Prototype Controller Manual

Software Versions: controller: proto rev2 interface: proto interface rev4

Changes with Latest Revisions: Updated version string to give information on reason for last reset. Added a 0.25 second delay in take from shelf and take from door routines to allow grabber to close before robot pulls back.

Sections:

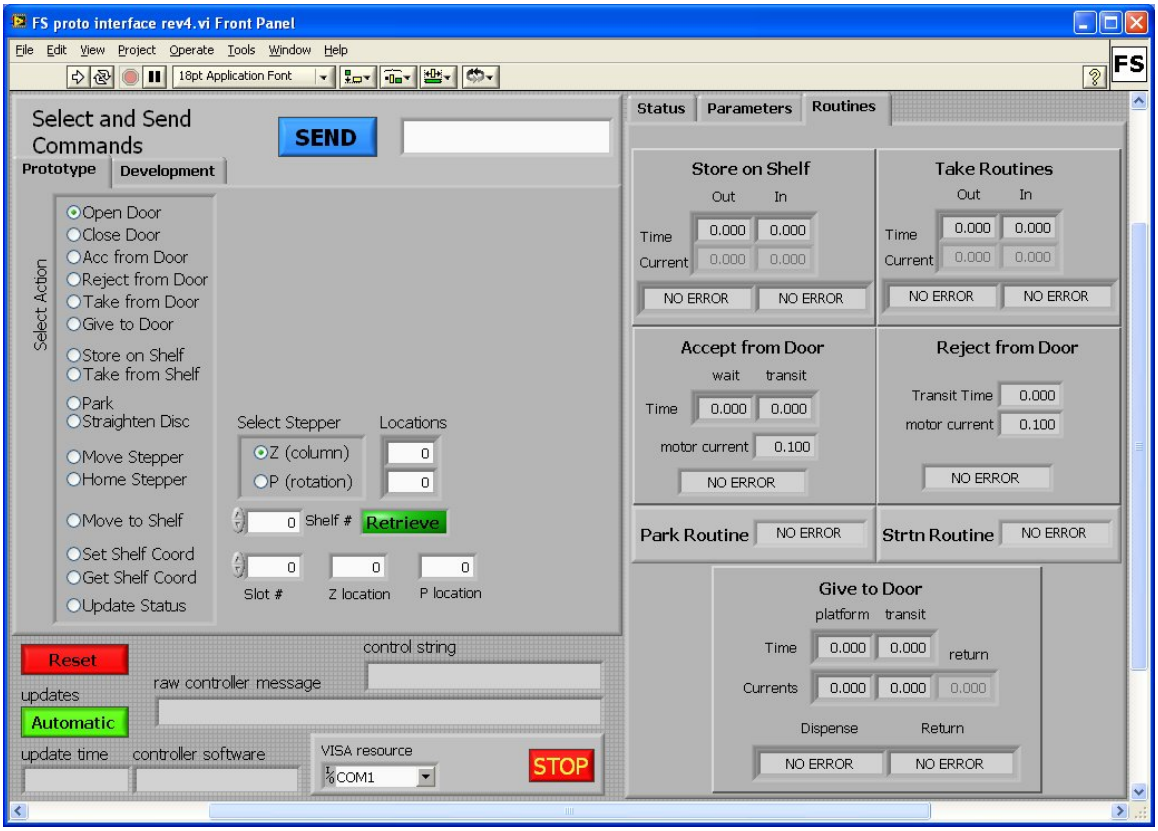
Interface Software:

Installation

To install the software launch the 'setup' program in the 'interface software' folder and follow the instructions on the screen. The installer will place a shortcut to the program on your desktop.

General Operation

The interface program starts running upon launch. When it is running, the arrow in the upper left corner is black.



Screen 1

Stop/Restart: To stop the program, press the red 'STOP' button – the black arrow in the upper left corner of the window will turn to white. To restart the program, click the white arrow. When you stop the program, it saves the parameters listed on the 'Parameters' page (see below) to a text file.

Reset Button: This resets the controller. This is the fastest way to stop a stepper, motor or solenoid if something is going wrong. The first thing the controller software does when it resets is put all outputs in the off state. However, you will lose the stepper locations since they are reset to zero upon reset.

updates Button: When updates is set to 'Automatic' the interface sends an update command to the controller once every second as long as other commands are not being sent or being executed. This is helpful for checking the operation of the switches. Click the button to 'disable' to disable the updates. Disabling the updates keeps the 'control string' and 'raw controller message' fields from being overwritten.

update time: This displays the last time an update response was received from the controller.

controller software: This displays the software version message sent from the controller at startup/reset.

VISA resource: This is the com port the program uses to communicate with the controller. The default is COM1. If you are connected using a different com port, you will need to select it in this field and then stop and restart the program.

control string: This is the string sent from the interface to the controller.

raw controller message: This is the last string received from the controller.

Sending Commands

To send a command, you first setup the command then you click the 'SEND' button. To setup a command, first you select an action by clicking on the circle next to the action desired, then if necessary you select a stepper or enter data for the command. If the 'SEND' button is not visible, then a command has been sent and the interface is waiting to receive the 'ready' message from the controller. See below for a description of the data returned by the controller and for detailed routine descriptions. The commands are separated into two screens 'Prototype' and 'Development'. Move between the two sets of commands by clicking on the tabs.

Prototype Commands (see Screen 1 above):

Open Door: No data required. This opens the interface door. The door will remain open until a close door command is sent or until the controller hasn't received any command in 10 seconds. If you leave the door open too long, the solenoid will get hot.

Close Door: No data required. This closes the interface door.

Acc from Door: No data required. This starts the routine that accepts media into the interface door.

Reject from Door: No data required. This starts the routine that moves media from the transfer area out to the customer.

Take from Door: No data required. This starts the routine that moves the media from the interface door area onto the robot.

Give to Door: No data required. This starts the routine that moves the media from the robot onto the interface door.

Store on Shelf: No data required. This starts the routine that moves media from the robot onto a storage shelf.

Take from Shelf: No data required. This starts the routine that takes media from a slot and places it on the robot.

Park: No data required. This moves the grabber to the park or ready position.

Straighten Disc: No data required. This starts the routine that moves a disc from the full back position to the ready position and straightens it on the robot during the move.

Move Stepper: Select the stepper to move and enter the location to move to. All locations are in stepper steps from home. This moves the stepper indicated to the location entered.

Home Stepper: Select the stepper to home. This sends the stepper to trip the home switch.

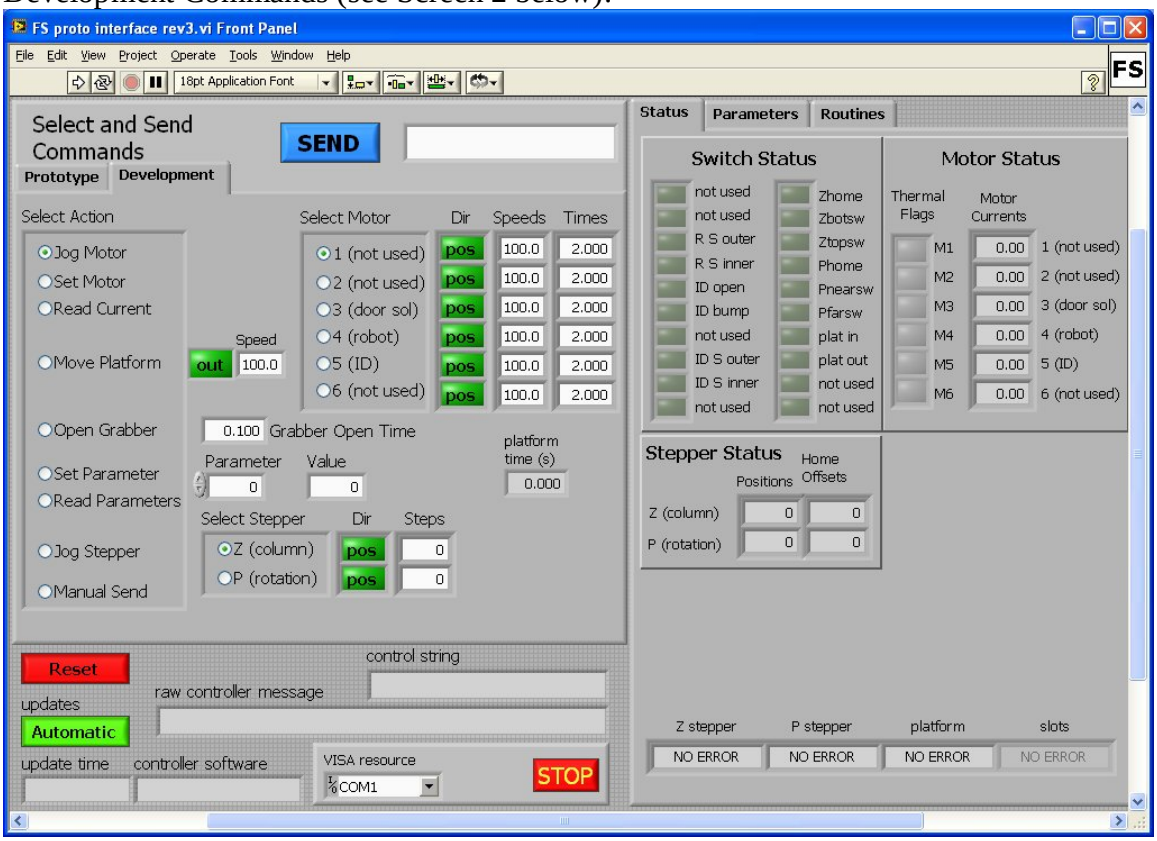
Move to Shelf: Enter a slot number and indicate 'Retrieve' or 'Store' position with the button next to the slot number. This will send the robot to the slot indicated in a position to retrieve or store media. Valid slot coordinates must have already been set for the slot entered.

Set Shelf Coord: Enter the shelf number (not the same field as for Move to Shelf), and the Z location and P location in the fields to the right. This commands stores the coordinates in the EEprom for use by the Move to Shelf command.

Get Slot Coord: Enter the shelf number. This command returns the coordinates in the same fields used to send slot coordinates.

Update Status: No data required. This command requests the controller return status information.

Development Commands (see Screen 2 below):



Screen 2

Jog Motor: Select a motor and direction and enter the speed and time. The speed is in per cent of full speed and the time is in seconds. The maximum for time is 65.535 seconds. This command activates the indicated motor (or solenoid) in the indicated polarity at the speed/voltage entered for the amount of time entered. CAUTION: this command does not use any switches to turn the motor off nor does it check the motor current for over current/jam conditions. If you jog the platform motor (4) and it reaches the end of its travel, you could strip the gears. Use this command carefully – if the platform does jam during a jog command, you can stop the motor by clicking the 'Reset' button.

Set Motor: This command works the same way as Jog Motor except that it turns the motor on and leaves it on until you set the speed to 0. The same cautions as described above apply. An additional precaution should be observed with this command: Always turn a motor to off (speed 0 or send a jog command to stop) before you change the direction of the motor.

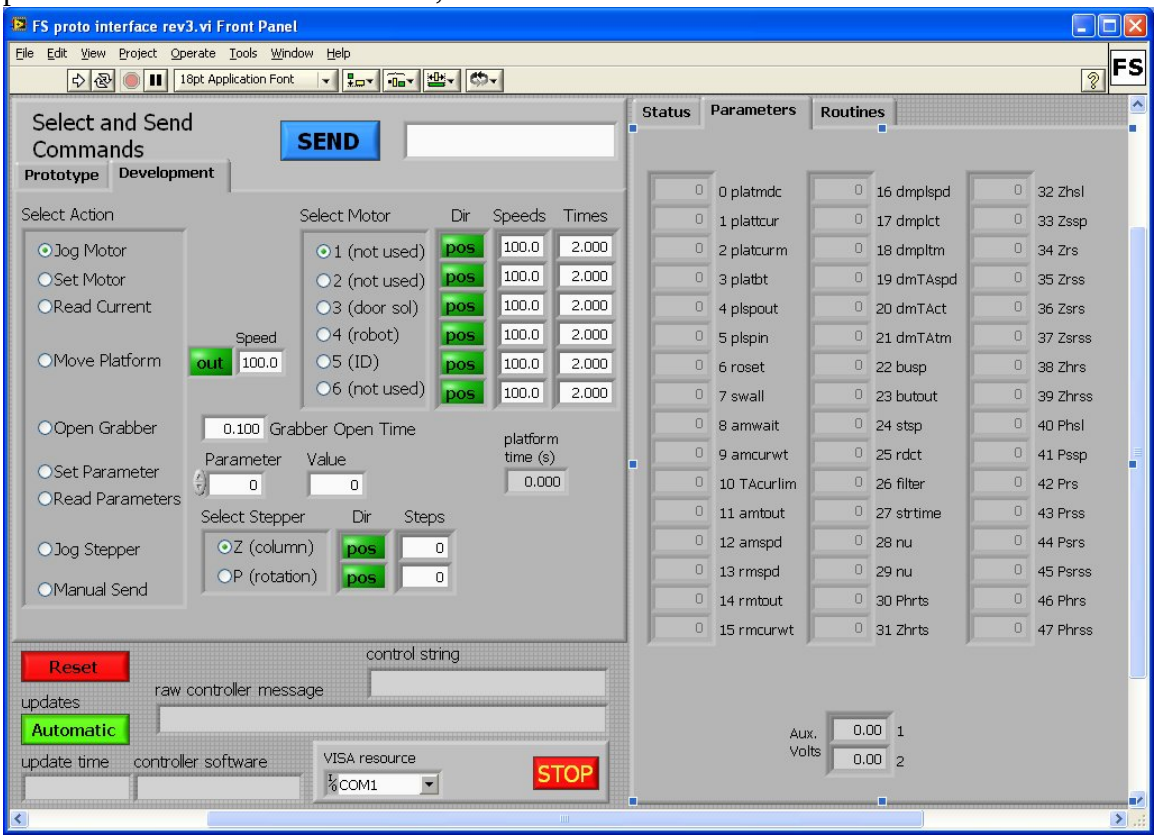
Read Current: Select motor. This returns the current reading from the driver chip of the motor selected.

Move Platform: Select the direction and set the speed. This command moves the robot platform in the direction indicated until the limit switch is tripped. The motor current is checked during the move and returned and the time of the move is recorded and returned. If an error occurs during the move, it is displayed in the platform field at the bottom of the Status tab. The current is displayed in the Motor Status area and the transit time is displayed in the lower right area of the 'Development' tab. The timeout limit and motor current limit are set using the parameters.

Open Grabber: Enter the amount of time in seconds to activate the sidewall solenoid in the 'Grabber Open Time' field. This command activates the grabber solenoid for the amount of time indicated. The maximum time is 10 seconds.

Set Parameter: Enter the parameter number and value. This command writes the parameter value into the EEprom for use in routines.

Read Parameters: No data required. This command returns all the parameters stored in the EEprom and displays them on the 'Parameters'. For a full list of the parameters and their initial values, see the Parameters Table below.



Screen 3

Jog Stepper: Select the stepper and direction and enter the number of steps to move. This command moves the stepper selected in the direction indicated for the number of steps entered. This move, like all stepper moves, is stopped if a limit switch is tripped.

Manual Send: Enter the string to send to the controller in the field next to the 'SEND' button. Do not include the "!" character at the start of the string, or the "x" character at the end of the string. The interface will add those for you. This command sends the string entered with the start and end characters to the controller. This might be used if functionality is added to the controller without updating the interface.

Display of Data Received from the Controller

Routines (see Screen 1 above):
The times, currents and errors recorded during the Store on Shelf, Take (both from door and shelf), Accept from Door, Reject from Door, Park, Straighten, and Give to Door routines are displayed on the 'Routines' tab.

Status (see Screen 2 above):
The status of the switches as received from an update command return is displayed in the Switch Status box. The stepper locations from the update data is displayed in the Stepper Status box. The status of the thermal flags from the update data is displayed in the Motor Status box. When a stepper is homed, the number of steps different from the last home routine is displayed in the Stepper Status Box (the first homes after power up or reset will return meaningless offsets). The currents read during a Read Current command or during a Move Platform command are displayed in the Motor Status box.

Parameters (see Screen 3 above):
When a Read Parameters command return is received the parameters are displayed on this tab,

Parameter Table

Parameter Description	abbreviation	#	Value	units	
platform move duration counter	platmdc	0	25	outerloops	
platform move time before current read	plattcur	1	200	ms	
platform jam current	platcurm	2	100	adc counts	
platform brake time	platbt	3	200	ms	
platform speed -- out	plspout	4	255		
platform speed -- in	plspin	5	255		
robot Z offset	roset	6	???	steps	
sidewall solenoid fire time	swall	7	250	ms	???? not used?
wait for customer in accept media	amwait	8	5000	ms	
accept media current read time	amcurwt	9	500	ms	
transfer motor current limit	TAcurlim	10	100	adc counts	
accept media timeout	amtout	11	1500	ms	
accept media motor speed	amspd	12	255		
reject media motor speed	rmspd	13	255		
reject media timeout	rmtout	14	1500	ms	
reject media current read	rmcurwt	15	500	ms	

time					
dispense platform speed	dmplspd	16	255		
dispense platform current time	dmplct	17	500	ms	
dispense platform move time	dmpltm	18	10000	ms	
dispense transfer speed	dmTAspd	19	255		
dispense transfer current time	dmTAct	20	500	ms	
dispense transfer time	dmTAtm	21	2000	ms	
backup speed	busp	22	255		
backup timeout	butout	23		ms	
straightening speed	stsp	24	255		
retrieve disc check time	rdct	25	500	ms	
optical sensor filter setting	filter	26	5	readings	
straighten timeout	strtm	27	500	ms	
platform switch filter	Pfilter	28	5	readings	also used w/ bump switch
not used		29			
P home return steps	Phrts	30	800	steps	number of steps moved back after home
Z home return steps	Zhrts	31	400	steps	number of steps moved back after home
Z home step limit	Zhsl	32	1000	number	limit during slow phase
Z start step period	Zssp	33	2482	usec	
Z ramp steps	Zrs	34	600	number	
Z ramp step size	Zrss	35	3	usec	
Z stop ramp steps	Zsrs	36	100	number	
Z stop ramp step size	Zsrss	37	25	usec	
Z home ramp steps	Zhrs	38	200	number	for start and slowdown ramp
Z home ramp step size	Zhrss	39	4	usec	for slowdown ramp
P home step limit	Phsl	40	1000	number	limit during slow phase
P start step period	Pssp	41	2482	usec	
P ramp steps	Prs	42	600	number	
P ramp step size	Prss	43	3	usec	
P stop ramp steps	Psrs	44	100	number	
P stop ramp step size	Psrss	45	25	usec	
P home ramp steps	Phrs	46	100	number	for start and slowdown ramp
P home ramp step size	Phrss	47	4	usec	for slowdown ramp

Controller Software:

The commands below are summarized in a table at the end of this section.

Controller Software Commands

Format used in command descriptions:

Values in () are variables sent to the controller or sent by the controller to the host.

If the value is preceded by "dec" it is a decimal number sent as ASCII characters.

If the value is not preceded by "dec" it is a single ASCII character. In all current cases it is a direction symbol (dir), either "+" for positive or "-" for negative, or "B" for brake in the set motor command. If a character other than these two are sent the direction defaults to positive.

All other characters are literal. The commands start with "!" followed by a identifier character, variables are separated by commas, and the commands end with "x".

Command: open door

String sent: !Ox

String returned: none

Description: This command activates the transfer area door solenoid.

Command: close door

String sent: !Cx

String returned: none

Description: This command deactivates the transfer area door solenoid.

Command: accept from door

String sent: !Ax

String returned: #A,(dec wait_time_(ms)),(dec transfer_time_(ms)),(dec current),(dec error)y

Description: This command initiates a retrieve routine to accept media from a customer and place it in the transfer area. The times returned are in milliseconds; the current value returned is in the same format as a 'read current' command. The applicable errors for this command are: 0 (no error), 4 (time out), and 5 (motor overcurrent). If the command timesout the controller checks the motor current a second time. If the current is above the limit, the overcurrent error is returned instead of the timeout error. The sequence of events initiated by this command is:

1) wait for outer transfer area switch to trip
2) start transfer area motor in a negative (inward) direction
3) wait for the bump switch to trip
4) turn off transfer area motor

If a timeout error is received and the transfer time = 0 then the routine timed out waiting for the first switch to trip. If transfer time > 0 then the timeout happened while trying to move the media into the transfer area.

Command: reject from door

String sent: !Rx

String returned: #R,(dec transfer_time_(ms)),(dec current),(dec error)y

Description: This command initiates a reject routine to reject media from the transfer area out to the customer. The time returned is in milliseconds; the current value returned is in the same format as a 'read current' command. The applicable errors for this command are: 0 (no error), 4 (time out), and 5 (motor overcurrent). If the command timesout the controller checks the motor current a second time. If the current is above the limit, the overcurrent error is returned instead of the timeout error. The sequence of events initiated by this command is:

1) start transfer area motor in a positive (outward) direction
2) wait 500 ms.
3) wait for the inner transfer area switch to un-trip
4) turn off transfer area motor

Command: take from door

String sent: !Fx

String returned: #F,(dec coarse_time_out),(dec fine_time_out),(dec current_out),(dec error#out),(dec coarse_time_in),(dec fine_time_in),(dec current_in),(dec error#in)y

Description: This command initiates a retrieve routine to retrieve media from the interface door and place it on the robot. The grabber must be in the park position to start this routine (you will get an error of 10 if not in park position) – if necessary, do a park command before starting this routine. The data returned consists of two sets – one set for the motion out and one for the motion back in. Each set of data returned is the same as that returned by a 'move platform' command and the timing should be evaluated in the same way. The applicable errors for this command are: 0 (no error), 4 (time out), 5 (motor overcurrent), 10 (grabber not in start position), 11 (grabber edge not seen during return), and 13 (disc not in grabber). The sequence of events initiated by this command is:

1) open grabber
2) move platform out to outer limit switch
3) close grabber
4) pause 250ms
5) move platform in to inner limit switch
    a. start ID belts moving to help disc off ID
    b. check for grabber using sensors
    c. then check for disc
    d. if no disc, stop, pulse grabber (to drop disc if too high)

e. check for disc again – if no disc, stop, if disc continue to inner limit switch

Command: give to door

String sent: !Dx

String returned: #D,(dec platform_time_(ms)),(dec transit_time_(ms)),(dec platform_current),(dec transfer_current),(dec transfer_error),(dec platform_return_current),(dec return_error)y

Description: This command initiates a dispense routine to move media from the robot to the transfer area. The disc must be straightened before it can be given to the door, otherwise you will get an error of 10 – send a straighten command before using this routine. The times returned are in milliseconds; the current values returned are in the same format as a 'read current' command. The applicable errors for this command are: 0 (no error), 1 (limit switch), 4 (time out), 5 (motor overcurrent), and 10 (disc not in ready position). If the command timesout the controller checks the motor current a second time. If the current is above the limit, the overcurrent error is returned instead of the timeout error. The sequence of events initiated by this command is:

1) start platform motor in a positive (outward) direction
2) wait for the inner ID sensor to trip
3) turn off robot
4) open grabber
5) turn on ID motor in a positive (outward) direction
6) wait for outer ID sensor to trip
7) turn off ID motor
8) close grabber
9) perform a park move
10) turn off platform motor

A limit switch error means that the platform tripped its outer switch before the inner ID sensor tripped. If an error occurs after the ID motor is turned on, the robot is returned to the full in position instead of the park/ready position.

Command: store on shelf

String sent: !Sx

String returned: #S(dec coarse_time_out),(dec fine_time_out),(dec current_out),(dec error#out),(dec time_in),(dec error#in)y

Description: This command initiates a store routine to place media on the robot in a slot and then retract the robot. The disc must be straightened before it can be stored, otherwise you will get an error of 10 – send a straighten command before doing a store routine. The robot starts forward and checks the motor current. If the current is above the limit, it reverses all the way back to the back limit. If the current is acceptable, it continues out until the front platform switch is tripped. It then opens the grabber, the

grabber returns to the park position and the grabber is closed. The data returned for the motion out is the same as that returned by a 'move platform' command and should be evaluated the same way. The data returned for the motion in is time and error code, to convert the time value returned to 'nominal' seconds divide by 10000.

Command: take from shelf

String sent: !Gx

String returned: #G(dec coarse_time_out),(dec fine_time_out),(dec current_out),(dec error#out),(dec coarse_time_in),(dec fine_time_in),(dec current_in),(dec error#in)y

Description: This command initiates a retrieve routine to retrieve media from a slot and place it on the robot. The grabber must be in the park position to start this routine (you will get an error of 10 if not in park position) – if necessary, do a park command before starting this routine. The data returned consists of two sets – one set for the motion out and one for the motion back in. Each set of data returned is the same as that returned by a 'move platform' command and the timing should be evaluated in the same way. The applicable errors for this command are: 0 (no error), 4 (time out), 5 (motor overcurrent), 10 (grabber not in start position), 11 (grabber edge not seen during return), and 13 (disc not in grabber). The sequence of events initiated by this command is:

1) open grabber
2) move platform out to outer limit switch
3) close grabber
4) pause 250ms
5) move platform in to inner limit switch
    a. check for grabber using sensors
    b. then check for disc
    c. if no disc, stop, pulse grabber (to drop disc if too high)
    d. check for disc again – if no disc, stop, if disc continue to inner limit switch

Command: park

String sent: !Bx

String returned: #B,(dec error)y

Description: This routine first checks the robot sensors. If both are tripped, it moves the robot in until the outer sensor un-trips, if the outer is not tripped, it moves the robot out until the inner sensor trips The applicable errors for this command are: 0 (no error), 1 (limit switch - platform), 4 (time out), and 5 (motor overcurrent).

Command: straighten disc

String sent: !Ex

String returned: #B,(dec error)y

Description:

1) checks that robot in full in position
2) starts robot forward (out)
3) opens grabber
4) stops robot when inner robot sensor trips (front edge of case/disc) or when timeout set by parameter expires.
5) closes grabber
6) initiates a park move to make sure the disc/case is in the ready position

The applicable errors for this command are: 0 (no error), 1 (limit switch - platform), 4 (time out), 5 (motor overcurrent), 9 (platform not in back position for start of routine), 14 (park to – parkmove timeout in step 6), 15 (park current – park move overcurrent in step 6), and 16 (park limit – limit switch error during park move in step 6). By setting the timeout parameter (#27) to a low value (500) a timeout error tells you there is not a disc on the robot.

Command: move stepper

String sent: !J,(dec stepper#),(dec location)x

String returned: #J(dec error#)y

Description: This command moves the stepper indicated to the location (stepper counts) directed. The Z stepper is 0 and the P stepper is 1. If a number other than 0 or 1 is given, it defaults to the Z stepper. During a P move or a positive Z move, if a limit switch is hit, the stepper is ramped down to a controlled stop and then reversed back into the operating range of the axis; if the home switch is hit on a P move, the stepper is stopped immediately and then reversed back into the operation range of the axis. During a negative Z move, if the Z home switch is hit, the stepper is stopped immediately and an error code of 12 is returned (in this case the stepper is not reversed). The applicable error codes for this command are: 0 (no error), 1 (limit or home switch stop), 2 (position underflow – position < 0), 3 (position overflow – position > 65,535), and 12 (Z home switch hit during a negative Z move).

Command: home stepper

String sent: !H,(dec stepper#)x

String returned: #H(dec offset_from_last_home),(dec error#)y

Description: This command homes the indicated stepper. Both steppers are homed in the negative direction. The Z stepper is 0 and the P stepper is 1. If a number other than 0 or 1 is given, it defaults to the Z stepper. After the home switch is hit, the offset_from_last_home is calculated, the stepper count is reset to zero and the stepper is reversed back beyond the limit switch nearest the home switch. The offset_from_last_home is the difference in steps between the current home and the last home for that axis. The applicable errors for this command are: 0 (no error), 1 (limit switch – home switch found before the limit switch), and 4 (time out – switch not found).

Command: move to shelf

String sent: !M(type),(dec shelf#)x

String returned: #M(dec error#)y

Description: This command moves the steppers to the coordinates/locations of the shelf indicated. If type = S (store) the Z coordinate read from the EEprom is used, if type = R (retrieve) the robot offset is subtracted from the Z coordinate. The shelf coordinates must first be saved in the EEprom (see below). If no coordinates have been set, the EEprom returns 65535 which is outside the software limits and an error is returned. Before reading the coordinates from the EEprom, the controller checks that the slot number is valid. It also checks that the coordinates are within the machine limits. During a P move or a positive Z move, if a limit switch is hit, the stepper is ramped down to a controlled stop and then reversed back into the operating range of the axis; if the home switch is hit on a P move, the stepper is stopped immediately and then reversed back into the operation range of the axis. During a negative Z move, if the Z home switch is hit, the stepper is stopped immediately and an error code of 12 is returned (in this case the stepper is not reversed). The applicable error codes for this command are: 0 (no error), 1 (limit or home switch stop), 2 (position underflow – position < 0), 3 (position overflow – position > 65,535), 6 (shelf coordinate invalid), 7 (shelf number out of range), and 12 (Z home switch hit during a negative Z move).

Command: set shelf coordinates

String sent: !U,(dec shelf#),(dec Zlocation),(dec Plocation)x

String returned: #U(dec error#)y

Description: This command sets the shelf coordinates to the values indicated. Before writing the coordinates to the EEprom, the controller checks that the shelf number is valid. It also checks that the coordinates are within the machine limits. The Z coordinate stored should be the Z position for placing a disc into a slot. The applicable error codes for this command are: 0 (no error), 6 (shelf coordinate invalid), and 7 (shelf number out of range).

Command: get shelf coordinates

String sent: !V,(dec shelf#)x

String returned: #V,(dec shelf#),(dec Zcoordinate),(dec Pcoordinate)y

Description: This command returns the coordinates for the Store position of the shelf.

Command: update status

String sent: !Px (this is also the command executed if the command identifier sent is not assigned to another command)

String returned: #P(dec status1),(dec status2),(dec Z_stepper_location),(P_stepper_location)y

Description: This command returns four numbers to the host. The first two numbers indicate the status of all the switches and the last two give the locations of the Z axis and P axis in stepper counts. Once the switch status numbers are converted to binary each bit indicates the status of a switch or a driver chip thermal flag – a 1 indicates that the switch is contacted or a driver chip is starting to overheat , a 0 indicates that the switch is not contacted or that a driver chip is not overheating. The driver chips switch the thermal flag at ~145C and shut the outputs off at ~170C.

status1 bit assignments:

Bit 0 (LSB)	not used
Bit 1	not used
Bit 2	robot outer sensor
Bit 3	robot inner sensor
Bit 4	interface door open switch
Bit 5	interface door bump switch
Bit 6	not used
Bit 7	interface door outer sensor
Bit 8	interface door inner sensor
Bit 9	not used
Bit 10	driver chip 1 thermal flag (chip not currently used)
Bit 11	driver chip 2 thermal flag (chip not currently used)
Bit 12	driver chip 3 thermal flag (drives door solenoid)
Bit 13	driver chip 4 thermal flag (drives robot motor)
Bit 14	driver chip 5 thermal flag (drives interface door motor)
Bit 15	driver chip 6 thermal flag (chip not currently used)

status2 bit assignments:

Bit 0 (LSB)	Z axis home switch
Bit 1	Z axis bottom limit switch
Bit 2	Z axis top limit switch
Bit 3	P axis home switch
Bit 4	P axis limit switch nearest to home switch
Bit 5	P axis limit switch farthest from home switch
Bit 6	platform in switch
Bit 7	platform out switch
Bit 8	not used
Bit 9	not used

Command: jog motor

String sent: !j,(dec motor#),(dir),(dec speed),(dec time(ms))x

String returned: none

Description: This command turns a motor on in the direction indicated at the speed directed for the number of milliseconds directed. This is also used to activate the grabber solenoid which is designated as motor 7. Activating the solenoid opens the grabber. When jogging the grabber a speed value of 255

turns the solenoid on while any other value turns it off. The speed of a motor can be any value between 0 (off) and 255 (full on). Motor speed is set by bank; bank 1 is motors 1, 2, and 3 and bank 2 is motors 4, 5, and 6. The motors are enabled and the directions are set independently; however, the speed of all enabled motors in a bank will be the same. The time can be any value between 0 and 65535. The term 'motor' is generic and can also mean a solenoid.

Motor number assignments:

Motor 1	not used
Motor 2	not used
Motor 3	door solenoid
Motor 4	robot motor
Motor 5	transfer area motor
Motor 6	not used
Motor 7	grabber solenoid

Command: set motor

String sent: !o,(dec motor#),(dir),(dec speed)x

String returned: none

Description: This command turns a motor on in the direction indicated at the speed directed. See the jog motor command for details on motor banks, motor speeds, and the sidewall solenoid. To turn a motor off the speed needs to be set to 0. A motor should be turned off before its direction is reversed to reduce stress on the driver chip. If dir is "B" the brakes are set on the motor – this shorts the motor leads and has a braking effect where as setting the speed to 0 just turns the power off and allows the motor to coast to a stop.

Command: read current

String sent: !c,(dec motor#)x

String returned: #c(dec motor#),(dec reading)y

Description: This command reads the current signal coming from the motor driver chip for the motor indicated. The reading is a value between 0 and 255 with 255 being a nominal drive current of 2.5 amps.

Command: move platform

String sent: !f(dir),(dec speed)x

String returned: #f(dec coarse_time),(dec fine_time),(dec current),(dec error#)y

Description: This command moves the platform in (dir = "-") or out (dir = "+") at the speed directed. The controller monitors for jams by reading the motor current during the move (and at the end if the move timesout). When the applicable limit switch is activated, the motor brakes are applied. If the switch isn't tripped in a set amount of time, the motor is stopped and the routine aborted. To convert the coarse_time and fine_time numbers to 'nominal' seconds use the formula: seconds = 0.2 + (((coarse_time *

65535) + fine_time) * 6.5E-6). The current returned is in the same format as the read current command. The applicable errors for this command are: 0 (no error), 4 (time out), and 5 (motor overcurrent). If the command timesout the controller checks the motor current a second time. If the current is above the limit, the overcurrent error is returned instead of the timeout error.

Command: open grabber
String sent: see jog motor above
String returned: none
Description: This is a special case of the jog motor command.

Command: set parameter
String sent: !p,(dec parameter#),(dec value)x
String returned: none
Description: This command sets the parameter indicated to the value given. Currently there is space for 48 parameters (0 thru 47). These parameters are used for things like stepper speeds and ramps and some or all of them will be 'hard coded' in the production software.

Command: read parameters
String sent: !q,(dec parameter#)x
String returned: #q,(48 dec values separated by ",")y
Description: This command returns the values of the 48 parameters. These parameters are used for things like stepper speeds and ramps and some or all of them may be 'hard coded' in the production software.

Command: jog stepper
String sent: !s,(dec stepper#),(dir),(dec steps)x
String returned: #s(dec error#)y
Description: This command moves the stepper indicated the directed number of steps in the directed direction. The Z stepper is 0 and the P stepper is 1. If a number other than 0 or 1 is given, it defaults to the Z stepper. During a P move or a positive Z move, if a limit switch is hit, the stepper is ramped down to a controlled stop and then reversed back into the operating range of the axis; if the home switch is hit on a P move, the stepper is stopped immediately and then reversed back into the operation range of the axis. During a negative Z move, if the Z home switch is hit, the stepper is stopped immediately and an error code of 12 is returned (in this case the stepper is not reversed). The applicable error codes for this command are: 0 (no error), 1 (limit or home switch stop), 2 (position underflow – position < 0), 3 (position overflow – position > 65,535), and 12 (Z home switch hit during a negative Z move).

Command: manual send
String sent: !(contents of field next to send button)x
String returned: none

Description: This command is used to send the contents of the field next to the send button to the controller. You do not need to include the "!" at the start of the string or the "x" at the end, the interface will add those characters for you.

Development Software Messages

Message: ready for a command
String sent: none
String returned: #Ry
Description: The controller sends this message when it is ready to receive a command, either at startup or after it has completed a command.

Message: controller software version
String sent: none
String returned: #(string)y
Description: The controller sends this message at startup. The string is something like "proto rev1W ----- " and identifies the software version the controller is running. After the version there are five characters that identify the cause of the last reset. The characters and types of resets are:

1) all '-''s: normal power up (also has been seen if the controller resets from a direct static zap)
2) B: power brown out reset
3) M: master clear button (or reset from PC) reset
4) W: watchdog timer reset
5) F: stack pointer overflow
6) U: stack underflow

Development Controller Reset from Serial Port

The controller board has a controller reset circuit that is controlled by the host's serial port. This has been useful for when something goes wrong and steppers or motors need to be turned off quickly. The reset restarts the controller software and one of the first things the software does is set the brakes on all motors. The reset works by using the DTR line (pin 4) of the RS232 port. For the controller to run, the DTR line needs to be asserted (set to +12 volts). To reset the controller, unassert the DTR line for ~100ms and then reassert it.

Description	String Sent	String Returned	
Ready	----- none ------	#ry	
update	!Px	#P(dec shiftin),(dec directin),(Zpos),(Ppos),y	
jog motor	!j,(dec motor#),(dir),(dec speed),(dec time(ms))x	none	
set motor	!o,(dec motor#),(dir),(dec speed)x	none	
read current	!c,(dec motor#)x	#c(dec motor#),(dec reading)y	
take from door	!Fx	#F(dec coarse time out),(dec fine time out),(dec current out),(dec error#out),(dec coarse time in),(dec fine time in),(dec current in),(dec error#in),y	
set parameter	!p,(dec param),(dec value)x	none	
read parameters	!qx	#q{48 dec values separated by ","}y	
move platform	!f(dir),(dec speed)x	#f(dec coarse time),(dec fine time),(dec current),(dec error#)y	
jog stepper	!s,(dec stepper#),(dir),(dec steps)x	#s(dec error#)y	
move stepper to location	!J,(dec stepper#),(dec location)x	#J(dec error#)y	
home stepper	!H,(dec stepper#)x	#H(dec offset from last home),(dec error#)y	
get software version	---- none ---- sent on startup/reset	#v(string software version)y	
set shelf coordinates	!U,(dec slot#),(dec Zpos),(dec Ppos)x	#U(dec error#)y	
move to shelf	!M(type),(dec slot#)x	#M(dec error#)y	type = R for retrieve and S for store
open door	!Ox	none	
close door	!Cx	none	

get shelf coordinates	!V,(dec slot#)x	#V(dec slot#),(dec Zpos),(dec Ppos)y
store on shelf	!Sx	#S(dec coarse time out),(dec fine time out),(dec current out),(dec error#out),(dec time in),(dec error#in)y
take from shelf	!Gx	#G(dec coarse time out),(dec fine time out),(dec current out),(dec error#out),(dec coarse time in),(dec fine time in),(dec current in),(dec error#in),y
accept from door	!Ax	#A(dec wait time (ms)),(dec transfer time (ms)),(dec current),(dec error)y
reject from door	!Rx	#R(dec transit time (ms)),(dec current),(dec error)y
give to door	!Dx	#D(dec plat time (ms)),(dec transit time (ms)),(dec plat out current),(dec transfer current),(dec transfer error),(dec return current),(dec return error)y
park	!Bx	#B(dec error)y
straighten disc	!Ex	#E(dec error)y

Assignments:		Assignments:	
platform motor	M4	Z stepper	#0 (default)
grabber solenoid	M7 (on = speed of 255)	P stepper	#1
door solenoid	M3		
transfer area motor	M5		

Error #	Meaning
0	no error
1	limit switch
2	stepper position underflow
3	stepper position overflow
4	timeout
5	motor overcurrent
6	slot parameters invalid (location > max)
7	slot out of range (slot # > max)
8	backup timeout – not in use
9	not back
10	grabber not in ready position
11	grabber edge not detected on retrieve
12	Z home switch hit during a non-homing move
13	disc not seen on retrieve
14	park move timed out during straighten routine
15	park move over current during straighten routine
16	park move limit switch trip during straighten routine

	1	2	3	4	5	6	7	9	10	11	12	13
Routine	limit switch	stepper under-flow	stepper over-flow	time out	motor over-current	invalid slot parameter	slot out of range	platform not back	grabber not in ready position	grabber edge not seen on retrieve	Z home switch hit during a move	disc not in grabber
move platform												
jog stepper												
move stepper to location												
home stepper												
set shelf positions												
move to shelf												
store on shelf or give to door (out motion)												
store on shelf or give to door (in motion)												

take from door or shelf (out motion)												
take from door or shelf (in motion)												
accept from door												
reject from door												
give to door (out motion)												
give to door (in motion)												
park												
straighten disc												

Additional error codes for ‘straighten disc’:

#14: park move timeout at end of straighten routine

#15: park move overcurrent at end of straighten routine

#16: park move limit switch error at end of straighten routine

Detailed Error Descriptions:

Accept from Door:

4 (timeout)

1) current and transfer time = 0, wait time ≠ 0: indicates that the outer ID sensor didn't trip in the timeout period defined by parameter 8. Wait time will = (parameter 8) + 1
2) transfer time ≠ 0: indicates that the ID sensors did not see the trailing edge of the disc in the timeout period defined by parameter 11. The order of sensors after the outer ID sensor is tripped and the motor is turned on is supposed to be: ID outer sees the hole (untrips), ID outer sees the other edge of the hole (trips), ID outer sees the trailing edge of disc (untrips). Transfer time will = (parameter 11) + 1. Current is the motor current at the end of the timeout period.

5 (motor overcurrent)

1) transfer time = 0, wait time ≠ 0: indicates that the outer ID sensor did trip in the timeout period defined by parameter 8. However, when the controller checked the current on the ID motor at the start of the transfer (time after turn-on defined by parameter 9), it exceeded the limit defined by parameter 10. The current returned is the current read at the start of the transfer. Motor was turned off and the routine was stopped at this point.
2) transfer time ≠ 0: indicates that the ID sensors did not see the trailing edge of the disc in the timeout period defined by parameter 11 and when the motor current was checked at the end of the timeout period it exceeded the limit defined by parameter 10. Transfer time will = (parameter 11) + 1. Motor was stopped at the end of the transfer time and the routine was ended at this point.

Reject from Door:

4 (timeout)

1) indicates that the ID sensors did not 'see' the trailing edge of the disc in the timeout period defined by parameter 14. The order of sensor actions is expected to be ID inner sensor is first tripped before the start of the routine or during the first 500ms, it then un-trips when the trailing edge of the disc goes past the sensor. The motor is then stopped. Transfer time will = (parameter 14) + 1. Current is the motor current at the end of the timeout period. Motor was stopped and the routine was ended at this point.

5 (motor overcurrent)

1) transfer time = 0: indicates that when the controller checked the current on the ID motor at the start of the transfer (time after turn-on defined by parameter 15), it exceeded the limit defined by parameter 10. The current returned is the current read at the start of the transfer. Motor was stopped and the routine was ended at this point.
2) transfer time ≠ 0: indicates that the ID sensors did not 'see' the trailing edge of the disc in the timeout period defined by parameter 14 and when the motor current was checked at the end of the timeout period it exceeded the limit

defined by parameter 10. Transfer time will = (parameter 14) + 1. The motor was stopped and the routine was ended at this point.

Take from Door and Take from Shelf:

Out motion errors

4 (timeout)

1) indicates that the limit switch wasn't tripped during the motion to move the grabber out to the end of the platform in the time defined by parameter 0. The motor is reversed and the controller attempts to park the grabber, the motor and grabber are then turned off and the routine is ended. The out current value holds the current read at the end of the timeout period.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion to move the grabber out (time after turn-on defined by parameter 1) or at the end of the timeout period and it exceeded the limit defined by parameter 2. If the coarse out time value = (parameter 0) + 1 then the over current came at the end of the time out period. The motor is reversed and the controller attempts to park the grabber, the motor and grabber are then turned off and the routine is ended. The out current value holds the current reading for the out motion.

In motion errors when there are out motion errors (during attempted park motion)

4 (timeout)

1) indicates that the sensor wasn't tripped in the time defined by the code which is ~5 seconds.. The motor is stopped and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current after timeout it exceeded the limit defined by parameter 2. The motor is stopped and the routine is ended. The in current value holds the current reading during the return park move.

In motion errors if there are no out motion errors

4 (timeout)

1) indicates that the limit switch wasn't tripped during the motion to move the grabber all the way in to the end of the platform in the time defined by parameter 0, but a over current condition wasn't detected. The motor is then turned off and the routine is ended. The in current value holds the current read at the end of the timeout period.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion to move the grabber out (time after turn-on defined by parameter 1) or at the end of the timeout period and it exceeded the limit defined by parameter 2. If the coarse in time value = (parameter 0) + 1 then the over current came at the end of the time out period. The in current value holds the current reading for the in motion.

11 (grabber edge not detected on retrieve)

1) when the robot starts back with the disc, it looks for the edge of the grabber to cause the outer robot sensor to untrip. If it doesn't see this edge in ~3 seconds, it attempts to move the robot back until the inner robot limit switch is tripped (a platform in move) and then returns this error. During the platform in move the motor current is checked, if it exceeds the value set by parameter 2, the motor is stopped and the routine is ended. If the platform in move is completed without error, the motor is stopped and the routine is ended. The in current value holds the current reading for the platform in motion.

13 (disc not seen on retrieve)

1) once the grabber edge is detected, the controller starts looking for the outer robot sensor to trip to indicate that a disc is in the grabber. If it doesn't see the disc in the amount of time defined by parameter 25, it stops the robot motor, pulses the grabber open for 0.25 seconds and checks for the disc again (the idea is that the disc is too high and needs to be dropped). If it still doesn't see the disc this error is returned and the routine is ended.

Give to Door:

Transfer error

1 (limit switch)

1) indicates that the limit switch on the robot was tripped before the ID inner sensor tripped (saw the disc). The robot motor is stopped and the routine is ended. Note: the robot is in the full out position.

4 (timeout)

1) transit time = 0: indicates that the ID inner sensor didn't trip in the time defined by parameter 18. The robot motor is stopped and the routine is ended.
2) transit time ≠ 0: once the ID inner switch is tripped, the ID motor is started and the controller starts looking for the ID outer sensor to trip. If the ID outer sensor doesn't trip in the time defined by parameter 21, this error is returned, the ID motor is stopped, the grabber is turned off and the routine is stopped. Note that the robot is not returned to its inner position by the routine if an error is encountered.

5 (motor overcurrent)

1) transit time = 0:when the robot starts outward, the controller waits a period of time defined by parameter 17 then checks the robot motor current. If it exceeds the value defined by parameter 1, then is error is returned, the motor is stopped and the routine is ended The transfer current value holds the current reading for the robot motor.
2) transit time ≠ 0: When the ID inner sensor trips, the ID motor is started. The controller waits a time defined by parameter 17 then checks the motor current, if the reading exceeds the limit defined by parameter 10, this error is returned. The motor is turned off, the grabber is turned off and the controller attempts to return the grabber to the full in position (platform in move). The transfer current value holds the current reading for the ID motor.

10 (grabber not in position)

1) indicates that the disc isn't in the park position (park position = robot inner sensor tripped and outer sensor not tripped). The routine will not start.

return error (during move of platform to park or full in position)

If Transfer error = 0: this error are from the Park routine – see that description.

If Transfer error ≠ 0: this error (and current value) is from the Move Platform routine – see that description.

Store on Shelf:

Out error

4 (timeout)

1) indicates that the limit switch wasn't tripped in the time defined by parameter 0. The motor is reversed, the grabber stays closed and the controller attempts to return the robot to the complete in position and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion (time after turn-on defined by parameter 1) or at the end of the timeout period and it exceeded the limit defined by parameter 2. The routine checks at the start and stops the motor if the limit is exceeded and if there is a timeout it checks again before it stops the motor. The motor is reversed, the grabber stays closed and the controller attempts to return the robot to the complete in position and the routine is ended.

In error when there is an out error

4 (timeout)

1) indicates that the limit switch wasn't tripped during the attempt to return the robot to its full in position in the time defined by parameter 0. The motor is stopped and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion to return the robot to its full in position (time after turn-on defined by parameter 1) or at the end of the timeout period and it exceeded the limit defined by parameter 2. The routine checks at the start and stops the motor if the limit is exceeded and if there is a timeout it checks again before it stops the motor.

In error when no out error (during move of platform back to full in position, or can't start error)

4 (timeout)

1) indicates that the limit switch wasn't tripped during the move to return the robot to the full back position in the time defined by parameter 0. The motor is stopped and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion to return platform to the full back position (time after turn-on defined by parameter 1) or at the end of the timeout period and it

exceeded the limit defined by parameter 2. The routine checks at the start and stops the motor if the limit is exceeded and if there is a timeout it checks again before it stops the motor. Return current holds the current reading for the robot motor.

10 (grabber not in position)

2) indicates that the disc isn't in the park position (park position = robot inner sensor tripped and outer sensor not tripped). The routine will not start.

Move Platform:

4 (timeout)

1) indicates that the limit switch wasn't tripped in the time defined by parameter 0. The motor is stopped and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current on the robot motor at the start of the motion (time after turn-on defined by parameter 1) or at the end of the timeout period and it exceeded the limit defined by parameter 2. The routine checks at the start and stops the motor if the limit is exceeded and if there is a timeout it checks again before it stops the motor.

Park:

4 (timeout)

1) indicates that the sensor wasn't tripped in the time defined by the code which is ~5 seconds. The motor is stopped and the routine is ended.

5 (motor overcurrent)

1) indicates that when the controller checked the current after timeout it exceeded the limit defined by parameter 2. The motor is stopped and the routine is ended.

Straighten:

1 (limit switch)

1) indicates that the platform switch was tripped before routine timed out or saw the edge of the case/disc The motor is stopped and the routine is ended.

4 (timeout)

1) indicates the controller didn't see the edge of the disc/case before the timeout period ended. The motor is stopped and the routine is ended.

9 (not back)

1) indicates the robot is not all the way back at start of routine – routine is ended.

14 (#14)

1) indicates the park move at the end of routine timed out. The motor is stopped and the routine is ended.

15 (#15)

1) indicates the park move at the end of routine had an overcurrent condition. The motor is stopped and the routine is ended.

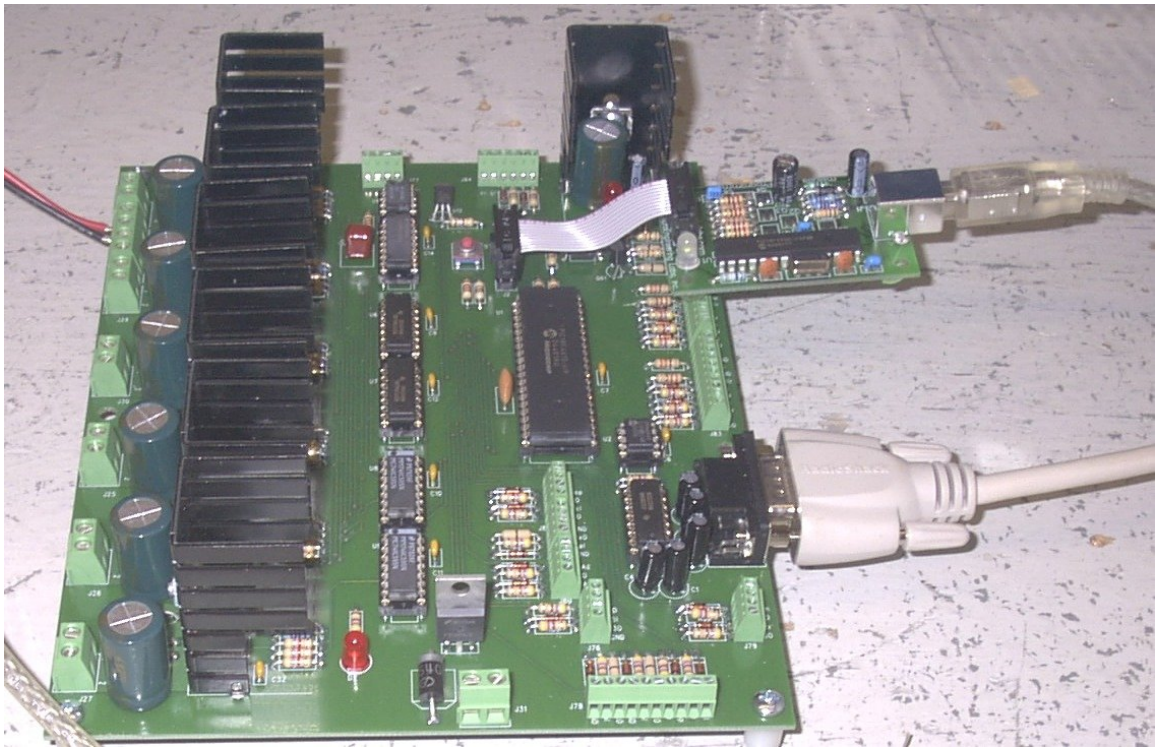
Higher Level Sequences:

Below are some suggested sequences to accomplish some higher level tasks.

1) Accept Case from Customer
    a. Open Door (O)
    b. Accept from Door (A)
    c. Close Door (C)
2) Reject Case back to Customer
    a. Open Door (O)
    b. Reject From Door (R)
    c. Close Door (C)
3) Move Media from Door to Robot
    a. Close Door (C)
    b. Take from Door (F)
4) Move Media from Robot to Door
    a. Close Door (C)
    b. Straighten Disc (E)
    c. Give to Door (D)
5) Hand Disk to Customer
    a. Open Door (O)
    b. Reject from Door (R)
    c. Close Door (C)
    d. Update Status (P)
6) Jam/Remove Jam
    a. Jog Motor (j) + or - (platform or interface)
    b. Open/Close Door (O) or (C)
    c. Update Status (P)
7) Unauthorized Door Open
    a. Check Switch Status (P)
    b. Jog Motor (j) or Set Motor (o)

How to Program the Controller:

1) Connect the programmer to one of the PC's USB ports and to the controller board.



2) Open 'melabs Programmer' on the PC. If the configuration window doesn't open with the programmer window, open it from the View menu (View>>Configuration).

3) Open the hex file from the Programmer window (see figures below).

4) In the configuration window, change Oscillator to HS (see below).

5) Confirm that the programmer is set to program a PIC18F4610 by looking in the window next to the erase icon. If necessary, select PIC18F4610 from the pull down menu by clicking on the down arrow to the right of the window.

6) Make sure power is applied to the controller board.

7) Connect the controller board to the PC's RS232 port.

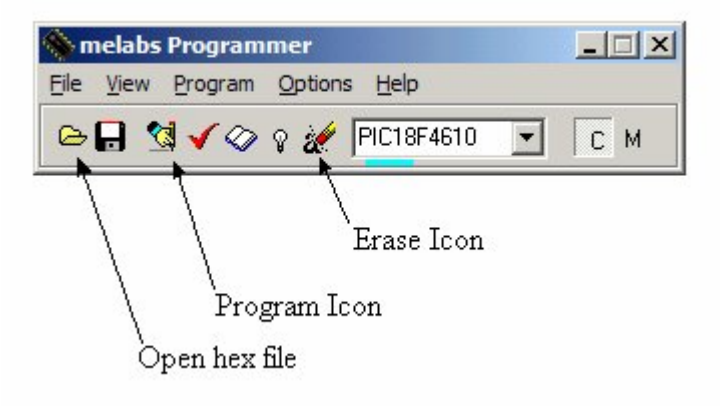
8) Run the interface program on the PC.

9) Click the erase icon in the programmer window. When complete, click OK in the popup window.

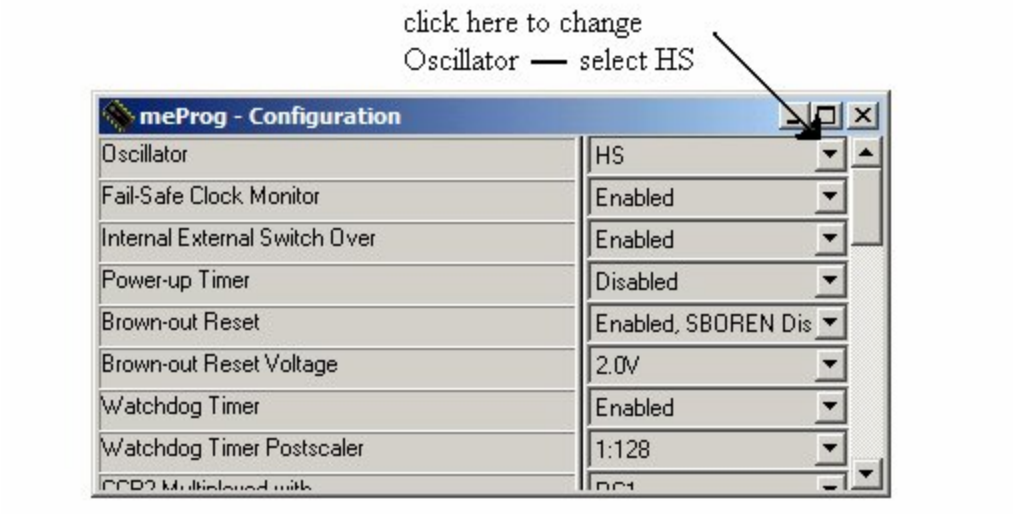
10) Click the program icon in the programmer window. When complete, click OK in the popup window. The controller is now programmed and starts immediately.

Figures:

Programmer Window



Configuration Window



Notes:

1) Do not apply power to the controller board if the programmer is connected to the board, but not plugged into an active USB port. The green LED on the programmer will be on if the USB port is active. Doing so may cause the DC motors to run for a couple of seconds before turning off.

2) If the controller appears to program properly, but doesn't communicate with the host PC, it is probably because you programmed it with an incorrect Oscillator setting. Set the oscillator to HS (step 4) and reprogram.
3) The programmer can stay connected to the controller board while it is running as long as it is connected to the USB port.
4) A "Target device does not match the selected device" error, during erase or programming, can be caused by any of these conditions:
    a. No power to the controller board.
    b. Programmer not connected to the controller board.
    c. Interface program not running.
    d. Programmer not set to program a PIC18F4610.
    e. Connector on ribbon cable offset on the board connector.