots reposan en el modo de ahorro de energía mientras no hay ningún producto a atender
- viene el producto
- se activa el primer brazo libre a atenderlo
- si no viene otro producto el segundo brazo libre sigue en reposo y mientras el primer brazo efectúa su función
- si viene otro producto y el primer brazo está atendiendo al otro producto entonces se involucra en el trabajo el segundo brazo.
- los brazos terminan el proceso de atención y vuelven al estado de reposo.

Asumimos que la cinta transportadora no puede superar la velocidad del funcionamiento de los dos robots, es decir,

$$el\ período\ de\ llegada\ del\ producto \geq \frac{tiempo\ de\ atención}{2} \quad ,$$

puesto que hay dos robots.

De una forma más clara lo podemos ver expresado con una máquina de estados donde las transiciones marcan distintos eventos.

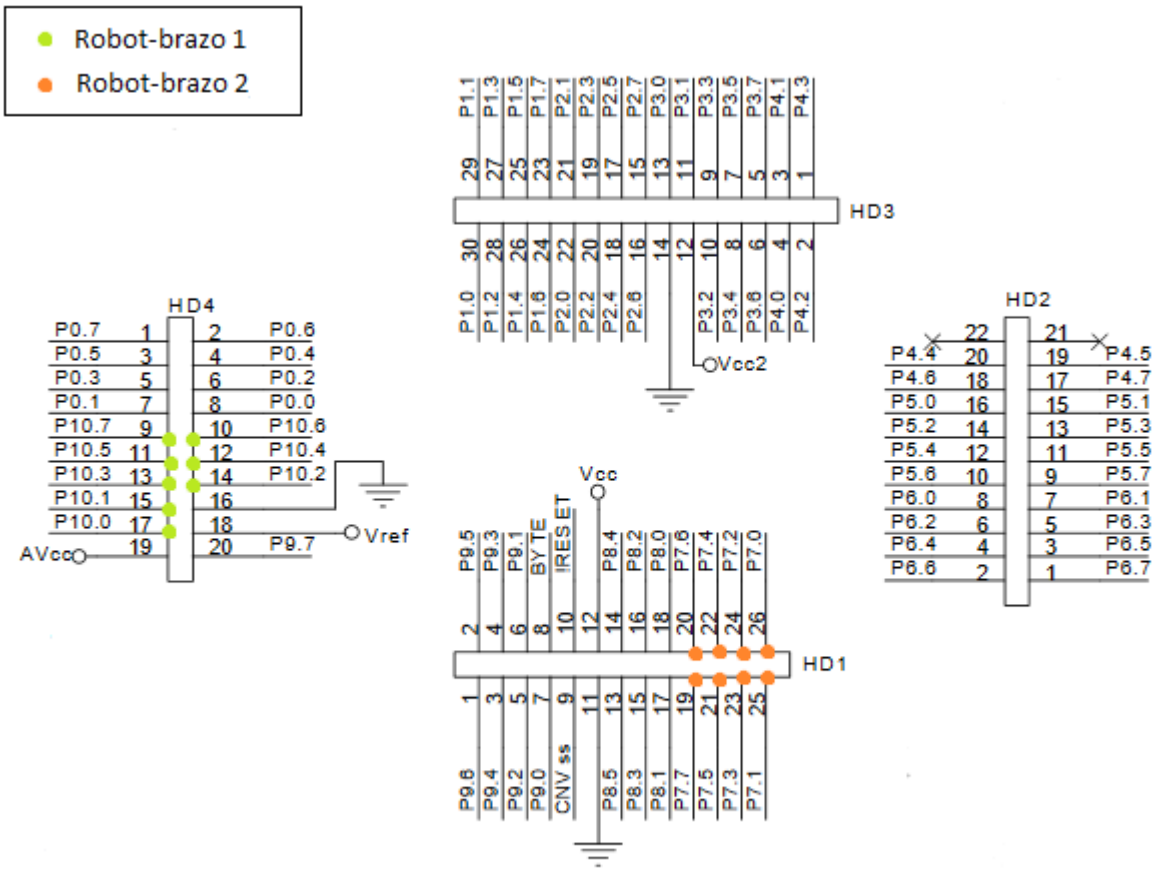


2. Componentes.

1. El microcontrolador Renesas M16C

La parte central del proyecto constituye el microcontrolador con el cual realizamos el trabajo principal mediante manejo de las señales. El microcontrolador tiene 10 puertos GPIO, puertos del uso genérico, los que podemos configurar como más nos convenga.

En el esquema vemos los puertos que se encuentran en disposición y cuales de ellos se utilizan en el proyecto.



Necesitamos 8 pines para cada robot puesto que en la construcción de un robot se utilizan 2 motores, cada uno de los cuales se controla mediante 4 señales. En la sección de los motores lo veremos con más detalles.

De forma resumida podemos ver las características del microcontrolador.

		M16C/60	M16C/62	M16C/62PU
Frecuencia de operación (Mínimo tiempo de ejecución de la instrucción)		10 MHz (100 ns)	16 MHz (62,5 ns)	24 MHz (41,7 ns)
Velocidad de proceso		6 MIPS	8 MIPS	10 MIPS
Número de pines (empaquetado)		100 (Plástico QFP)	100 (Plástico QFP)	100 (Plástico QFP)
No de Instrucciones Básicas		91	91	91
Capacidad Memoria	**ROM**	Externa 64 Kbytes	Flash 256 Kbytes	Flash 384 Kbytes
	RAM	10 Kbytes	20 Kbytes	31 Kbytes
Puertos E/S		11 de 8 bits (87 pines E/S + P8_5 entrada /NMI)	11 de 8 bits (87 pines E/S + P8_5 entrada /NMI)	11 de 8 bits (87 pines E/S + P8_5 entrada /NMI)
Temporizadores (16 bits)		5 de salida TA 3 de entrada TB	5 de salida TA 6 de entrada TB	5 de salida TA 6 de entrada TB
Interrupciones externas		17 internas + 5 externas	25 internas + 8 externas + 4 por software + 7 niveles	29 internas + 8 externas + 4 por software + 7 niveles
Temporizador *Watchdog*		1 de 15 bits con preescalar	1 de 15 bits con preescalar	1 de 15 bits con preescalar

Las características del microcontrolador que se emplea se encuentran en la última columna. Los puntos más interesantes (manejados en el proyecto) son la frecuencia del reloj, puertos de entrada/salida, temporizadores e interrupciones.

Se dispone de 5 temporizadores internos (TA) los que cuentan en función de la frecuencia del reloj. Para tener más margen de temporización se ofrece la división de la frecuencia principal entre diferentes escalares (2, 4, 8, 16 y32). En el proyecto se utilizan 4 temporizadores TA[0-3] asociadas a las interrupciones internas.

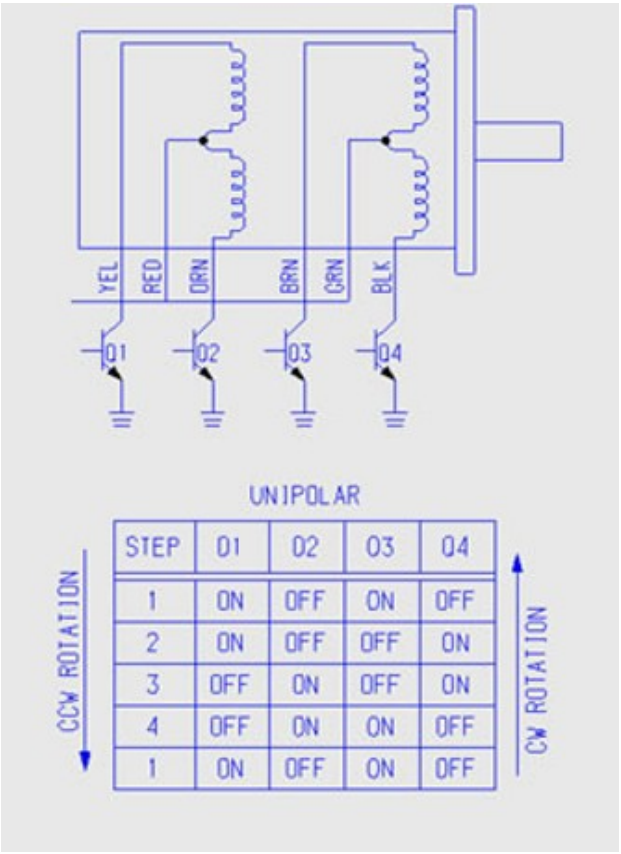
El microcontrolador no necesita mucha capacidad computacional ni tampoco grandes cantidades de memoria. Se tiene 31 Kbytes de RAM de los cuales se utilizan en el trabajo pocos bytes alocados dinámicamente para guardar variables de estados.

Un punto importante también es la posibilidad de tener las interrupciones externas lo que permitirá a reaccionar al microcontrolador ante eventos exteriores. Se utilizan 2 interrupciones externas por botones INT[0-1] que simulan los sensores de la llegada del producto.

Como punto adicional se utilizan los diodos leds, uno para cada robot, como indicadores del estado del funcionamiento.

2. Motor paso-a-paso.

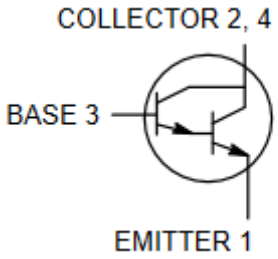
Otro componente del sistema es el motor paso-a-paso Gear Motors 26M type 'V' series 26M048B2U. Es un motor unipolar que permite efectuar revoluciones mediante una serie de pasos. Cada paso se emplea mediante activación de pines según el diagrama que se encuentre en el datasheet del motor.



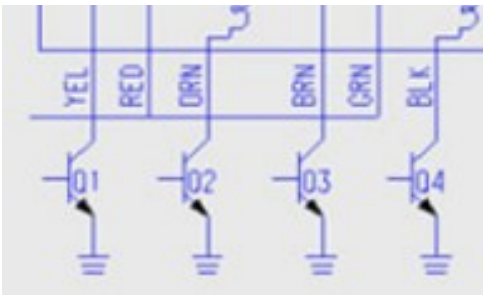
El movimiento se efectúa activando y desactivando secuencialmente las patillas correspondientes a pasos. Si las activamos en el orden creciente, como indica el esquema del datasheet, se efectúa el movimiento de 7.5 grados en sentido contra reloj; y si lo hacemos en el orden decreciente se efectúa el movimiento en sentido de reloj respectivamente.

La frecuencia con la que hacemos cada cambio marca la velocidad que se mide en revoluciones por segundo. En el proyecto se efectúa un movimiento cada interrupción por lo tanto se debe tener en cuenta que el tiempo de atención de una interrupción no tiene que superar el período de la misma. Así que hay un límite de velocidad. Más todavía si están funcionando dos robots simultáneamente tenemos las interrupciones interrumpidas por otras interrupciones lo cual hay que tratar con mucho cuidado.

3. Transistores BD677

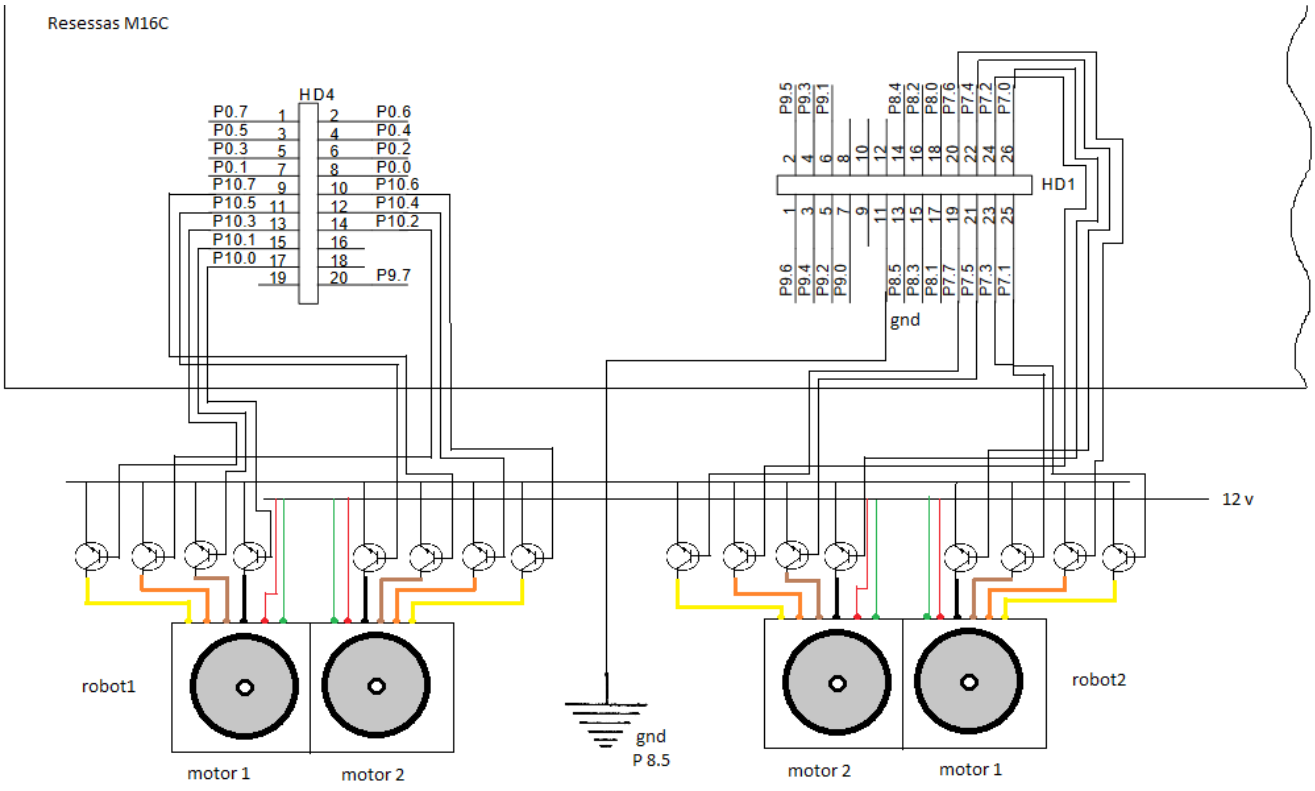


Además para controlar el motor se necesitan transistores de tipo NPN que tiene bajo su control el manejo de la tierra.



Al activar la base del transistor se crea un campo magnético que cierra el dieléctrico creando el contacto.

3. Cableado



4. Código

```
;
*******************************************************************************
;*
;*    DISEÑO DE SISTEMAS BASADOS EN MICROCONTROLADORES. CURSO 2016/2017
;*    PROYECTO FINAL    : final_project_3DKM16C.a30
;*
*******************************************************************************
;
;----- INCLUYE FICHERO SFR -----------------------------------------------
;
      .list off
      .include sfr62s.inc
      .list on
;
;----- DEFINICIÓN DE SÍMBOLOS 3DKM16C ------------------------------------
;
VramTOP           .equ  000400h          ; inicio de la RAM interna
VramEND           .equ  007CFFh          ; final de la RAM interna
VprogTOP          .equ  0A0000h          ; inicio del área de programa
Vvector           .equ  0FFFDCh          ; inicio tabla de interrupciones fija
Vintbase          .equ  0FA000h          ; inicio tabla de interrupciones
                                         ;     variable
VIstack           .equ  007CFFh          ; puntero de pila (SP)
SB_base           .equ  000380h          ; dirección base de SB
;
;----- AREA DE DATOS RAM -------------------------------------------------
;
      .section    memory,data
      .org  VramTOP
; variables guardados en la memoria dinámica
; variables para robot 1
estado_actual_robot1_motor1:        .blkb       1
sentido_actual_robot1_motor1:       .blkb       1
estado_actual_robot1_motor2:        .blkb       1
sentido_actual_robot1_motor2:       .blkb       1
estado_articulacion_robot1:         .blkb       1
; variables para robot 2
estado_actual_robot2_motor1:        .blkb       1
sentido_actual_robot2_motor1:       .blkb       1
estado_actual_robot2_motor2:        .blkb       1
sentido_actual_robot2_motor2:       .blkb       1
estado_articulacion_robot2:         .blkb       1
;
;----- AREA DE PROGRAMA ROM  ---------------------------------------------
      .section    prog,code
      .org  VprogTOP
;
;----- limpieza de la RAM e inicialización de temporizador y puertos  --------
;
reset:
      ; primero limpiamos la memoria RAM
      mov.w #(VramEND+1-VramTOP)/2,R3
      mov.w #VramTOP,A1
      sstr.w

      ; tenemos la tabla de vector de interrupciones del usuario
      ldintb      #Tabla_Vector_Usuario

      ; inicializamos todas variables que vamos a utilizar
      ; variables del robot 1
```

```
mov.b       #0, estado_actual_robot1_motor1    ; variable de estado de
                                               ;     paso del motor 0
mov.b       #0, sentido_actual_robot1_motor1   ; el sentido del motor
                                               ;     (cw)
mov.b       #0, estado_actual_robot1_motor2    ; variable de estado de
                                               ;     paso del motor 1
mov.b       #0, sentido_actual_robot1_motor2   ; el sentido del motor
                                               ;     (cw)
mov.b       #0, estado_articulacion_robot1     ; el estado inicial de la
                                               ;     articulación
; variables del robot 2
mov.b       #0, estado_actual_robot2_motor1    ; ítem
mov.b       #0, sentido_actual_robot2_motor1
mov.b       #0, estado_actual_robot2_motor2
mov.b       #0, sentido_actual_robot2_motor2
mov.b       #0, estado_articulacion_robot2
; variables contadores que controlan la longitud de movimientos
mov.w       #0, r0
mov.w       #0, r1
mov.w       #0, r2
mov.w       #0, r3

; la configuración de los puertos, leds y temporizadores
; configuramos los leds como las salidas
; cada led indica si el robot esta funcionando
bset  pd2_0             ; funcionamiento brazo 1
bset  pd2_2             ; funcionamiento brazo 2

; configuramos los puertos para controlar los motores
; P7_[0-3] como la salida motor brazo 1
bset  pd7_0
bset  pd7_1
bset  pd7_2
bset  pd7_3

; P7_[4-7] como la salida motor mano 1
bset  pd7_4
bset  pd7_5
bset  pd7_6
bset  pd7_7

; P10_[0-3] como la salida motor brazo 2
bset  pd10_0
bset  pd10_1
bset  pd10_2
bset  pd10_3

; P10_[4-7] como la salida motor mano 2
bset  pd10_4
bset  pd10_5
bset  pd10_6
bset  pd10_7

; actualizamos los estados iniciales de los motores
; por defecto todos los bits están a 0, por lo tanto activamos solo los
;     bits necesarios
; estado inicial de los motores brazo 1
bnot  p7_0
bnot  p7_3


; estado inicial de los motores mano 1
bnot  p7_4
```

```
      bnot  p7_7

      ; estado inicial de los motores brazo 2
      bnot  p10_0
      bnot  p10_3

      ; estado inicial de los motores mano 2
      bnot  p10_4
      bnot  p10_7

      ; configuramos los timers que controlan el movimiento de los motores
      ; ta0 controla brazo 1
      mov.b #40h,ta0mr        ; TA0 en modo contador con f32
      mov.b #1,ta0ic          ; Limpia el bit de petición de interrupción.
                              ;     Nivel de prioridad 1
    mov.w   #8000,ta0         ; 8000 periodos de 2 us = 16 ms. (el reloj es de
                              ;     36MHz/32)

      ; ta1 controla mano 1
      mov.b       #0, ta1mr   ; TA1 en modo contador con f2
      mov.b       #1, ta1ic   ; Limpia el bit de petición de interrupción.
                              ;     Nivel de prioridad 1
      mov.w #8000,ta1         ; 8000 periodos de 125 ns = 1 ms. (el reloj
                              ;     es de 36MHz/2)

      ; ta2 controla brazo 2
      mov.b #40h,ta2mr        ; TA2 en modo contador con f32
      mov.b #1,ta2ic          ; Limpia el bit de petición de interrupción. Nivel
                              ;     de prioridad 1
    mov.w   #8000,ta2         ; 8000 periodos de 2 us = 16 ms. (el reloj es de
                              ;     36MHz/32)

      ; ta3 controla mano 2
      mov.b       #0, ta3mr   ; TA3 en modo contador con f2
      mov.b       #1, ta3ic   ; Limpia el bit de petición de interrupción.
                              ;     Nivel de prioridad 1
      mov.w #8000,ta3         ; 8000 periodos de 125 ns = 1 ms. (el reloj es de
                              ;     36MHz/2)

      ; solo configuramos los contadores porque el manejo va a través de
      ;     interrupciones.
      ; a todas interrupciones asignamos igual nivel de prioridad ya que no
      ;     tenemos las preferencias en este caso.

      ; para simular los sensores configuramos los botones como origines de
      ;     interrupciones
      ;     escribiendo en sus registros de control.
      mov.b       #01h, int0ic      ; interrupción por el botón int0
      mov.b       #01h, int1ic      ; interrupción por el botón int1
;
;
;----- programa principal ----------------------------------------------------
;
;
; todos el funcionamiento del sistema se produce a través de eventos que ;
      generan las interrupciones para poder entrar siempre si es posible en ;
      el modo de ahorro de energía
;
main:
      jmp   main
;
;
;
```

```
;----- manejadores del robot brazo 1 ----------------------------------------
;----- manejador de la rutina de interrupción del TA0 ------------------------
;
; comentare detalladamente una interrupción y el resto con menos detalles
;     porque cada interrupción asignada a un temporizador es prácticamente
;     idéntica.
sw_ta0:
      ; primero y antes de todo controlamos la longitud de articulación que
      ;     depende del
      ;     tiempo y de la variable (r0 en este caso) la que cuenta el numero de
      ;     los movimientos.
      ; lamentablemente no tuve suficiente tiempo para averiguar cuantos pasos
      ;     hay que hacer para girar una revolución. las medidas por lo tanto
      ;     son aproximadas.
      add.w #1, r0            ; sumamos uno al contador
      cmp.w #6400, r0         ; comprobamos si ha llegado al punto correcto
      jnz fol1
      ; si ha llegado al punto
      add.b #1, estado_articulacion_robot1     ; cambiamos el estado
      xor.b #1, sentido_actual_robot1_motor1   ; cambiamos el sentido del motor
      mov.w #0, r0                             ; reseteamos el contador
      bnot  ta0s                               ; parar el brazo mientras
                                               ;     trabaja fijador
      bset  ta1s        ; ha movido el brazo -> mandar a sujetar el producto
      jmp salir_de_sw_ta0                      ; salir de la interrupción
; si no ha llegado al punto
fol1:
      ; primero comprobamos cual es el sentido del motor
      cmp.b sentido_actual_robot1_motor1, 0
      jz cw_r1_b1
      jmp ccw_r1_b1

; si es contra reloj
ccw_r1_b1:
      ; cambiar de un estado a otro hay que identificar en que estado estamos
      ; y luego pasar a través de la etiqueta para hacer siguiente transición
      cmp.b estado_actual_robot1_motor1, 0          ; 3-0-1
      jnz paso_01_r1_b1
      cmp.b estado_actual_robot1_motor1, 1          ; 0-1-2
      jnz paso_12_r1_b1
      cmp.b estado_actual_robot1_motor1, 2          ; 1-2-3
      jnz paso_23_r1_b1
      cmp.b estado_actual_robot1_motor1, 3          ; 2-3-0
      jnz paso_30_r1_b1

; los cambios de pasos se consiguen cambiando estados de las patillas
paso_01_r1_b1:
      bnot p7_2
      bnot p7_3
      add.b #1, estado_actual_robot1_motor1    ; cambio de estado
      jmp salir_de_sw_ta0                      ; salir de la interrupción
; el resto se maneja idénticamente
paso_12_r1_b1:
      bnot p7_0
      bnot p7_1
      add.b #1, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

paso_23_r1_b1:
      bnot p7_2
      bnot p7_3
      add.b #1, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0
```

```
paso_30_r1_b1:
      bnot p7_0
      bnot p7_1
      mov.b #0, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

; si es en sentido de reloj
cw_r1_b1:
      cmp.b estado_actual_robot1_motor1, 0          ; 3-0-1
      jnz paso_10_r1_b1
      cmp.b estado_actual_robot1_motor1, 1          ; 0-1-2
      jnz paso_21_r1_b1
      cmp.b estado_actual_robot1_motor1, 2          ; 1-2-3
      jnz paso_32_r1_b1
      cmp.b estado_actual_robot1_motor1, 3          ; 2-3-0
      jnz paso_03_r1_b1

paso_10_r1_b1:
      bnot p7_0
      bnot p7_1
      add.b #1, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

paso_21_r1_b1:
      bnot p7_2
      bnot p7_3
      add.b #1, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

paso_32_r1_b1:
      bnot p7_0
      bnot p7_1
      add.b #1, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

paso_03_r1_b1:
      bnot p7_2
      bnot p7_3
      mov.b #0, estado_actual_robot1_motor1
      jmp salir_de_sw_ta0

salir_de_sw_ta0:  reit
;
;----- manejador de la rutina de interrupción del TA1 ------------------------
;
sw_ta1:
      cmp.b sentido_actual_robot1_motor2, 0
      jz cw_r1_b2
      jmp ccw_r1_b2

ccw_r1_b2:
      cmp.b estado_actual_robot1_motor2, 0          ; 3-0-1
      jnz paso_01_r1_b2
      cmp.b estado_actual_robot1_motor2, 1          ; 0-1-2
      jnz paso_12_r1_b2
      cmp.b estado_actual_robot1_motor2, 2          ; 1-2-3
      jnz paso_23_r1_b2
      cmp.b estado_actual_robot1_motor2, 3          ; 2-3-0
      jnz paso_30_r1_b2

paso_01_r1_b2:
      bnot p7_6
```

```
      bnot p7_7
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_12_r1_b2:
      bnot p7_4
      bnot p7_5
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_23_r1_b2:
      bnot p7_6
      bnot p7_7
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_30_r1_b2:
      bnot p7_4
      bnot p7_5
      mov.b #0, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

cw_r1_b2:
      cmp.b estado_actual_robot1_motor2, 0          ; 3-0-1
      jnz paso_10_r1_b2
      cmp.b estado_actual_robot1_motor2, 1          ; 0-1-2
      jnz paso_21_r1_b2
      cmp.b estado_actual_robot1_motor2, 2          ; 1-2-3
      jnz paso_32_r1_b2
      cmp.b estado_actual_robot1_motor2, 3          ; 2-3-0
      jnz paso_03_r1_b2

paso_10_r1_b2:
      bnot p7_4
      bnot p7_5
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_21_r1_b2:
      bnot p7_6
      bnot p7_7
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_32_r1_b2:
      bnot p7_4
      bnot p7_5
      add.b #1, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

paso_03_r1_b2:
      bnot p7_6
      bnot p7_7
      mov.b #0, estado_actual_robot1_motor2
      jmp salir_de_sw_ta1

; la articulación de la mano de un robot es especial porque con ella termina
;     el movimiento de la atención del producto.
salir_de_sw_ta1:
      add.w #1, r1            ; sumamos el contador
      cmp.w #6400, r1         ; comprobamos si ha llegado al punto
      jnz fol_reit_1
      ; si la llegado al punto
```

```
      cmp.b #2, estado_articulacion_robot1     ; comprobamos si es el
                                               ;     movimiento final
      jz final_articulacion_robot1
      ; si no es el movimiento final
      xor.b #1, sentido_actual_robot1_motor2   ; cambiamos el sentido del motor
      mov.w #0, r1                             ; reseteamos el temporizador
      bnot  ta1s                               ; paramos el fijador
      bset  ta0s                               ; mandamos a girar el brazo
fol_reit_1: reit

; si es el movimiento final, se ha atendido el producto
final_articulacion_robot1:
            xor.b #1, sentido_actual_robot1_motor2   ; cambiamos el sentido del
                                                     ;     motor
            mov.w #0, r1                             ; reseteamos el temporizador
            mov.b #0, estado_articulacion_robot1     ; volvemos al estado
                                                     ;     inicial del robot
            bnot  ta1s                               ; apagamos el temporizador
            bnot  p2_0       ; desactivamos el indicador del funcionamiento
            jmp   fol_reit_1                                 ; y salimos
;
;----- manejadores del robot brazo 2 ----------------------------------------
;----- manejador de la rutina de interrupción del TA2 -----------------------
;
sw_ta2:
      add.w #1, r2
      cmp.w #6400, r2
      jnz continuar_r2_b1
      add.b #1, estado_articulacion_robot2
      xor.b #1, sentido_actual_robot2_motor1
      mov.w #0, r2
      bnot  ta2s  ; parar el brazo mientras trabaja fijador
      bset  ta3s  ; ha terminado mover el brazo -> mandar a sujetar el producto
      jmp salir_de_sw_ta2
continuar_r2_b1:
      cmp.b sentido_actual_robot2_motor1, 0
      jz    cw_r2_b1
      jmp ccw_r2_b1

ccw_r2_b1:
      cmp.b estado_actual_robot2_motor1, 0           ; 3-0-1
      jnz paso_01_r2_b1
      cmp.b estado_actual_robot2_motor1, 1           ; 0-1-2
      jnz paso_12_r2_b1
      cmp.b estado_actual_robot2_motor1, 2           ; 1-2-3
      jnz paso_23_r2_b1
      cmp.b estado_actual_robot2_motor1, 3           ; 2-3-0
      jnz paso_30_r2_b1

paso_01_r2_b1:
      bnot p10_2
      bnot p10_3
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_12_r2_b1:
      bnot p10_0
      bnot p10_1
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_23_r2_b1:
      bnot p10_2
```

```
      bnot p10_3
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_30_r2_b1:
      bnot p10_0
      bnot p10_1
      mov.b #0, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

cw_r2_b1:
      cmp.b estado_actual_robot2_motor1, 0          ; 3-0-1
      jnz paso_10_r2_b1
      cmp.b estado_actual_robot2_motor1, 1          ; 0-1-2
      jnz paso_21_r2_b1
      cmp.b estado_actual_robot2_motor1, 2          ; 1-2-3
      jnz paso_32_r2_b1
      cmp.b estado_actual_robot2_motor1, 3          ; 2-3-0
      jnz paso_03_r2_b1

paso_10_r2_b1:
      bnot p10_0
      bnot p10_1
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_21_r2_b1:
      bnot p10_2
      bnot p10_3
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_32_r2_b1:
      bnot p10_0
      bnot p10_1
      add.b #1, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

paso_03_r2_b1:
      bnot p10_2
      bnot p10_3
      mov.b #0, estado_actual_robot2_motor1
      jmp salir_de_sw_ta2

salir_de_sw_ta2:  reit

;----- manejador de la rutina de interrupción del TA3 ------------------------
;
sw_ta3:
      cmp.b sentido_actual_robot2_motor2, 0
      jz cw_r2_b2
      jmp ccw_r2_b2

ccw_r2_b2:
      cmp.b estado_actual_robot2_motor2, 0          ; 3-0-1
      jnz paso_01_r2_b2
      cmp.b estado_actual_robot2_motor2, 1          ; 0-1-2
      jnz paso_12_r2_b2
      cmp.b estado_actual_robot2_motor2, 2          ; 1-2-3
      jnz paso_23_r2_b2
      cmp.b estado_actual_robot2_motor2, 3          ; 2-3-0
      jnz paso_30_r2_b2
```

```
paso_01_r2_b2:
      bnot p10_6
      bnot p10_7
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

paso_12_r2_b2:
      bnot p10_4
      bnot p10_5
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

paso_23_r2_b2:
      bnot p10_6
      bnot p10_7
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

paso_30_r2_b2:
      bnot p10_4
      bnot p10_5
      mov.b #0, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

cw_r2_b2:
      cmp.b estado_actual_robot2_motor2, 0          ; 3-0-1
      jnz paso_10_r2_b2
      cmp.b estado_actual_robot2_motor2, 1          ; 0-1-2
      jnz paso_21_r2_b2
      cmp.b estado_actual_robot2_motor2, 2          ; 1-2-3
      jnz paso_32_r2_b2
      cmp.b estado_actual_robot2_motor2, 3          ; 2-3-0
      jnz paso_03_r2_b2

paso_10_r2_b2:
      bnot p10_4
      bnot p10_5
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

paso_21_r2_b2:
      bnot p10_6
      bnot p10_7
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

paso_32_r2_b2:
      bnot p10_4
      bnot p10_5
      add.b #1, estado_actual_robot2_motor2
      jmp salir_de_sw_ta1

paso_03_r2_b2:
      bnot p10_6
      bnot p10_7
      mov.b #0, estado_actual_robot2_motor2
      jmp salir_de_sw_ta3

salir_de_sw_ta3:
            add.w #1, r3
            cmp.w #6400, r3
            jnz fol_reit_2
            cmp.b #2, estado_articulacion_robot2
```

```
            jz final_articulacion_robot2
            xor.b #1, sentido_actual_robot2_motor2
            mov.w #0, r3
            bnot  ta3s  ; parar el fijador
            bset  ta2s  ; mandar a girar el brazo
fol_reit_2: reit

final_articulacion_robot2:
            xor.b #1, sentido_actual_robot2_motor2
            mov.w #0, r3
            mov.b #0, estado_articulacion_robot2
            bnot  ta3s
            bnot  p2_2
            jmp   fol_reit_2
;
;----- manejador de la rutina de interrupción del INT0 -----------------------
;
; para simular el sensor utilizamos la interrupción generada por el botón
sw_int0:
      ; para tolerar los rebotes aprovecho el indicador del funcionamiento
      ; si el brazo ya esta funcionando, ignorar la interrupción
      btst  p2_0
      jnz salir_de_sw_int0
      ; si el brazo no esta involucrado al trabajo
      bnot  p2_0              ; se activa el indicador de funcionamiento
      bset  ta0s              ; y se arranca el contador
salir_de_sw_int0: reit
;
;----- manejador de la rutina de interrupción del INT1 -----------------------
;
; la segunda interrupción es idéntica a la anterior
sw_int1:
      btst  p2_2
      jnz salir_de_sw_int1
      bnot  p2_2
      bset  ta2s
salir_de_sw_int1: reit
;
;----- manejador de la rutina dummy ------------------------------------------
;
dummy:
      reit
;
;----- (tabla de vectores variable) ------------------------------------------
; en el vector de interrupciones marcamos las entradas que utilizamos:
;     los timers y las int0 e int1
      .section    int_ta0,int_ta1,int_ta2,int_ta3,int_int0,int_int1,romdata
      .org        Vintbase
Tabla_Vector_Usuario:
      .lword            dummy       ; No0  Break Interrupt
      .lword            dummy       ; No1  Break Interrupt
      .lword            dummy       ; No2  Break Interrupt
      .lword            dummy       ; No3  Break Interrupt
      .lword            dummy       ; No4  Break Interrupt
      .lword            dummy       ; No5  Break Interrupt
      .lword            dummy       ; No6  Break Interrupt
      .lword            dummy       ; No7  Break Interrupt
      .lword            dummy       ; No8  Break Interrupt
      .lword            dummy       ; No9  Break Interrupt
      .lword            dummy       ; No10 Bus Clash Detect
      .lword            dummy       ; No11 DMA0
      .lword            dummy       ; No12 DMA1
      .lword            dummy       ; No13 KEY IN Interrupt
```

```
      .lword            dummy       ; No14 A-D Interrupt
      .lword            dummy       ; No15 UART2 Transmission Interrupt
      .lword            dummy       ; No16 UART2 receive Interrupt
      .lword            dummy       ; No17 UART0 Transmission Interrupt
      .lword            dummy       ; No18 UART0 receive Interrupt
      .lword            0FF900H     ; No19 UART1 Transmission Interrupt
      .lword            0FF900H     ; No20 UART1 receive Interrupt
      .lword            sw_ta0      ; No21 TimerA0 Interrupt
      .lword            sw_ta1  ; No22 TimerA1 Interrupt
      .lword            sw_ta2      ; No23 TimerA2 Interrupt
      .lword            sw_ta3      ; No24 TimerA3 Interrupt
      .lword            dummy       ; No25 TimerA4 Interrupt
      .lword            dummy       ; No26 TimerB0 Interrupt
      .lword            dummy       ; No27 TimerB1 Interrupt
      .lword            dummy       ; No28 TimerB2 Interrupt
      .lword            sw_int0     ; No29 INIT0(Active Low) Interrupt
      .lword            sw_int1     ; No30 INIT1(Active Low) Interrupt
      .lword            dummy       ; No31 INIT2(Active Low) Interrupt
      .lword            dummy       ; No32 S/W Interrupt
      .lword            dummy       ; No33 S/W Interrupt
      .lword            dummy       ; No34 S/W Interrupt
      .lword            dummy       ; No35 S/W Interrupt
      .lword            dummy       ; No36 S/W Interrupt
      .lword            dummy       ; No37 S/W Interrupt
      .lword            dummy       ; No38 S/W Interrupt
      .lword            dummy       ; No39 S/W Interrupt
      .lword            dummy       ; No40 S/W Interrupt
      .lword            dummy       ; No41 S/W Interrupt
      .lword            dummy       ; No42 S/W Interrupt
      .lword            dummy       ; No43 S/W Interrupt
      .lword            dummy       ; No44 S/W Interrupt
      .lword            dummy       ; No45 S/W Interrupt
      .lword            dummy       ; No46 S/W Interrupt
      .lword            dummy       ; No47 S/W Interrupt
      .lword            dummy       ; No48 S/W Interrupt
      .lword            dummy       ; No49 S/W Interrupt
      .lword            dummy       ; No50 S/W Interrupt
      .lword            dummy       ; No51 S/W Interrupt
      .lword            dummy       ; No52 S/W Interrupt
      .lword            dummy       ; No53 S/W Interrupt
      .lword            dummy       ; No54 S/W Interrupt
      .lword            dummy       ; No55 S/W Interrupt
      .lword            dummy       ; No56 S/W Interrupt
      .lword            dummy       ; No57 S/W Interrupt
      .lword            dummy       ; No58 S/W Interrupt
      .lword            dummy       ; No59 S/W Interrupt
      .lword            dummy       ; No60 S/W Interrupt
      .lword            dummy       ; No61 S/W Interrupt
      .lword            dummy       ; No62 S/W Interrupt
      .lword            dummy       ; No63 S/W Interrupt
;
;----- interrupci¢n de reset (tabla de vectores fija) ------------------------
;
      .section    int_reset,romdata
      .org        Vvector+(8*4)
      .lword            reset
;
;----- program end ----------------------------------------------------------
;
      .end
;
```

5. Posibles ampliaciones

1. Control de alimentación.

Una posible modificación, que fue conseguida en el proyecto en práctica, del sistema sería el control de alimentación. En un sistema industrial tiene mucha importancia prever casos de una falta de alimentación y avisar al usuario o el controlador del sistema en qué estado se encuentra.

La dicha funcionalidad se consigue involucrando en el trabajo un conversor analógico-digital. Dicho conversor está periódicamente muestreando la fuente de alimentación y mediante una interrupción corrige el valor actual de la alimentación. Comparando este valor con un threashold. En el caso de falta de alimentación se detiene el sistema indicando por un led que falta la alimentación.

2. Cola de eventos.

Otra ampliación del proyecto sería prever la situación cuando todos los robots están funcionando y llega un producto más así formando una cola. Esta modificación sería posible con sensores adicionales o con sensores que pueden detectar el caso.

Se resuelve con modificando el uso de la variable que controla una articulación. En el proyecto este numero es constante pero si lo modificamos un robot realizará varias articulaciones seguidas.

3. Control de zonas solapadas.

Los dos robots-brazos tienen una zona en común que podemos comparar con un recurso compartido. Esta ampliación se puede conseguir cunado se tiene un control del movimiento de los motores con precisión. Harían falta también los sensores adicionales para saber si un robot está en la zona solapada o no. También teniendo precisión se podría saberlo utilizando contadores que controlan longitudes de movimiento.