Application Note: QCI-AN055 QuickSilver Controls, Inc.
Date: 7 August 2008 www.QuickSilverControls.com

# 3-Axes Pick and Place

Associated files:

- Basic Pick and Place – C.qcp
- Basic Pick and Place – X.qcp
- Basic Pick and Place – Y.qcp
- Intermediate Pick and Place – Master.qcp
- Intermediate Pick and Place – Slave 1.qcp
- Intermediate Pick and Place – Slave 2.qcp
- XY Coord.txt

## Required Software

QuickControl
Rev 4.6 or higher

# Overview

This document describes a Pick and Place application. This particular application is a precision system for filling a carton with golf balls. The system places golf balls at 18 predetermined points in a carton and then moves a new carton in place when complete. The 3-axes communicate with each other to move the hose to a predetermined point and remove a locking pin to drop a golf ball then move the full carton to make room for an empty carton.

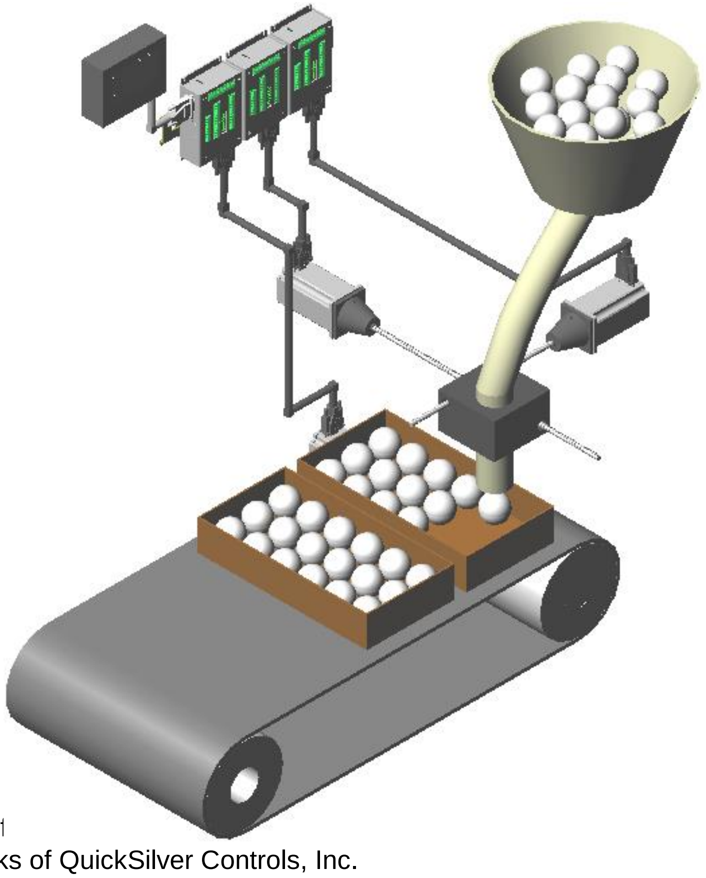
The application notes contains two example:

- Basic: Uses I/O to communicate between controllers.
- Intermediate: Uses CANopen® to communicate between controllers.

The user should have an understanding of the following topics:

- Homing Techniques (QCI-AN001)
- Register Files (QCI-AN048)
- CANopen® (CANopen User Manual)-required for intermediate example only.

This application note means to provide an understanding of a basic 3-axes system using QuickSilver controllers and motors.



# System Parameters

This example requires:

- 3 x Controllers, Motors and Associated Cables
- 1 x Conveyer Belt
- 1 x Actuator
- 1 x Ball Hopper
- Multiple Sensors, Balls and Cartons
- Optional HMI or PC controller

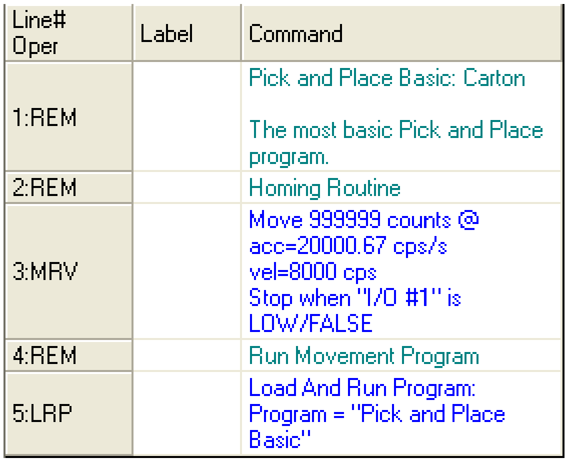
# Basic Pick and Place Example

This example uses I/O to communicate between the controllers.

## Basic Pick and Place - C.qcp

### Pick and Place Homing

This program moves positive to find a sensor home position.



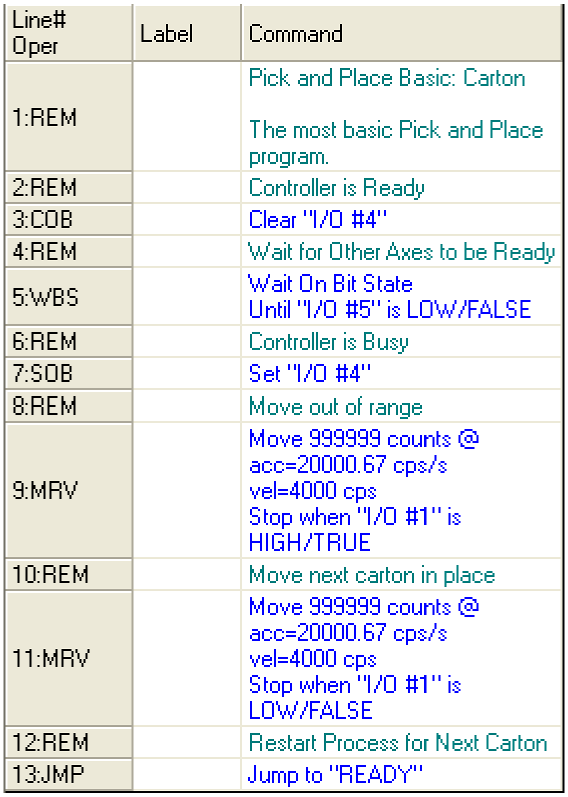
| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Basic: Carton<br><br>The most basic Pick and Place program. |
| 2:REM | | Homing Routine |
| 3:MRV | | Move 999999 counts @<br>acc=20000.67 cps/s<br>vel=8000 cps<br>Stop when "I/O #1" is LOW/FALSE |
| 4:REM | | Run Movement Program |
| 5:LRP | | Load And Run Program:<br>Program = "Pick and Place Basic" |

### Pick and Place Basic

The first operation sets the controller to a ready state and waits for the other controllers to finish filling the carton with balls.

Once ready, the filled carton moves down the conveyer while an empty carton moves into place for filling.

The process repeats for the next carton.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Basic: Carton<br><br>The most basic Pick and Place program. |
| 2:REM | | Controller is Ready |
| 3:COB | | Clear "I/O #4" |
| 4:REM | | Wait for Other Axes to be Ready |
| 5:WBS | | Wait On Bit State<br>Until "I/O #5" is LOW/FALSE |
| 6:REM | | Controller is Busy |
| 7:SOB | | Set "I/O #4" |
| 8:REM | | Move out of range |
| 9:MRV | | Move 999999 counts @<br>acc=20000.67 cps/s<br>vel=4000 cps<br>Stop when "I/O #1" is HIGH/TRUE |
| 10:REM | | Move next carton in place |
| 11:MRV | | Move 999999 counts @<br>acc=20000.67 cps/s<br>vel=4000 cps<br>Stop when "I/O #1" is LOW/FALSE |
| 12:REM | | Restart Process for Next Carton |
| 13:JMP | | Jump to "READY" |

## Basic Pick and Place – X.qcp and Basic Pick and Place – Y.qcp

### Pick and Place Homing

This homing sequence uses the Hard Stop home sequence to determine the minimum and maximum travel. This requires some initialization steps:

1) Disable Kill Motor Conditions – needed to keep controller from ending the program when hitting a Hard Stop.
2) Set Error Limits – needed to let the motor register sense a Hard Stop quickly.
3) Set Torque Limits – needed to prevent the motor from damaging the equipment.

The first move command is in a negative direction. This finds the upper limit then moves away a small amount away before setting the actual position.

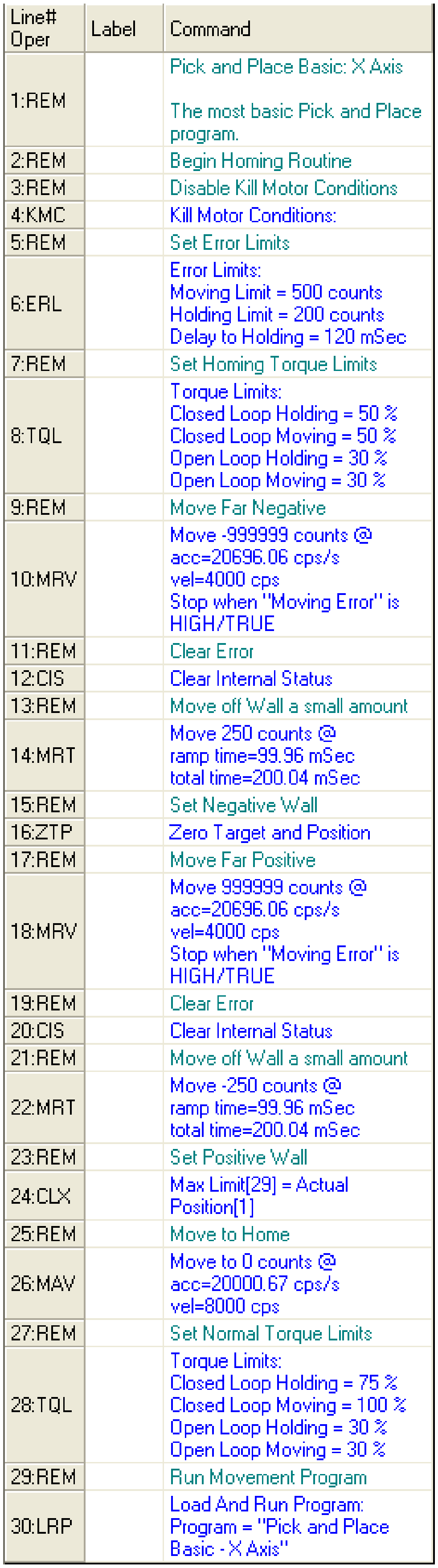
The next move is in a positive direction. This finds the lower limit. This works the same as the far positive move.

The next operation sets the normal Torque Limits then runs the movement program.

Both X-axis and Y-axis use the homing routine. Line 32 loads either the X or Y Pick and Place program.

Refer to the Application Note “QCI-AN001 Homing Techniques” for further explanation.

After homing, the program load and runs “Pick and Place Basic – X” and begins the movement sequence.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Basic: X Axis<br><br>The most basic Pick and Place program. |
| 2:REM | | Begin Homing Routine |
| 3:REM | | Disable Kill Motor Conditions |
| 4:KMC | | Kill Motor Conditions: |
| 5:REM | | Set Error Limits |
| 6:ERL | | Error Limits:<br>Moving Limit = 500 counts<br>Holding Limit = 200 counts<br>Delay to Holding = 120 mSec |
| 7:REM | | Set Homing Torque Limits |
| 8:TQL | | Torque Limits:<br>Closed Loop Holding = 50 %<br>Closed Loop Moving = 50 %<br>Open Loop Holding = 30 %<br>Open Loop Moving = 30 % |
| 9:REM | | Move Far Negative |
| 10:MRV | | Move -999999 counts @<br>acc=20696.06 cps/s<br>vel=4000 cps<br>Stop when "Moving Error" is HIGH/TRUE |
| 11:REM | | Clear Error |
| 12:CIS | | Clear Internal Status |
| 13:REM | | Move off Wall a small amount |
| 14:MRT | | Move 250 counts @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 15:REM | | Set Negative Wall |
| 16:ZTP | | Zero Target and Position |
| 17:REM | | Move Far Positive |
| 18:MRV | | Move 999999 counts @<br>acc=20696.06 cps/s<br>vel=4000 cps<br>Stop when "Moving Error" is HIGH/TRUE |
| 19:REM | | Clear Error |
| 20:CIS | | Clear Internal Status |
| 21:REM | | Move off Wall a small amount |
| 22:MRT | | Move -250 counts @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 23:REM | | Set Positive Wall |
| 24:CLX | | Max Limit[29] = Actual Position[1] |
| 25:REM | | Move to Home |
| 26:MAV | | Move to 0 counts @<br>acc=20000.67 cps/s<br>vel=8000 cps |
| 27:REM | | Set Normal Torque Limits |
| 28:TQL | | Torque Limits:<br>Closed Loop Holding = 75 %<br>Closed Loop Moving = 100 %<br>Open Loop Holding = 30 %<br>Open Loop Moving = 30 % |
| 29:REM | | Run Movement Program |
| 30:LRP | | Load And Run Program:<br>Program = "Pick and Place Basic - X Axis" |

## Basic Pick and Place – X.qcp

### Pick and Place Basic – X Axis

This program uses a For-Next loop to control the amount of balls per carton based on the Register File Array, XY Coord.

The next command clears I/O #4 to signal a Ready Status to the other axes. Then this axis waits for the other axes to be ready.

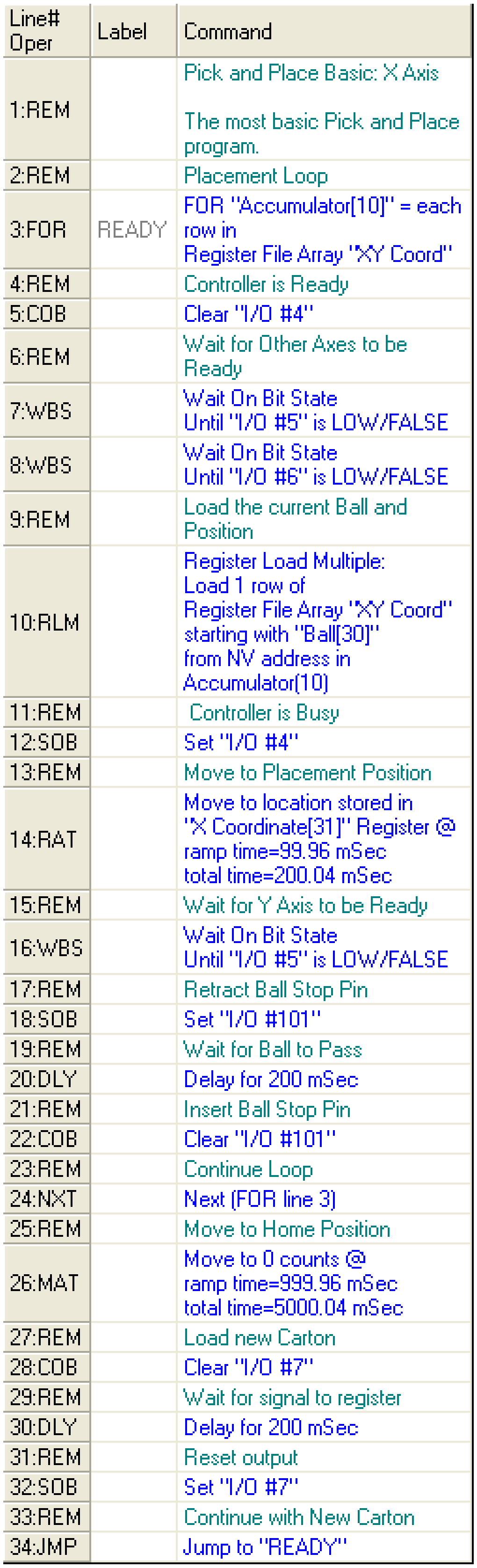
The next command loads data from the Register File Array, “XY Coord.txt”. The file contains the drop locations.

The axis sets I/O #4 to signal a Busy Status to the other axes. Then this axis begins a move command to the location loaded from XY Coord.

The axis waits for the Y-axis to be finished moving before beginning the ball drop procedure. This procedure triggers an actuator to retract a stop pin and replace the pin 200 milliseconds later.

After the ball drop, the loop repeats. This will continue until the end of the XY Coord file. At the end of the file, the axis moves back to the home position and then clears I/O #7. This tells the C axis to load the next carton.

The program then jumps back to the beginning to fill the next carton.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Basic: X Axis<br><br>The most basic Pick and Place program. |
| 2:REM | | Placement Loop |
| 3:FOR | READY | FOR "Accumulator[10]" = each row in<br>Register File Array "XY Coord" |
| 4:REM | | Controller is Ready |
| 5:COB | | Clear "I/O #4" |
| 6:REM | | Wait for Other Axes to be Ready |
| 7:WBS | | Wait On Bit State<br>Until "I/O #5" is LOW/FALSE |
| 8:WBS | | Wait On Bit State<br>Until "I/O #6" is LOW/FALSE |
| 9:REM | | Load the current Ball and Position |
| 10:RLM | | Register Load Multiple:<br>Load 1 row of<br>Register File Array "XY Coord"<br>starting with "Ball[30]"<br>from NV address in<br>Accumulator(10) |
| 11:REM | | Controller is Busy |
| 12:SOB | | Set "I/O #4" |
| 13:REM | | Move to Placement Position |
| 14:RAT | | Move to location stored in<br>"X Coordinate[31]" Register @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 15:REM | | Wait for Y Axis to be Ready |
| 16:WBS | | Wait On Bit State<br>Until "I/O #5" is LOW/FALSE |
| 17:REM | | Retract Ball Stop Pin |
| 18:SOB | | Set "I/O #101" |
| 19:REM | | Wait for Ball to Pass |
| 20:DLY | | Delay for 200 mSec |
| 21:REM | | Insert Ball Stop Pin |
| 22:COB | | Clear "I/O #101" |
| 23:REM | | Continue Loop |
| 24:NXT | | Next (FOR line 3) |
| 25:REM | | Move to Home Position |
| 26:MAT | | Move to 0 counts @<br>ramp time=999.96 mSec<br>total time=5000.04 mSec |
| 27:REM | | Load new Carton |
| 28:COB | | Clear "I/O #7" |
| 29:REM | | Wait for signal to register |
| 30:DLY | | Delay for 200 mSec |
| 31:REM | | Reset output |
| 32:SOB | | Set "I/O #7" |
| 33:REM | | Continue with New Carton |
| 34:JMP | | Jump to "READY" |

## Basic Pick and Place – Y.qcp

### Pick and Place Basic – Y Axis

This program uses a For-Next loop to control the amount of balls per carton based on the Register File Array, XY Coord.

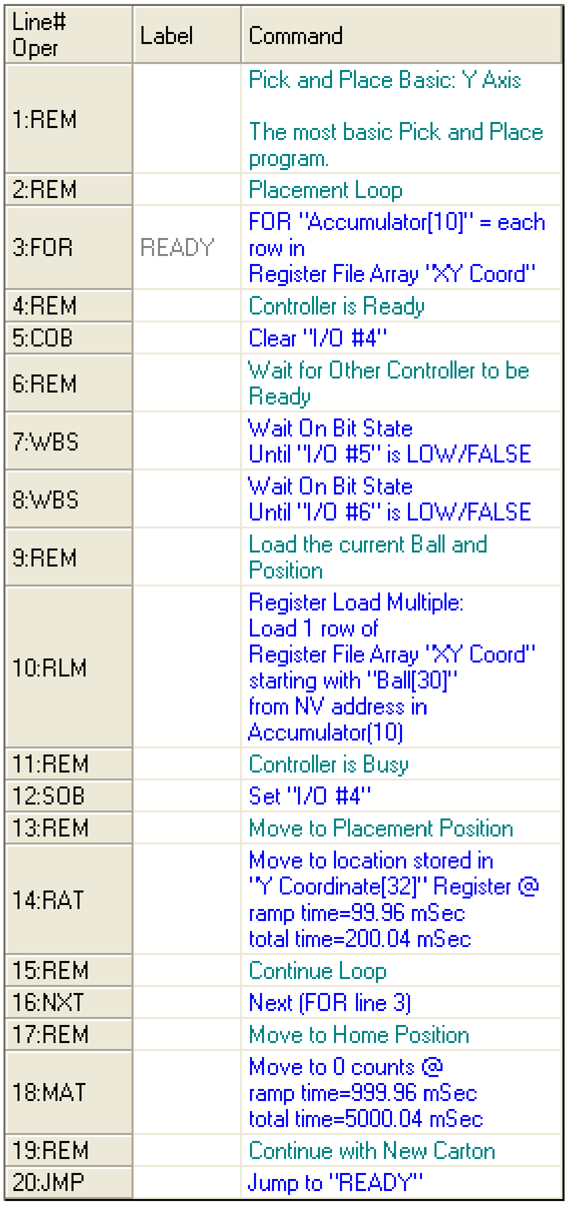
The next command clears I/O #4 to signal a Ready Status to the other axes. Then this axis waits for the other axes to be ready.

The next command loads data from the Register File Array, “XY Coord.txt”. The file contains the drop locations.

The axis sets I/O #4 to signal a Busy Status to the other axes. Then this axis begins a move command to the location loaded from XY Coord.

After the ball drop, the loop repeats. This will continue until the end of the XY Coord file.

The program then jumps back to the beginning to fill the next carton.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Basic: Y Axis<br>The most basic Pick and Place program. |
| 2:REM | | Placement Loop |
| 3:FOR | READY | FOR "Accumulator[10]" = each row in Register File Array "XY Coord" |
| 4:REM | | Controller is Ready |
| 5:COB | | Clear "I/O #4" |
| 6:REM | | Wait for Other Controller to be Ready |
| 7:WBS | | Wait On Bit State<br>Until "I/O #5" is LOW/FALSE |
| 8:WBS | | Wait On Bit State<br>Until "I/O #6" is LOW/FALSE |
| 9:REM | | Load the current Ball and Position |
| 10:RLM | | Register Load Multiple:<br>Load 1 row of<br>Register File Array "XY Coord"<br>starting with "Ball[30]"<br>from NV address in<br>Accumulator(10) |
| 11:REM | | Controller is Busy |
| 12:SOB | | Set "I/O #4" |
| 13:REM | | Move to Placement Position |
| 14:RAT | | Move to location stored in<br>"Y Coordinate[32]" Register @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 15:REM | | Continue Loop |
| 16:NXT | | Next (FOR line 3) |
| 17:REM | | Move to Home Position |
| 18:MAT | | Move to 0 counts @<br>ramp time=999.96 mSec<br>total time=5000.04 mSec |
| 19:REM | | Continue with New Carton |
| 20:JMP | | Jump to "READY" |

# Intermediate Pick and Place

This example uses CANopen® to communicate between controllers.

## Intermediate Pick and Place - Master.qcp

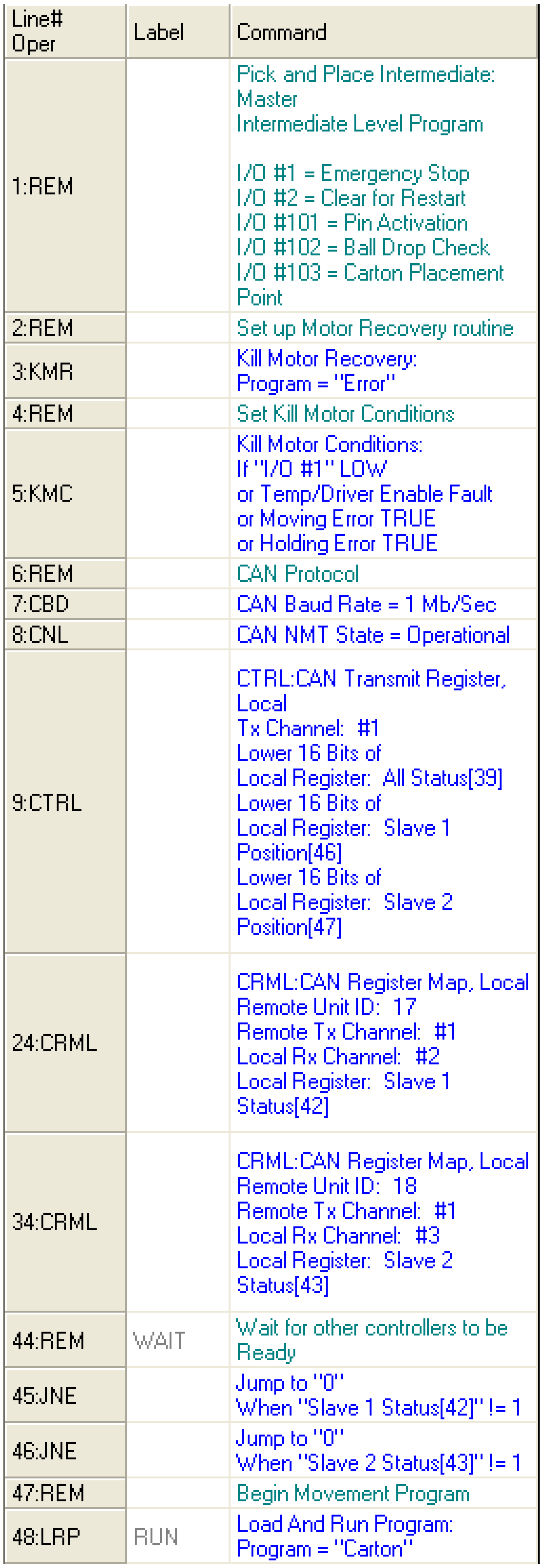
### Initialization

This program will initialize the Master controller for control of the whole system.

The first command establishes a Kill Motor Recovery routine, Error. The Kill Motor triggers with an emergency stop switch, a Temperature/Driver Enable Fault, a Moving Error or Holding Error.

The next set of commands sets up the CANopen Protocol, which allows the controller to transmit and receive registers from the Slave controllers. The Master transmits All Status[39], Slave 1 Position[46] and Slave 2 Position[47] while receiving the Slave 1 Status[42] and Slave 2 Status[43]. All of the register numbers in the Master program are local.

After initializing CANopen and Kill Motor Conditions, the Master waits for the Slaves to transmit a Ready Status before loading the Carton program.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Intermediate:<br>Master<br>Intermediate Level Program<br><br>I/O #1 = Emergency Stop<br>I/O #2 = Clear for Restart<br>I/O #101 = Pin Activation<br>I/O #102 = Ball Drop Check<br>I/O #103 = Carton Placement Point |
| 2:REM | | Set up Motor Recovery routine |
| 3:KMR | | Kill Motor Recovery:<br>Program = "Error" |
| 4:REM | | Set Kill Motor Conditions |
| 5:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW<br>or Temp/Driver Enable Fault<br>or Moving Error TRUE<br>or Holding Error TRUE |
| 6:REM | | CAN Protocol |
| 7:CBD | | CAN Baud Rate = 1 Mb/Sec |
| 8:CNL | | CAN NMT State = Operational |
| 9:CTRL | | CTRL:CAN Transmit Register, Local<br>Tx Channel: #1<br>Lower 16 Bits of<br>Local Register: All Status[39]<br>Lower 16 Bits of<br>Local Register: Slave 1 Position[46]<br>Lower 16 Bits of<br>Local Register: Slave 2 Position[47] |
| 24:CRML | | CRML:CAN Register Map, Local<br>Remote Unit ID: 17<br>Remote Tx Channel: #1<br>Local Rx Channel: #2<br>Local Register: Slave 1 Status[42] |
| 34:CRML | | CRML:CAN Register Map, Local<br>Remote Unit ID: 18<br>Remote Tx Channel: #1<br>Local Rx Channel: #3<br>Local Register: Slave 2 Status[43] |
| 44:REM | WAIT | Wait for other controllers to be Ready |
| 45:JNE | | Jump to "0"<br>When "Slave 1 Status[42]" != 1 |
| 46:JNE | | Jump to "0"<br>When "Slave 2 Status[43]" != 1 |
| 47:REM | | Begin Movement Program |
| 48:LRP | RUN | Load And Run Program:<br>Program = "Carton" |

# Intermediate Pick and Place - Master.qcp

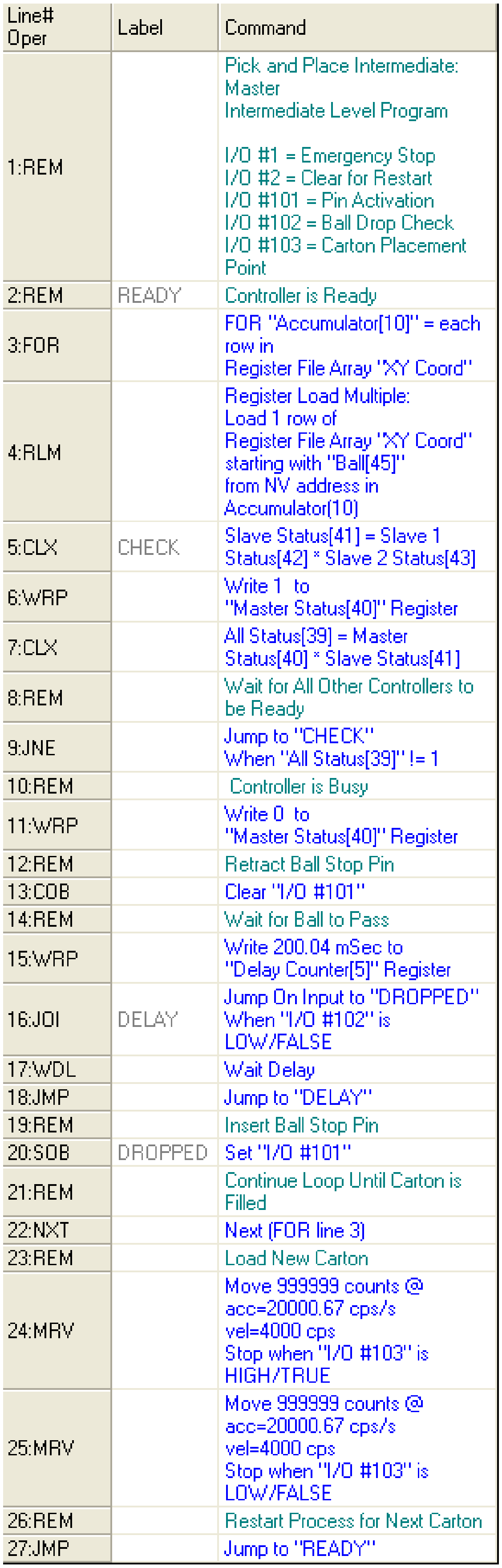
## Carton

This program loads data from the file “XY Coord” into the registers Ball[45], Slave 1 Position[46] and Slave 2 Position[47] for transmission to the Slaves. Once all 18 balls have been loaded, this program executes a move, which runs the conveyer belt to remove a full carton and load an empty carton.

The first command establishes a loop, which will run until the end of the Register File Array, XY Coord.

The program checks the Status of the Slaves to confirm their movements are complete before retracting the holding pin, dropping a golf ball and then replacing the holding pin.

After running through all of the data in XY Coord, the controller moves the next carton in place then restarts the sequence.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Intermediate: Master<br>Intermediate Level Program<br><br>I/O #1 = Emergency Stop<br>I/O #2 = Clear for Restart<br>I/O #101 = Pin Activation<br>I/O #102 = Ball Drop Check<br>I/O #103 = Carton Placement Point |
| 2:REM | READY | Controller is Ready |
| 3:FOR | | FOR "Accumulator[10]" = each row in<br>Register File Array "XY Coord" |
| 4:RLM | | Register Load Multiple:<br>Load 1 row of<br>Register File Array "XY Coord"<br>starting with "Ball[45]"<br>from NV address in<br>Accumulator(10) |
| 5:CLX | CHECK | Slave Status[41] = Slave 1 Status[42] * Slave 2 Status[43] |
| 6:WRP | | Write 1 to<br>"Master Status[40]" Register |
| 7:CLX | | All Status[39] = Master Status[40] * Slave Status[41] |
| 8:REM | | Wait for All Other Controllers to be Ready |
| 9:JNE | | Jump to "CHECK"<br>When "All Status[39]" != 1 |
| 10:REM | | Controller is Busy |
| 11:WRP | | Write 0 to<br>"Master Status[40]" Register |
| 12:REM | | Retract Ball Stop Pin |
| 13:COB | | Clear "I/O #101" |
| 14:REM | | Wait for Ball to Pass |
| 15:WRP | | Write 200.04 mSec to<br>"Delay Counter[5]" Register |
| 16:JOI | DELAY | Jump On Input to "DROPPED"<br>When "I/O #102" is<br>LOW/FALSE |
| 17:WDL | | Wait Delay |
| 18:JMP | | Jump to "DELAY" |
| 19:REM | | Insert Ball Stop Pin |
| 20:SOB | DROPPED | Set "I/O #101" |
| 21:REM | | Continue Loop Until Carton is Filled |
| 22:NXT | | Next (FOR line 3) |
| 23:REM | | Load New Carton |
| 24:MRV | | Move 999999 counts @<br>acc=20000.67 cps/s<br>vel=4000 cps<br>Stop when "I/O #103" is<br>HIGH/TRUE |
| 25:MRV | | Move 999999 counts @<br>acc=20000.67 cps/s<br>vel=4000 cps<br>Stop when "I/O #103" is<br>LOW/FALSE |
| 26:REM | | Restart Process for Next Carton |
| 27:JMP | | Jump to "READY" |

# Intermediate Pick and Place - Master.qcp

## Error

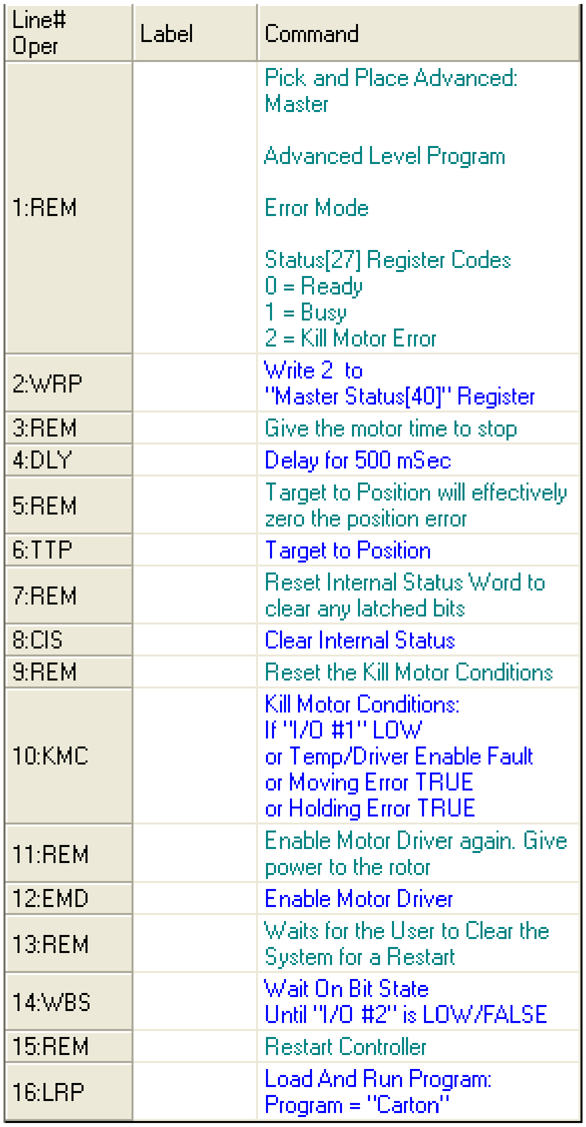
This program recovers the controller and motor from a Kill Motor event.

The first command sets the Status to an Error state and then waits for 500 milliseconds before beginning recovery.

The Target to Position command sets the current actual position as the current target position. The motor then Clears Internal Status which clears the Kill Motor error.

The Kill Motor Conditions reset and the Motor Driver re-enables before waiting for an All Clear command from the user.

The last command returns operation at the movement program.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Advanced:<br>Master<br><br>Advanced Level Program<br><br>Error Mode<br><br>Status[27] Register Codes<br>0 = Ready<br>1 = Busy<br>2 = Kill Motor Error |
| 2:WRP | | Write 2 to<br>"Master Status[40]" Register |
| 3:REM | | Give the motor time to stop |
| 4:DLY | | Delay for 500 mSec |
| 5:REM | | Target to Position will effectively zero the position error |
| 6:TTP | | Target to Position |
| 7:REM | | Reset Internal Status Word to clear any latched bits |
| 8:CIS | | Clear Internal Status |
| 9:REM | | Reset the Kill Motor Conditions |
| 10:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW<br>or Temp/Driver Enable Fault<br>or Moving Error TRUE<br>or Holding Error TRUE |
| 11:REM | | Enable Motor Driver again. Give power to the rotor |
| 12:EMD | | Enable Motor Driver |
| 13:REM | | Waits for the User to Clear the System for a Restart |
| 14:WBS | | Wait On Bit State<br>Until "I/O #2" is LOW/FALSE |
| 15:REM | | Restart Controller |
| 16:LRP | | Load And Run Program:<br>Program = "Carton" |

# Intermediate Pick and Place - Master.qcp

## Error

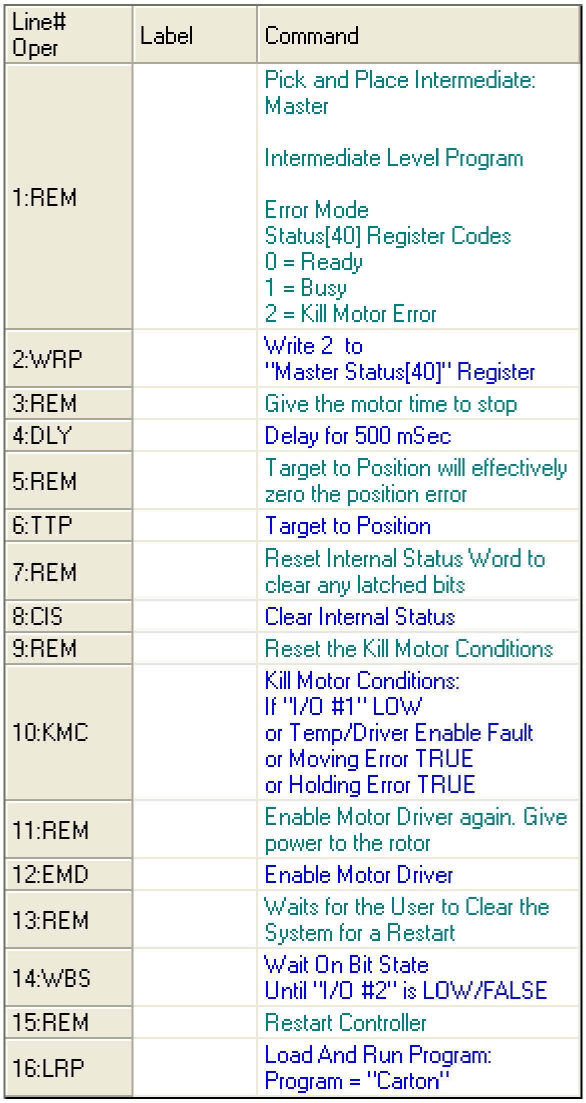
This program recovers the controller and motor from a Kill Motor event.

The first command sets the Status to an Error state and then waits for 500 milliseconds before beginning recovery.

The Target to Position command sets the current actual position as the current target position. The motor then Clears Internal Status which clears the Kill Motor error.

The Kill Motor Conditions reset and the Motor Driver re-enables before waiting for an All Clear command from the user.

The last command returns operation at the movement program.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Intermediate:<br>Master<br><br>Intermediate Level Program<br><br>Error Mode<br>Status[40] Register Codes<br>0 = Ready<br>1 = Busy<br>2 = Kill Motor Error |
| 2:WRP | | Write 2 to<br>"Master Status[40]" Register |
| 3:REM | | Give the motor time to stop |
| 4:DLY | | Delay for 500 mSec |
| 5:REM | | Target to Position will effectively zero the position error |
| 6:TTP | | Target to Position |
| 7:REM | | Reset Internal Status Word to clear any latched bits |
| 8:CIS | | Clear Internal Status |
| 9:REM | | Reset the Kill Motor Conditions |
| 10:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW<br>or Temp/Driver Enable Fault<br>or Moving Error TRUE<br>or Holding Error TRUE |
| 11:REM | | Enable Motor Driver again. Give power to the rotor |
| 12:EMD | | Enable Motor Driver |
| 13:REM | | Waits for the User to Clear the System for a Restart |
| 14:WBS | | Wait On Bit State<br>Until "I/O #2" is LOW/FALSE |
| 15:REM | | Restart Controller |
| 16:LRP | | Load And Run Program:<br>Program = "Carton" |

# Intermediate Pick and Place – Slave 1.qcp and Intermediate Pick and Place – Slave 2.qcp

## Initialization

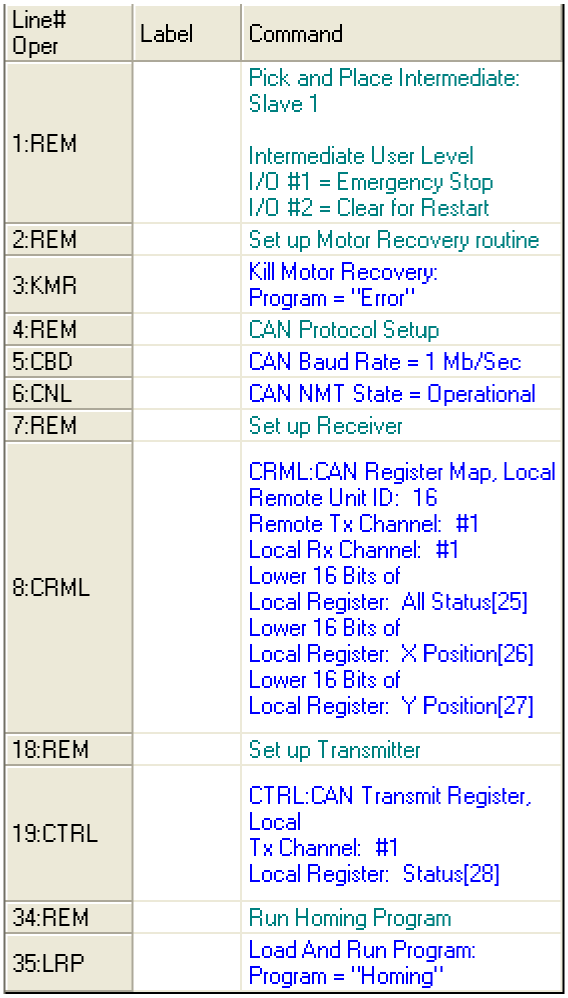
This program will initialize the Slave controllers.

The first command establishes a Kill Motor Recovery routine, Error.

The next set of commands sets up the CANopen® Protocol, which allows the controller to transmit and receive registers from the Master controller. The Slave controllers receive register All Status[25], X Position[26] and Y Position[27] from the Master controller while transmitting Status[28].

After initializing CANopen, the Slave begins the Homing routine.

There is no difference between Slave 1 and Slave 2.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Pick and Place Intermediate:<br>Slave 1<br><br>Intermediate User Level<br>I/O #1 = Emergency Stop<br>I/O #2 = Clear for Restart |
| 2:REM | | Set up Motor Recovery routine |
| 3:KMR | | Kill Motor Recovery:<br>Program = "Error" |
| 4:REM | | CAN Protocol Setup |
| 5:CBD | | CAN Baud Rate = 1 Mb/Sec |
| 6:CNL | | CAN NMT State = Operational |
| 7:REM | | Set up Receiver |
| 8:CRML | | CRML:CAN Register Map, Local<br>Remote Unit ID: 16<br>Remote Tx Channel: #1<br>Local Rx Channel: #1<br>Lower 16 Bits of<br>Local Register: All Status[25]<br>Lower 16 Bits of<br>Local Register: X Position[26]<br>Lower 16 Bits of<br>Local Register: Y Position[27] |
| 18:REM | | Set up Transmitter |
| 19:CTRL | | CTRL:CAN Transmit Register, Local<br>Tx Channel: #1<br>Local Register: Status[28] |
| 34:REM | | Run Homing Program |
| 35:LRP | | Load And Run Program:<br>Program = "Homing" |

## Intermediate Pick and Place – Slave 1.qcp and Intermediate Pick and Place – Slave 2.qcp

### Homing

This program will home the Slave controllers to the minimum and maximum points.

This homing sequence uses the Hard Stop sequence to locate to the minimum and maximum travel. This requires some initialization steps:
1) Disable Kill Motor Conditions – needed to keep controller from ending the program when hitting a Hard Stop.
2) Set Error Limits – needed to let the motor register sense a Hard Stop quickly.
3) Set Torque Limits – needed to prevent the motor from damaging the equipment.

The first move command is in a negative direction. This finds the upper limit then moves away a small amount away before setting the actual position.

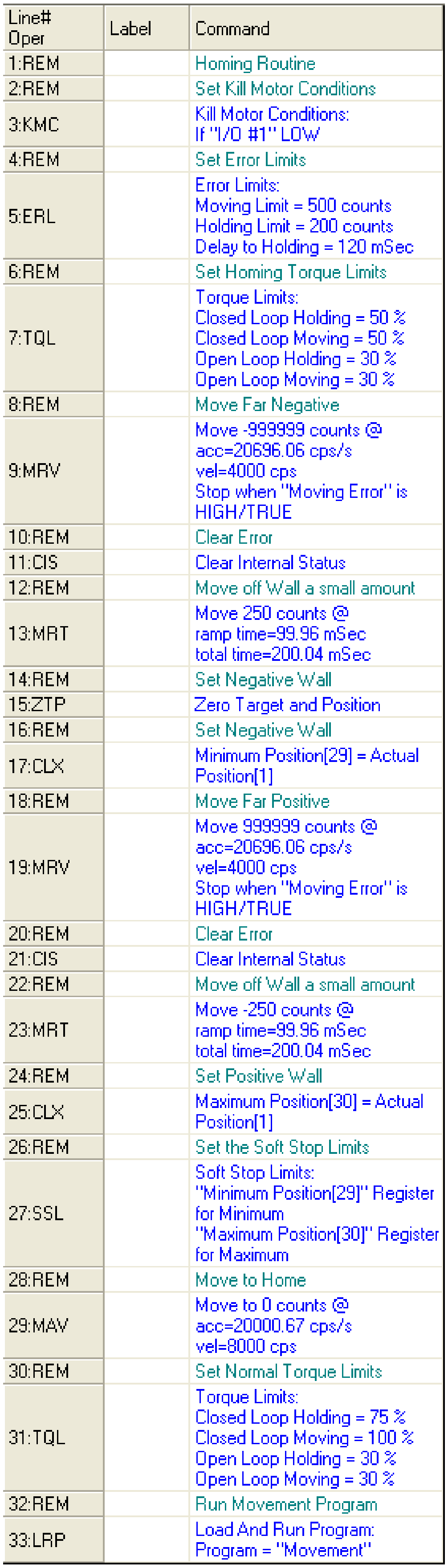
The next move is in a positive direction. This finds the lower limit. This works the same as the far positive move.

The next operation sets the normal Torque Limits then runs the movement program.

Both Slave axes use the homing routine. Line 33 loads either the Slave 1 or 2 Movement program.

Refer to the QCI-AN001_HomingTechniques application note for further explanation.

There is no difference between Slave 1 and Slave 2.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Homing Routine |
| 2:REM | | Set Kill Motor Conditions |
| 3:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW |
| 4:REM | | Set Error Limits |
| 5:ERL | | Error Limits:<br>Moving Limit = 500 counts<br>Holding Limit = 200 counts<br>Delay to Holding = 120 mSec |
| 6:REM | | Set Homing Torque Limits |
| 7:TQL | | Torque Limits:<br>Closed Loop Holding = 50 %<br>Closed Loop Moving = 50 %<br>Open Loop Holding = 30 %<br>Open Loop Moving = 30 % |
| 8:REM | | Move Far Negative |
| 9:MRV | | Move -999999 counts @<br>acc=20696.06 cps/s<br>vel=4000 cps<br>Stop when "Moving Error" is HIGH/TRUE |
| 10:REM | | Clear Error |
| 11:CIS | | Clear Internal Status |
| 12:REM | | Move off Wall a small amount |
| 13:MRT | | Move 250 counts @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 14:REM | | Set Negative Wall |
| 15:ZTP | | Zero Target and Position |
| 16:REM | | Set Negative Wall |
| 17:CLX | | Minimum Position[29] = Actual Position[1] |
| 18:REM | | Move Far Positive |
| 19:MRV | | Move 999999 counts @<br>acc=20696.06 cps/s<br>vel=4000 cps<br>Stop when "Moving Error" is HIGH/TRUE |
| 20:REM | | Clear Error |
| 21:CIS | | Clear Internal Status |
| 22:REM | | Move off Wall a small amount |
| 23:MRT | | Move -250 counts @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 24:REM | | Set Positive Wall |
| 25:CLX | | Maximum Position[30] = Actual Position[1] |
| 26:REM | | Set the Soft Stop Limits |
| 27:SSL | | Soft Stop Limits:<br>"Minimum Position[29]" Register for Minimum<br>"Maximum Position[30]" Register for Maximum |
| 28:REM | | Move to Home |
| 29:MAV | | Move to 0 counts @<br>acc=20000.67 cps/s<br>vel=8000 cps |
| 30:REM | | Set Normal Torque Limits |
| 31:TQL | | Torque Limits:<br>Closed Loop Holding = 75 %<br>Closed Loop Moving = 100 %<br>Open Loop Holding = 30 %<br>Open Loop Moving = 30 % |
| 32:REM | | Run Movement Program |
| 33:LRP | | Load And Run Program:<br>Program = "Movement" |

# Intermediate Pick and Place – Slave 1.qcp and Intermediate Pick and Place – Slave 2.qcp

## Movement

This program will process the movement sequence.

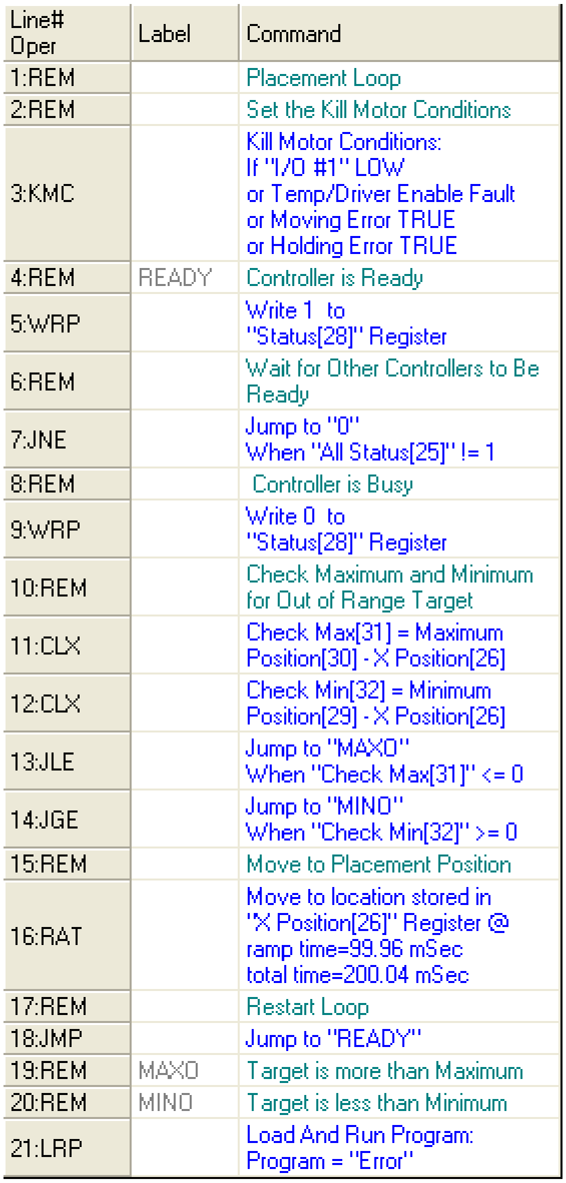
The first command sets the Kill Motor Conditions to the I/O #1, an emergency stop control, Temperature/Driver Enable Fault, Moving Error, or Holding Error.

The next command begins the movement loop and sets the ready status. It then waits for the Master controller to transmit the ready status.

Once this ready status is received, the controller decides if the target position is out of range of motion. The target must be between the minimum and maximum points found in the homing sequence or the controller goes into an error state.

After checking, the motor moves the point transmitted from the Master controller and then the loop restarts.

The difference between Slave 1 and Slave 2 in this program is between Lines 11 and 16. X Position[26] is switched with Y Position[27].



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Placement Loop |
| 2:REM | | Set the Kill Motor Conditions |
| 3:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW<br>or Temp/Driver Enable Fault<br>or Moving Error TRUE<br>or Holding Error TRUE |
| 4:REM | READY | Controller is Ready |
| 5:WRP | | Write 1 to<br>"Status[28]" Register |
| 6:REM | | Wait for Other Controllers to Be Ready |
| 7:JNE | | Jump to "0"<br>When "All Status[25]" != 1 |
| 8:REM | | Controller is Busy |
| 9:WRP | | Write 0 to<br>"Status[28]" Register |
| 10:REM | | Check Maximum and Minimum for Out of Range Target |
| 11:CLX | | Check Max[31] = Maximum Position[30] - X Position[26] |
| 12:CLX | | Check Min[32] = Minimum Position[29] - X Position[26] |
| 13:JLE | | Jump to "MAX0"<br>When "Check Max[31]" <= 0 |
| 14:JGE | | Jump to "MIN0"<br>When "Check Min[32]" >= 0 |
| 15:REM | | Move to Placement Position |
| 16:RAT | | Move to location stored in<br>"X Position[26]" Register @<br>ramp time=99.96 mSec<br>total time=200.04 mSec |
| 17:REM | | Restart Loop |
| 18:JMP | | Jump to "READY" |
| 19:REM | MAX0 | Target is more than Maximum |
| 20:REM | MIN0 | Target is less than Minimum |
| 21:LRP | | Load And Run Program:<br>Program = "Error" |

# Intermediate Pick and Place – Slave 1.qcp and Intermediate Pick and Place – Slave 2.qcp

## Error

This program recovers the controller and motor from a Kill Motor event.

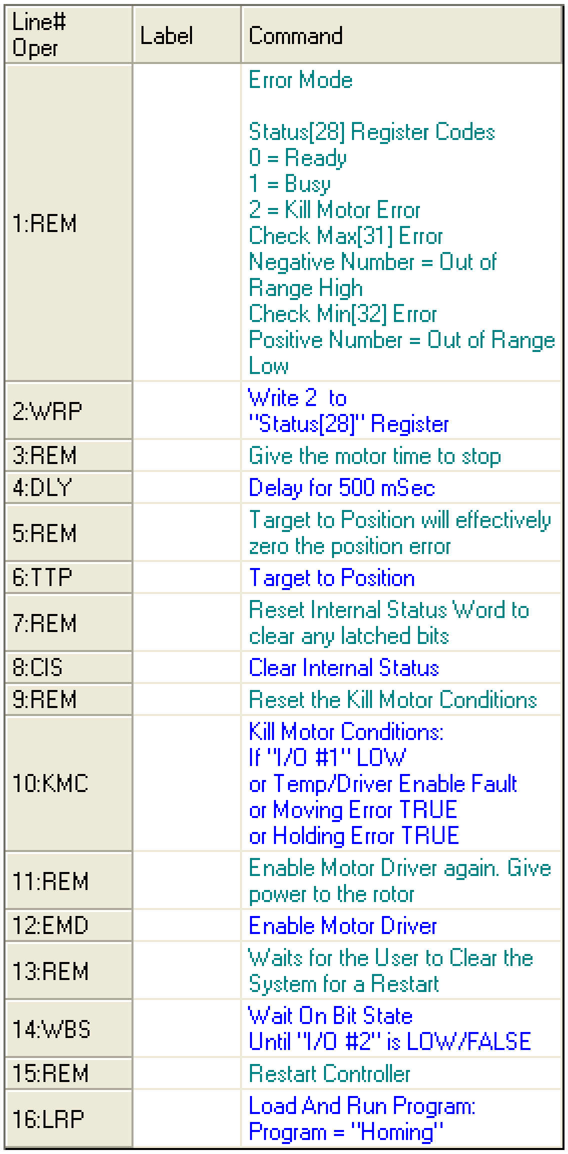
The first command sets the Status to an Error state and then waits for 500 milliseconds before beginning recovery.

The Target to Position command sets the current actual position as the current target position. The motor then Clears Internal Status which clears the Kill Motor error.

The Kill Motor Conditions reset and the Motor Driver re-enables before waiting for an All Clear command from the user.

The last command returns operation at the homing program.

There is no difference between Slave 1 and Slave 2.



| Line#<br>Oper | Label | Command |
|---|---|---|
| 1:REM | | Error Mode<br><br>Status[28] Register Codes<br>0 = Ready<br>1 = Busy<br>2 = Kill Motor Error<br>Check Max[31] Error<br>Negative Number = Out of Range High<br>Check Min[32] Error<br>Positive Number = Out of Range Low |
| 2:WRP | | Write 2 to "Status[28]" Register |
| 3:REM | | Give the motor time to stop |
| 4:DLY | | Delay for 500 mSec |
| 5:REM | | Target to Position will effectively zero the position error |
| 6:TTP | | Target to Position |
| 7:REM | | Reset Internal Status Word to clear any latched bits |
| 8:CIS | | Clear Internal Status |
| 9:REM | | Reset the Kill Motor Conditions |
| 10:KMC | | Kill Motor Conditions:<br>If "I/O #1" LOW<br>or Temp/Driver Enable Fault<br>or Moving Error TRUE<br>or Holding Error TRUE |
| 11:REM | | Enable Motor Driver again. Give power to the rotor |
| 12:EMD | | Enable Motor Driver |
| 13:REM | | Waits for the User to Clear the System for a Restart |
| 14:WBS | | Wait On Bit State<br>Until "I/O #2" is LOW/FALSE |
| 15:REM | | Restart Controller |
| 16:LRP | | Load And Run Program:<br>Program = "Homing" |

CANopen® and CiA® are registered community trade marks of CAN in Automation e.V.