```
; Robot Arm Sequencing Program.
;
; Movements Driven from Data Table.
; Supports concurrent movements
;
;
;      Version           1.06
;      last Modified:    3rd Oct 2004
;      Modified By:Andrew Wells
;
;      Copyright:        Andrew Wells, 2004
;
;      This source code can be freely copied and modified, but the following comment lines must
;      be retained
;
;#########################################################
;# Modified from Original MeccCode II                    #
;# Source supplied by Meccanisms Limited                 #
;# P O Box 26-179, Epsom, Auckland 1003m New Zealand     #
;# www.meccanisms.com                                    #
;# support@meccanisms.com                                #
;# MeccCode II Copyright (c) 2004 Meccanisms Limited     #
;#########################################################

;#############################################################################################################
;
;      Global Variable Declarations
;
;#############################################################################################################


; GetMovement() returns one of the following Sequence types
; the value of the types is not important, only that they are unique
Declare Constant   MoveSeq      = 0    ; normal movement
Declare Constant   ExitSeq      = 1    ; exit the sequence
Declare Constant   EndSeq= 2    ; end of a group of movements
Declare Constant   UnknownSeq   = 3    ;

Declare      Byte  MA_Dir       ; ARM - current direction for this movement ("F" or "B" to suit
SetMotor)
Declare      Byte  MB_Dir       ; Base
Declare      Byte  MS_Dir       ; Shoulder
Declare      Byte  MW_Dir       ; Wrist

Declare      Byte  MA_PulseCount      ; ARM - number of pulses counted for this movement
Declare      Byte  MB_PulseCount      ; Base
Declare      Byte  MS_PulseCount      ; Shoulder
Declare      Byte  MW_PulseCount      ; Wrist

Declare      Byte  MA_Distance        ; ARM - number of pulses required to complete this movement
Declare      Byte  MB_Distance        ; Base
Declare      Byte  MS_Distance        ; Shoulder
Declare      Byte  MW_Distance        ; Wrist


Declare      Byte  MA_Speed           ; Arm - current speed during this movement
Declare      Byte  MB_Speed           ; Base
Declare      Byte  MS_Speed           ; Shoulder
Declare      Byte  MW_Speed           ; Wrist

Declare      Byte  MA_Decel           ; Decelleration Zone counter
Declare      Byte  MB_Decel
Declare      Byte  MS_Decel
Declare      Byte  MW_Decel

Declare      Byte  MA_Accel           ; Acceleration Zone counter
Declare      Byte  MB_Accel
Declare      Byte  MS_Accel
Declare      Byte  MW_Accel

                                      ; These constants determine how quickly the movement accelerates
or decellerates
Declare      Byte  MA_Decel_Delta = 1 ; Decelleration Zone counter
Declare      Byte  MB_Decel_Delta = 1
Declare      Byte  MS_Decel_Delta = 1
Declare      Byte  MW_Decel_Delta = 1
```

```
Declare     Byte  MA_Accel_Delta = 1       ; Acceleration Zone counter
Declare     Byte  MB_Accel_Delta = 1
Declare     Byte  MS_Accel_Delta = 1
Declare     Byte  MW_Accel_Delta = 1


; Moving flags used to set up a movement so that the Event can handle to specific movement
Declare Boolean         ArmMoving    = 0
Declare Boolean         BaseMoving   = 0
Declare Boolean         ShoulderMoving     = 0
Declare Boolean         WristMoving = 0

Declare DataPointer     MovementPtr        ; points to the next movement from the Movement Table

Declare Byte      Sequence          ; used to hold the result of the latest GetMovement() Call
Declare Byte      Blockend

Declare Word      BounceTime   = 10   ; time to allow any bouncing to settle




;#############################################################################################
; Motor Characteristics and speeds
;
; For each movement, there is a max and min speed.
; There are different speeds for the two directions, to allow tuning for imbalances of weights,
different characteristics of motors etc
;
Declare Byte      MB_MinSpeed
Declare Byte      MB_MaxSpeed
Declare Byte      MBL_MinSpeed =15
Declare Byte      MBL_MaxSpeed =25
Declare Byte      MBR_MinSpeed =14
Declare Byte      MBR_MaxSpeed =25
Declare Byte      Base_Stall_Speed = 16

Declare Byte      MA_MinSpeed
Declare Byte      MA_MaxSpeed
Declare Byte      MAU_MinSpeed =14
Declare Byte      MAU_MaxSpeed =35
Declare Byte      MAD_MinSpeed =12
Declare Byte      MAD_MaxSpeed =24
Declare Byte      Arm_Stall_speed = 017

Declare Byte      MS_MinSpeed
Declare Byte      MS_MaxSpeed
Declare Byte      MSO_MinSpeed =15
Declare Byte      MSO_MaxSpeed =29
Declare Byte      MSI_MinSpeed =13
Declare Byte      MSI_MaxSpeed =23
Declare Byte      Shoulder_Stall_speed =   20

Declare Byte      MW_MinSpeed
Declare Byte      MW_MaxSpeed
Declare Byte      MWI_MinSpeed =13
Declare Byte      MWI_MaxSpeed =35
Declare Byte      MWO_MinSpeed =13
Declare Byte      MWO_MaxSpeed =35
Declare Byte      wrist_Stall_speed =      25

Declare WordStall_Delay=100   ; how long to wait between checking pulses, when waiting for a stall

; Constants to match which hardware ports are used for which Switch inputs
Declare Constant  base_input  =2
Declare Constant  arm_input         =1
Declare ByteArm_top_Switch
Declare ByteBase_Switch

; Microswitch status checks
Declare Constant  Closed            =     1
Declare Constant  Open        =     0

; Constants to match which motor ports are used for which movements
Declare Constant  ArmMotor          =2
```

```
Declare Constant   BaseMotor          =1
Declare Constant   WristMotor        =3
Declare Constant   ShoulderMotor     =4



; define which direction of SetMotor matches which logical direction
Declare Constant   ArmDown      = "F"
Declare Constant   ArmUp        = "B"
Declare Constant   BaseLeft     = "F"
Declare Constant   BaseRight    = "B"
Declare Constant   ShoulderIn   = "B"
Declare Constant   ShoulderOut  = "F"
Declare Constant   WristIn      = "B"
Declare Constant   WristOut     = "F"

Declare DataTable  HomeTune =   "16A6,P,P,P,P<"
End DataTable

;#########################################################################


Begin Program


Call Setup()                        ; do set up and initialisations


Sequence = GetMovement()            ; read the first movement record

; and work through each movement, until we get to the end
Do
      While SomethingMoving()       ; just wait while the Events finish off
            Wait 10                       ; any (all) the movements set up by
      End While                           ; the GetMovement above

      sequence = GetMovement()      ; Then get the next movement

Until sequence = exitseq            ; and keep going until we get to the end



End Program


;#########################################################################
;
;SomethingMoving()  - returns True if any of the Axis are moving
;
;#########################################################################

Declare Function SomethingMoving() Returns Boolean

If ArmMoving = True Then
            Return True
End If
If Wristmoving = True Then
            Return True
End If
If Shouldermoving = True Then
            Return True
End If
If Basemoving = True Then
            Return True
End If

Return False                        ; if nothing moving, then return false

End Function

;#############################################################################################
;
;     SETUP
;
;     locate all setup and initialisation code here
```

```
;
;#############################################################################################

Declare Function Setup()


Disable Events
SetEvent OnOff3Event, MB_Event
SetEvent OnOff4Event, MW_Event
SetEvent Timed1Event, MS_Event
SetEvent Timed2Event, MA_Event
Enable Events

Call CheckTestMode()                        ; see if we should be running test mode
                                            ; by being invoked from Programs 95...98

MovementPtr = SetPointer(MovementTable)     ; initialise the MovementTable Pointer

End Function


;#########################################################################
;
; GetMovement reads one line of the Movement Data Table and sets up a movement.
; The movement is then managed by the matching event as each feedback pulse is received
;
; GetMovement can be called recursively to support Groups of movements
;
; Movement Codes (in Movement Data table)
;
;      R      - restart sequence from start
;      C      - Go to known Homeposition
;      A      - Move Arm
;      B      - Move Base
;      H      - Open/Close Hand
;      S      - Move Shoulder
;      W      - Move Wrist
;      G      - Start of Movement Group
;      E      - End of Movement Group
;      X      - eXit
;
;
; returns a flag with one of the following options:
;            exitseq     found an X, so will end the whole program sequence
;            moveseq     Any of the normal movement types
;            endseqTHe end of a group of movements
;            unknownseq  Catch flag if the data table contains an error

Declare Function GetMovement() Returns Byte

Declare Byte Mode
Declare Byte Direction
Declare Byte Distance

Mode = ReadData(MovementPtr)
Direction = ReadData(MovementPtr)
Distance  = ReadData(MovementPtr)

Select Case Mode

      Case "G"
            blockend = GetMovement()       ; get the next movement line
            While blockend = moveseq
                  blockend = GetMovement() ; and keep getting them until we get to an "E" or or other
non-movement record
            End While
            Return blockend                     ; and signal that we're done

      Case "X"
            Return exitseq

      Case "R"
            MovementPtr = SetPointer(MovementTable)
            blockend = GetMovement()       ; and get the first real movement
            Return blockend
```

```
      Case "E"
            Return endseq                       ; signal the end of a group

      Case "A"
            Call Start_Arm(Direction, Distance)
            Return moveseq                      ; and we're done

      Case "B"
            Call Start_Base(Direction, Distance)                    ; set up to start moving the base
            Return moveseq                      ; and we're done

      Case "C"
            Call Homeposition(Direction)        ; move to known position
            blockend = GetMovement()      ; and get the next movement
            Return blockend

      Case "H"
            Call MoveHand(Direction)
            blockend = GetMovement()      ; and get the next movement
            Return blockend


      Case "S"
            Call Start_Shoulder(Direction, Distance)
            Return moveseq                      ; and we're done

      Case "W"
            Call Start_Wrist(Direction, Distance)
            Return moveseq                      ; and we're done

End Select

Return unknownseq

End Function


;#############################################################################################################
;
;     Start_Arm
;
;     Starts off a Arm Movement
;
;#############################################################################################################

Declare Function Start_Arm(SA_Dir As Byte, SA_Distance As Byte)

ArmMoving = True                            ; so we know that the arm is supposed to be moving
MA_Distance = SA_Distance                   ; and distance

MA_Accel = MA_Distance / 2                  ; and set the default "Accelerate until this point"
MA_Accel = MA_Accel - 1
MA_Decel = MA_Distance - MA_Accel           ; with symmetric "Decelerate from this point"

If SA_Dir = "U" Then
      MA_Dir = ArmUp                        ; translate the Up into Forward to suit the Set motor
command
      MA_MinSpeed = MAU_MinSpeed
      MA_MaxSpeed = MAU_MaxSpeed
      MA_Speed = MA_MinSpeed                ; start off at minimum speed
      SetMotor ArmMotor, ArmUp, MA_Speed    ; start the arm moving

Else
      MA_Dir = ArmDown
      MA_MinSpeed = MAD_MinSpeed
      MA_MaxSpeed = MAD_MaxSpeed
      MA_Speed = MA_MinSpeed
      SetMotor ArmMotor, ArmDown, MA_Speed ; start the arm moving
End If

Wait BounceTime                             ; wait to get rid of any rogue pulse
MA_PulseCount = 0                           ; and reset the pulsecount

End Function
```

```
;#########################################################################################################
;
;      Start_Base
;
;      Starts off a Base Movement
;
;#########################################################################################################

Declare Function Start_Base(SB_Dir As Byte, SB_Distance As Byte)

BaseMoving = True                           ; so we know that the base is supposed to be moving
MB_Distance = SB_Distance
MB_Accel = MB_Distance / 2                  ; and set the default "Accelerate until this point"
MB_Accel = MB_Accel - 1
MB_Decel = MB_Distance - MB_Accel           ; with symmetric "Decelerate from this point"


If SB_Dir = "L" Then
      MB_Dir = BaseLeft                     ; translate the Left into Forward to suit the Set motor
command
      MB_MinSpeed = MBL_MinSpeed
      MB_MaxSpeed = MBL_MaxSpeed
      MB_Speed = MB_MinSpeed
      SetMotor    BaseMotor, BaseLeft, MB_Speed  ; start the base  moving
Else
      MB_Dir = BaseRight
      MB_MinSpeed = MBR_MinSpeed
      MB_MaxSpeed = MBR_MaxSpeed
      MB_Speed = MB_MinSpeed
      SetMotor    BaseMotor, BaseRight, MB_Speed ; start the base  moving
End If

Wait BounceTime                             ; wait to get rid of any rogue pulse
MB_PulseCount = 0                           ; and reset the pulsecount

End Function

;#########################################################################################################
;
;      Start_Shoulder
;
;      Starts off a Shoulder Movement
;
;#########################################################################################################

Declare Function Start_Shoulder(SS_Dir As Byte, SS_Distance As Byte)

ShoulderMoving = True                       ; so we know that the base is supposed to be moving
MS_Distance = SS_Distance
MS_Accel = MS_Distance / 2                  ; and set the default "Accelerate until this point"
MS_Accel = MS_Accel - 1
MS_Decel = MS_Distance - MS_Accel           ; with symmetric "Decelerate from this point"




If SS_Dir = "O" Then
      MS_Dir = ShoulderOut                  ; translate the Out into Forward to suit the Set motor
command
      MS_MinSpeed = MSO_MinSpeed
      MS_MaxSpeed = MSO_MaxSpeed
      MS_Speed = MS_MinSpeed
      SetMotor    ShoulderMotor, ShoulderOut, MS_Speed ; start the shoulder moving
Else
      MS_Dir = ShoulderIn
      MS_MinSpeed = MSI_MinSpeed
      MS_MaxSpeed = MSI_MaxSpeed
      MS_Speed = MS_MinSpeed
      SetMotor    ShoulderMotor, ShoulderIn, MS_Speed  ; start the shoulder moving

End If

Wait BounceTime                             ; wait to get rid of any rogue pulse
MS_PulseCount = 0                           ; and reset the pulsecount

End Function
```

```
;#########################################################################################################
;
;      Start_Wrist
;
;      Starts off a Wrist Movement
;
;#########################################################################################################

Declare Function Start_Wrist(SW_Dir As Byte, SW_Distance As Byte)

WristMoving = True                    ; so we know that the Wrist is supposed to be moving
MW_Distance = SW_Distance                   ; this is the number of pulse to count before we stop
MW_Accel = MW_Distance / 2                  ; and set the default "Accelerate until this point"
MW_Accel = MW_Accel - 1
MW_Decel = MW_Distance - MW_Accel           ; with symmetric "Decelerate from this point"



If SW_Dir = "O" Then
      MW_Dir = WristOut                     ; translate the Out into Forward to suit the Set motor
command
      MW_MinSpeed = MWO_MinSpeed
      MW_MaxSpeed = MWO_MaxSpeed
      MW_Speed = MW_MinSpeed
      SetMotor    WristMotor, WristOut, MW_Speed ; start the wrist moving
Else
      MW_Dir = WristIn
      MW_MinSpeed = MWI_MinSpeed
      MW_MaxSpeed = MWI_MaxSpeed
      MW_Speed = MW_MinSpeed
      SetMotor    WristMotor, WristIn, MW_Speed  ; start the wrist moving

End If
Wait BounceTime                             ; wait to get rid of any rogue pulse
MW_PulseCount = 0                           ; and reset the pulsecount


End Function



;#########################################################################################################
;
;      MA_Event
;
;      Event Handler, invoked each time a pulse comes from the Arm Sensor,
;      Manages the movement of the arm, including Acceleration and Deceleration
;
;#########################################################################################################
Declare EventHandler MA_Event()

MA_PulseCount = MA_PulseCount + 1           ; count this pulse
                                            ; even if the ArmMoving is false, because HomePosition
manually
                                            ; starts the motors and waits for a stall condition

If ArmMoving = True Then

      If MA_PulseCount = MA_Distance Then

            Call StopOneMotor(ArmMotor)           ; if we're at the end
            ArmMoving = False
            Return                          ;

      End If

      If MA_PUlseCount < MA_Accel    Then  ; if we havent got to the end of the accelerate ZOne

            MA_Speed = MA_Speed + MA_Accel_Delta ; increase the speed

            If MA_Speed >= MA_MaxSpeed Then
                  MA_Speed = MA_MaxSpeed        ; unless we get to the maximum speed
                  MA_Accel = MA_PulseCount ; in which case we make this the end of the AccelZone
                  MA_Decel = MA_Distance - MA_Accel ; and set up a symmetric DecelZone
                  MA_Decel = MA_Decel - 1
```

```
            End If
      End If

      If MA_PulseCount >= MA_Decel   Then        ; if we're in the decelerate Zone

            If MA_Speed > MA_MinSpeed Then

                  MA_Speed = MA_Speed - MA_Decel_Delta ; slow down

                  If MA_Speed < MA_MInspeed Then
                        MA_Speed = MA_MinSpeed        ; but not too slow

                  End If
            End If

            ; check the end stop microswitch
            Arm_Top_Switch = ReadOnOff(Arm_Input)

            If Arm_Top_Switch = 1 Then     ; closed

                  Call StopOneMotor(ArmMotor)
                  ArmMoving = False
                  Return

            End If

      End If

      ; if we haven't decided to stop, now change the speed (or just coast alone at current speed)
      If MA_Dir = ArmUp Then
            SetMotor ArmMotor, ArmUp, MA_Speed
      Else
            SetMotor ArmMotor, ArmDown, MA_Speed
      End If

End If; ArmMoving = True


End EventHandler   ; MA_Event

;#############################################################################################
;
;     MB_Event
;
;     Event Handler, invoked each time a pulse comes from the Base Sensor,
;     Manages the movement of the arm, including Acceleration and Deceleration
;
;#############################################################################################
Declare EventHandler MB_Event()

MB_PulseCount = MB_PulseCount + 1          ; count this pulse
                                    ; even if the BaseMoving is false, because HomePosition manually
                                    ; starts the motors and waits for a stall condition


If BaseMoving = True Then

      If MB_PulseCount >= MB_Distance Then

            Call StopOneMotor(BaseMotor)         ; if we're at the end
            BaseMoving = False
            Return                        ;

      End If

      If MB_PulseCount < MB_Accel     Then ; if we havent got to the end of the accelerate ZOne

            MB_Speed = MB_Speed + MB_Accel_Delta ; increase the speed

            If MB_Speed >= MB_MaxSpeed Then

                  MB_Speed = MB_MaxSpeed   ; unless we get to the Maximum speed
                  MB_Accel = MB_PulseCount ; in which case we Make this the end of the AccelZone
                  MB_Decel = MB_Distance - MB_Accel ; and set up a symmetric DecelZone
                  MB_Decel = MB_Decel - 1
            End If
```

```
      End If

      If MB_PulseCount >= MB_Decel    Then       ; if we're in the decelerate Zone

            If MB_Speed > MB_MinSpeed Then
                  MB_Speed = MB_Speed - MB_Decel_Delta ; slow down

                  If MB_Speed < MB_MInspeed Then
                        MB_Speed = MB_MinSpeed         ; but not too slow
                  End If

            End If
            ; check the end stop microswitch
            Base_Switch = ReadOnOff(Base_Input)

            If Base_Switch = 1 Then  ; closed

                  Call StopOneMotor(BaseMotor)
                  BaseMoving = False
                  Return

            End If

      End If

      ; if we haven't decided to stop, now change the speed (or just coast alone at current speed)
      If MB_Dir = BaseRight Then
            SetMotor BaseMotor, BaseRight, MB_Speed
      Else
            SetMotor BaseMotor, BaseLeft, MB_Speed
      End If

End If; BaseMoving = True


End EventHandler   ; MB_Event

;#############################################################################################################
;
;     MS_Event
;
;     Event Handler, invoked each time a pulse comes from the Shoulder Sensor,
;     Manages the movement of the arm, including Acceleration and Deceleration
;
;#############################################################################################################
Declare EventHandler MS_Event()

MS_PulseCount = MS_PulseCount + 1          ; count this pulse
                                    ; even if the ShoulderMoving is false, because HomePosition
manually
                                    ; starts the motors and waits for a stall condition



If ShoulderMoving = True Then

      If MS_PulseCount = MS_Distance Then

            Call StopOneMotor(ShoulderMotor)           ; if we're at the end
            ShoulderMoving = False
            Return                           ;

      End If

      If MS_PUlseCount < MS_Accel     Then             ; if we havent got to the end of the accelerate
ZOne

            MS_Speed = MS_Speed + MS_Accel_Delta ; increase the speed

            If MS_Speed >= MS_MaxSpeed Then

                  MS_Speed = MS_MaxSpeed               ; unless we get to the Maximum speed
                  MS_Accel = MS_PulseCount       ; in which case we Make this the end of the AccelZone
                  MS_Decel = MS_Distance - MS_Accel     ; and set up a symmetric DecelZone
                  MS_Decel = MS_Decel - 1
```

```
            End If
      End If

      If MS_PulseCount >= MS_Decel    Then              ; if we're in the decelerate Zone

            If MS_Speed > MS_MinSpeed Then

                  MS_Speed = MS_Speed - MS_Decel_Delta ; slow down

                  If MS_Speed < MS_MInspeed Then

                        MS_Speed = MS_MinSpeed         ; but not too slow
                  End If
            End If

      End If

      ; now change the speed (or just coast alone at current speed)
      If MS_Dir = ShoulderOut Then
            SetMotor ShoulderMotor, ShoulderOUt, MS_Speed
      Else
            SetMotor ShoulderMotor, ShoulderIn, MS_Speed
      End If

End If; ShoulderMoving = True


End EventHandler   ; MS_Event


;#############################################################################################
;
;     MW_Event
;
;     Event Handler, invoked each time a pulse comes from the Wrist Sensor,
;     Manages the movement of the Wrist, including Acceleration and Deceleration
;
;#############################################################################################
Declare EventHandler MW_Event()

MW_PulseCount = MW_PulseCount + 1                ; count this pulse
                                                 ; even if the WristMoving is false, because
HomePosition manually
                                                 ; starts the motors and waits for a stall condition

If WristMoving = True Then

      If MW_PulseCount = MW_Distance Then

            Call StopOneMotor(WristMotor)        ; if we're at the end
            WristMoving = False
            Return                          ;

      End If

      If MW_PUlseCount < MW_Accel Then           ; if we havent got to the end of the accelerate ZOne

            MW_Speed = MW_Speed + MW_Accel_Delta ; increase the speed

            If MW_Speed >= MW_MaxSpeed Then      ; unless we get to the Maximum speed

                  MW_Speed = MW_MaxSpeed
                  MW_Accel = MW_PulseCount ; in which case we Make this the end of the AccelZone
                  MW_Decel = MW_Distance - MW_Accel ; and set up a symmetric DecelZone
                  MW_Decel = MW_Decel - 1
            End If
      End If

      If MW_PulseCount >= MW_Decel Then                ; if we're in the decelerate Zone

            If MW_Speed > MW_MinSpeed Then

                  MW_Speed = MW_Speed - MW_Decel_Delta ; slow down

                  If MW_Speed < MW_MInspeed Then
                        MW_Speed = MW_MinSpeed         ; but not too slow
```

```
                End If
            End If

      End If

      ; now change the speed (or just coast alone at current speed)
      If MW_Dir = WristOut Then
            SetMotor WristMotor, WristOut, MW_Speed
      Else
            SetMotor WristMotor, WristIn,  MW_Speed
      End If

End If; WristMoving = True


End EventHandler   ; MW_Event


;#########################################################################################################
;
;     MoveHand
;     opens or close the grab, slowly
;
;#########################################################################################################

Declare Function MoveHand(OpenClose As Byte)


Declare Byte servo_release_point = 20
Declare Byte Servo1_Current_Angle
Declare Byte Servo_Open =20
Declare Byte Servo_Close_Block =       70

If OpenClose = "O" Then


      ; Open
      Servo1_Current_Angle = Servo_Close_Block
      DisplayChars "H","O"

      Do

            SetServo 1,"F",Servo1_Current_Angle   ; move to closed
            Wait  3
            Servo1_Current_Angle = Servo1_Current_Angle - 1

      Until Servo1_Current_Angle < Servo_release_Point

      Wait  10
      SetServo 1,"B",Servo_Open

Else
      ; Close
      Servo1_Current_Angle = servo_release_point
      DisplayChars "H", "C"

      Do

            SetServo 1, "F", Servo1_Current_Angle
            Wait 10
            Servo1_Current_angle = Servo1_Current_Angle + 1

      Until Servo1_Current_angle >= Servo_Close_Block

      Wait 20

End If


End Function ; MoveHand

;#########################################################################
;
; MoveArm  - Sets up an arm movement for a required number of pulses
;            and waits until the movement is finished.
;            Can't use if you want to have concurrent movements
```

```
;
;
;#############################################################################

Declare Function MoveArm(MAF_Dir As Byte, MAF_Distance As Byte)

Call Start_Arm(MAF_Dir, MAF_Distance)

Do
      DisplayNumber MA_pulsecount          ; and we can display here because we don't expect any other
movement to be happening
      Wait 10                              ; wait until MA_Event decides that the arm has stopped

Until ArmMoving = False

End Function

;#############################################################################
;
; MoveBase  - Sets up an Base movement for a required number of pulses
;           and waits until the movement is finished.
;           Can't use if you want to have concurrent movements
;
;
;#############################################################################

Declare Function MoveBase(MBF_Dir As Byte, MBF_Distance As Byte)

Call Start_Base(MBF_Dir,MBF_Distance)      ; set everything up to start moving

Do
      DisplayNumber MB_pulsecount          ; and we can display here because we don't expect any other
movement to be happening
      Wait 10                              ; wait until MB_Event decides that the arm has stopped

Until BaseMoving = False

End Function

;#############################################################################
;
; MoveShoulder  - Sets up an Shoulder movement for a required number of pulses
;           and waits until the movement is finished.
;           Can't use if you want to have concurrent movements
;
;
;#############################################################################

Declare Function MoveShoulder(MSF_Dir As Byte, MSF_Distance As Byte)

Call Start_Shoulder(MSF_Dir, MSF_Distance)       ; set everything thing up to start moving

Do
      DisplayNumber MS_pulsecount          ; and we can display here because we don't expect any other
movement to be happening

      Wait 10                              ; wait until MA_Event decides that the arm has stopped

Until ShoulderMoving = False

End Function

;#############################################################################
;
; MoveWrist  - Sets up an Wrist movement for a required number of pulses
;           and waits until the movement is finished.
;           Can't use if you want to have concurrent movements
;
;
;#############################################################################

Declare Function MoveWrist(MWF_Dir As Byte, MWF_Distance As Byte)

Call Start_Wrist(MWF_Dir, MWF_Distance)    ; set wrist up to start moving.

Do
```

```
      DisplayNumber MW_pulsecount

      Wait 10                              ; wait until MW_Event decides that the Wrist has stopped

Until WristMoving = False

End Function


;###############################################################################
;
; HOMEPOSITION
;
; sets the arm into a known position, using the stall stops and relying on the pulse counts from the
encoders
; to stop when stalled.
;
;###############################################################################

Declare Function HomePosition(HomeRight As Byte)



; PlayTune         HomeTune


; see if arm is at top
Arm_Top_Switch = ReadOnOff(Arm_Input)

If Arm_Top_Switch = Closed Then

      DisplayChars "A","D"
      ; first get the arm down off the top switch, if jammed up there from last action
      SetMotor ArmMotor, armDown,    Arm_stall_speed
      Wait 200

End If

;Move ARM Up to top switch

DisplayChars "A","U"
SetMotor ArmMotor, ArmUp, Arm_Stall_Speed

Do

      Arm_Top_Switch = ReadOnOff(Arm_Input)

Until Arm_Top_Switch = Closed

Call StopOneMotor(ArmMotor)

DisplayChars "A","D"
; Move ARM Down to Require Level
Call MoveArm("D", 25)

; move WRIST IN to Stall

DisplayChars "W","I"
SetMotor wristmotor, WristIn, Wrist_Stall_Speed

Do

      MW_PulseCount = 0        ; clear pulse count
      Wait  stall_delay        ; wait for a bit
      DisplayNumber     MW_PulseCount

Until MW_PulseCount = 0        ; if we have any pulses, then keep going

StopMotors                     ; once we have no pulses, then stalled

DisplayChars "S","I"
; MOVE SHOULDER IN to Stall
SetMotor Shouldermotor, ShoulderIn, Shoulder_Stall_Speed

Do
```

```
      MS_PulseCount = 0
      Wait  stall_delay       ; wait for a bit
      DisplayNumber     MS_PulseCount

Until MS_PulseCount =    0           ; if we have any pulses, then keep going

StopMotors                    ; once we have no pulses, then stalled


; MOVE SHOULDER OUT to 90 Degrees

Call        MoveShoulder("O", 17)

; MOVE WRIST OUT To upright
Call        MoveWrist("O", 16)

StopTune

If HomeRight = "L" Then

      ; BASE to left switch
      SetMotor BaseMotor, BaseLeft,  Base_Stall_Speed
Else

      ; BASE to Right switch
      SetMotor BaseMotor, BaseRight, Base_Stall_Speed
End If

; and wait until the base_Switch is closed

Do

      Base_Switch = ReadOnOff(Base_Input)

Until Base_Switch = Closed

StopMotors                    ; once we have no pulses, then stalled

End Function ; HomePosition

;###########################################################################
;
; StopOneMotor
;
;     Function to Avoid having to use
;     SetMOtor MotorNo, Direction, 0
;     to stop a motor
;###########################################################################

Declare Function StopOneMotor(StopWhichMotor As Byte)

Select Case StopWhichMotor

      Case 1
                  SetMotor 1,Forward,0
      Case 2
                  SetMotor 2,Forward,0
      Case 3
                  SetMotor 3,Forward,0
      Case 4
                  SetMotor 4,Forward,0

End Select
End Function


;###########################################################################
;
; Test_Mode_Check
;
;     use the Program Number to see if we want to do test modes.
;
;     if so, then move backwards and forwards (or in/out or up/down) forever
;
;###########################################################################
```

```
Declare Function CheckTestMode()


Declare Byte ProgNum

ProgNum = GetProgramNumber()
Arm_Top_Switch = ReadOnOff(Arm_Input) ; use armswitch to decide whether
                                ; to move in or out first

Select Case Prognum

      Case 98
            If Arm_Top_Switch = Closed Then
                  Loop
                        Call MoveArm("U", 15)
                        Wait 200
                        Call MoveArm("D", 15)
                        Wait 200
                  End Loop
            Else
                  Loop
                        Call MoveArm("D",15)
                        Wait 200
                        Call MoveArm("U", 15)
                        Wait 200
                  End Loop
            End If

      Case 97
            If Arm_Top_Switch = Closed Then
                  Loop
                        Call MoveBase("L", 15)
                        Wait 200
                        Call MoveBase("R", 15)
                        Wait 200
                  End Loop
            Else
                  Loop
                        Call MoveBase("R",15)
                        Wait 200
                        Call MoveBase("L", 15)
                        Wait 200
                  End Loop
            End If


      Case 96
            If Arm_Top_Switch = Closed Then
                  Loop
                        Call MoveSHoulder("O", 15)
                        Wait 200
                        Call MoveShoulder("I", 15)
                        Wait 200
                  End Loop
            Else
                  Loop
                        Call MoveShoulder("I",15)
                        Wait 200
                        Call MoveShoulder("O", 15)
                        Wait 200
                  End Loop
            End If


      Case 95
            If Arm_Top_Switch = Closed Then
                  Loop
                        Call MoveWrist("O", 12)
                        Wait 200
                        Call MoveWrist("I", 12)
                        Wait 200
                  End Loop
            Else
                  Loop
                        Call MoveWrist("I",12)
```

```
                    Wait 200
                    Call MoveWrist("O", 12)
                    Wait 200
                End Loop

            End If

      Case Else

            Return

End Select

End Function ; CheckTestMode

;#########################################################

; Data Table include

Include File Hanoi_move_table.txt
```