Needle insertion test bed: users' and reference manual

Johnston, Daniel F.

https://doi.org/10.4224/21272488
IMTI ITFI Publication; no. IMTI-TR-039, 2007

INDUSTRIAL MATERIALS INSTITUTE

INTSTITUT DES MATÉRIAUX INDUSTRIELS

REPORT
RAPPORT

Pages: 80

Fig.
Diag. 62

For
Pour

Date: September 12, 2007

SAP Project #
# de Projet de SAP 55-DR601

Unclassified
(Classification)

REPORT NUMBER
IMTI-TR-039 (2007/10)

TITLE

Needle Insertion Test Bed – Users' and Reference Manual

Submitted by
Présenté par Dr. Gord Campbell
Group Leader

First Author
Auteur Premier Daniel F. Johnston

Approved
Approuvé Michel Dumoulin
Director

b 21814223 ms

NRC-IMI PUBLICATION APPROVAL FORM

This form **must** be completed and given to your Administrative Assistant **before** a manuscript or abstract is submitted for publication.

1. General information (Please Print)
Author(s) (listed in the order in which they will appear in the citation):

Author	Signature	Date
Daniel F. Johnston	[illegible]	2007-Sept-12

Title: Needle Insertion Test Bed – Users' and Reference Manual

Where will the paper/report be submitted (indicate charges if applicable):

Name, date and location of conference:

2. Approval

	Signature	Date
Principal Author:	[illegible]	2007-Sept-12
Reviewed by Group Leader:	[illegible]	Sept 20, 2007
Approved by Director:	[illegible]	Oct 4, 2007

N.B. A copy should be sent to the Director General

3. Category (this section to be completed by Director)
Please check one of the following:

☐ Journal Paper (Refereed)	☐ Book/Book Chapter
☐ Journal Paper (Non-refereed)	☐ Technical/Client Report
☐ Conference Proceedings (Ref.)	☐ Article in Trade Magazine
☐ Conference Proceedings (Non-ref.)	☐ IMTI Controlled Technical Report
X Technical Report	☐ **Other (specify)**
☐ General Report	

For Office Use Only:
IMI Pub. No: IMTI-TR-039(2007/10) Entered in ProCite: Oct 5/07
73

Table of Contents

List of Figures

Acknowledgements

The following people have contributed to the design and evolution of the Needle Insertion Test Bed;

- Ms. Victoria Hornblower was responsible for original concept and the design of the components, selection of motor, and purchasing. She also performed initial testing and operation of the system with manual commands to control the motor.

- Mr. Bill Wells supported the design process, located some surplus mechanical components which became part of the system, purchased or manufactured remaining components, assembled the initial system, and has been responsible for subsequent repackaging and upgrades (e.g. adding limit switches)

- Dr. Mile Ostojic has independently created a version of needle driver with a similar motor and controller, but higher quality mechanical hardware. It also includes a second motor and controller to rotate the needle. He created his own 'one insertion with rotation' control software. This higher quality needle driver is intended for robotic surgery applications in collaboration with a commercial company, and a patent application is filed for this device.

- Mr. Dan Johnston supplied the software for force data collection and then added the first, simple, 'one insertion' version of automatic motor control software. Recently he created the current control software with better safety and more force/speed options, suggested the addition of safety switches, and created this document.

- Mr. Steve Kruithof created some mechanical drawings from the original CAD models.

1. Introduction

A unique test bed is required to support existing research and collaborative projects. It must insert a needle into different materials and measure the force versus depth profile of this penetration. Some devices exist, such as syringe pumps, which could move at controlled speeds over a defined distance, but none of these would hold the new, patented, Needle Attachment Fixture (NAF) with its built-in force sensor.

The current version of the Needle Insertion Test Bed is shown in Figure 1. This document describes the device and how to use it. It also captures the design of its hardware and the details of the control software.

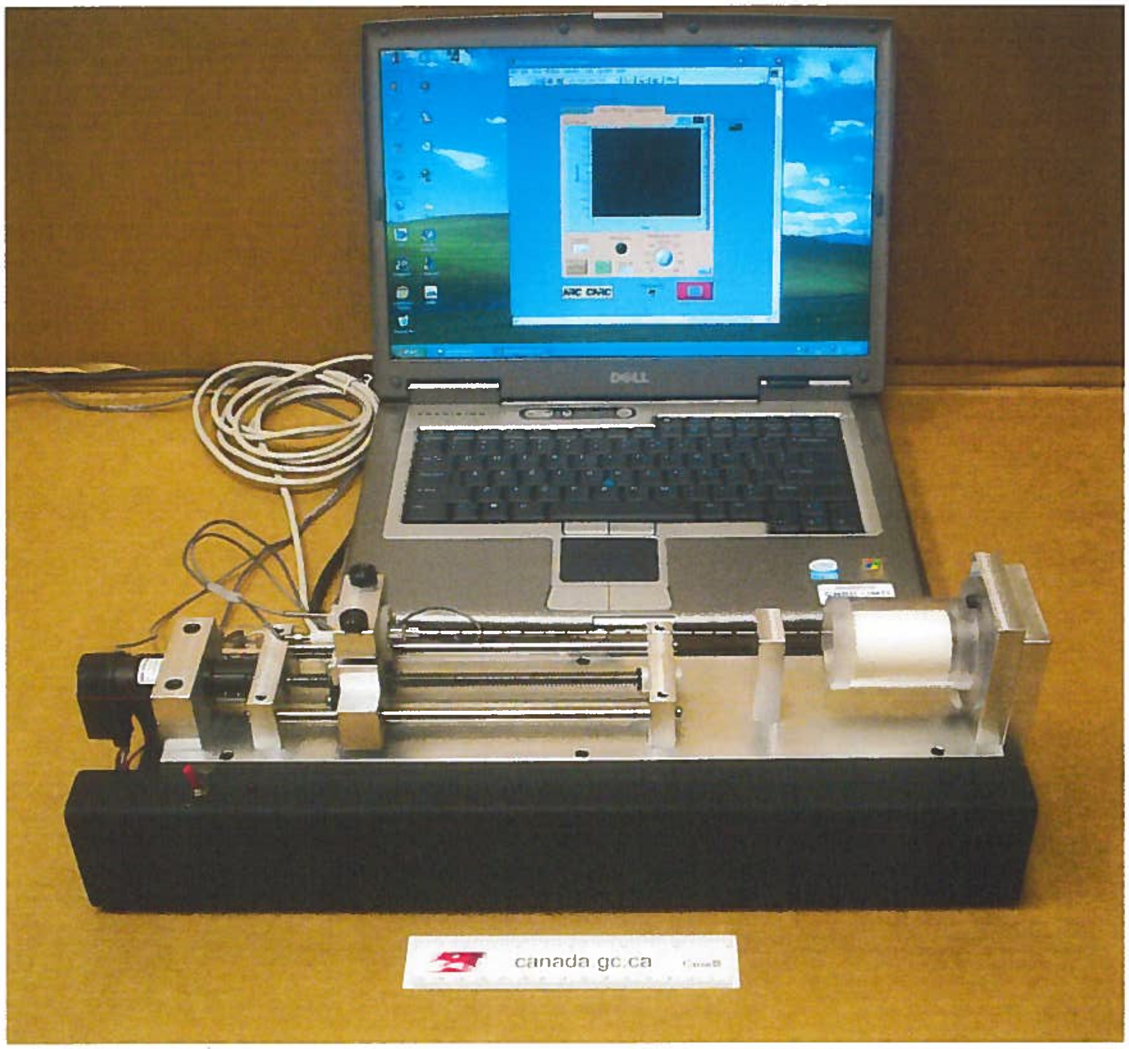


Figure 1 IMTI's Needle Insertion Test Bed (May, 2007)

2. Summary

The National Research Council of Canada, Industrial Materials Institute (IMI), has created a "phantom" for Prostate Cancer Brachytheraphy. This phantom is a 3D physical model of the prostate and surrounding tissue that is accurate in geometry and scale, has similar response for the ultrasound imaging, and is realistic in needle penetration forces. The initial phantom was created with support from a national grant, and then refined as part of a project with a commercial company. This enhanced phantom will be used to train doctors in this common medical procedure. Other phantoms will be developed to mimic different body geometry and other medical conditions.

To select materials for the components of these phantoms it is necessary to test many samples of material for their (force) resistance to needle penetration. These tests must be done with a consistent and repeatable needle velocity, so a new device was needed to perform these tests. A motor on this new device will drive standard needles into material samples with a controlled velocity and to a defined depth. Force data will be recorded during this needle penetration. Then the device will withdraw the needle from the sample and prepare for the next insertion.

The Needle Insertion Test Bed was designed and developed to perform these material tests. This document was created to record the design of the test bed, and will emphasize the operation and software for this testing system. History of the device is provided in Section 3. A complete description on how to use the device for material evaluation will be found in Section 5. Some details of the mechanical and software design will be found in Sections 4 and 6, and in the appendices. Descriptions of the Needle Attachment Fixture and its data acquisition system can be found in other documents.

3. History

A grant was awarded to NRC by the Canadian Prostate Cancer Research Initiative (CPCRI) to develop a phantom with needle penetration forces similar to those encountered during actual prostate brachytheraphy treatments. Part of the research project created a system to measure these needle forces versus depth in the body. Phantom development required the systematic testing of different materials for penetration by the same types of medical needles. A Needle Insertion Test Bed was developed to perform these tests. Standard medical needles for prostate cancer brachytheraphy and prostate biopsy would be used in the device.

A search for existing mechanical hardware began in May, 2006. A suitable motor and controller was purchased and a borrowed power supply was connected. Needles were attached via a force-sensing Needle Attachment Fixture (NAF). The new device was tested using the vendor's supplied software running on the data collection computer. Figure 2 shows this early system ready for material testing. Motions of the motor were controlled by laboriously entering sequences of commands into the vendor's software.

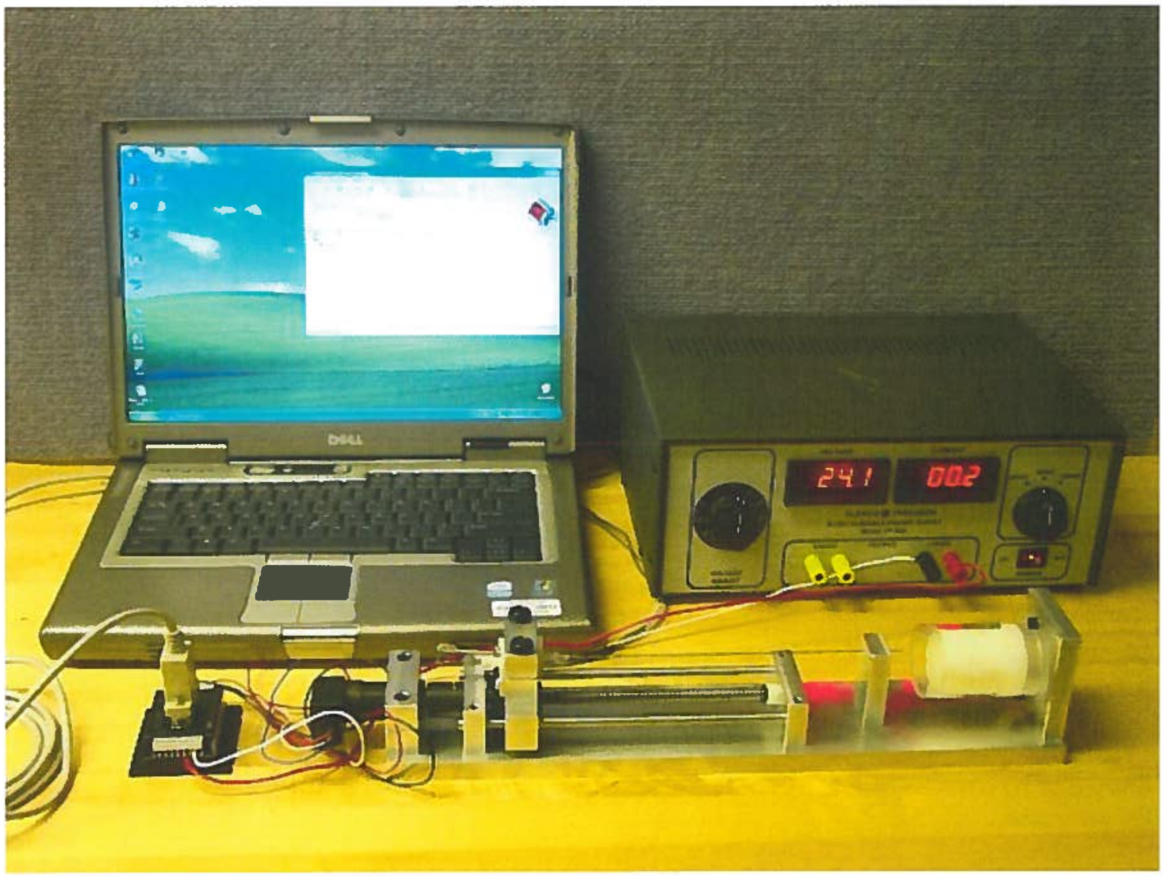


Figure 2 Early version of Needle Insertion Test Bed

By the fall of 2006 the system was being heavily used so the motor and controller hardware was re-packaged, with its own power supply, into a single, solid assembly. Software was created that integrated the existing collection of force data with several standard motor commands. Now the user could easily define the distance the motor would move and the speed it would drive the needle. Still, the software could only provide a single insertion motion and speed, and there was no provision for motor limit switches or any motor status checking. Several needles were destroyed during testing - but no one was hurt.

In April 2007 a new version of the control software was created. This version supports up to five combinations of distance and speeds for the motor so that motions can imitate the techniques observed in actual medical procedures. The new system also provides much more checking of safety status. New software was designed to make the device as safe and automatic as possible. Limit switches have been added to improve reliability and safety during the needle penetration and force data collection.

4. System Description

General descriptions of the mechanical and electrical components of the Needle Insertion Test Bed are found in this section of the document. This device was created to solve an immediate materials testing need for an active research project. Salvaged, new and manufactured components were assembled to create a suitable test system in a very short time and with a very low cost. Improvements have been to remove deficiencies identified when the device was used. The following information on the system is to be considered the "as built" description.

4.1. *Mechanical Components*

Note: Refer to Figures 3 and 4 for components mentioned in this section, and to Appendix A for component drawings.

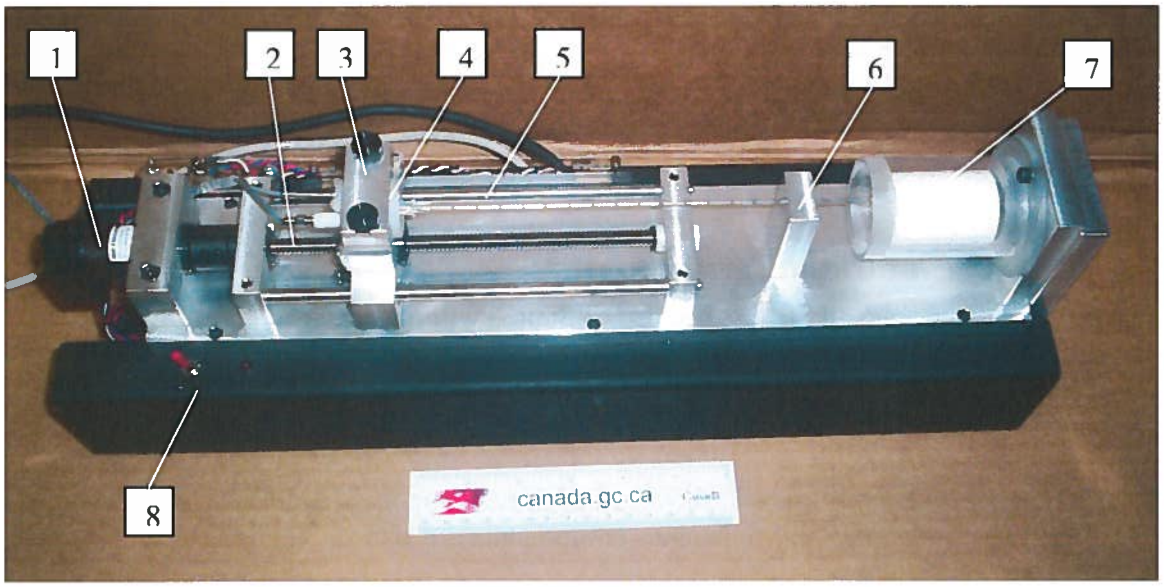


Figure 3 New Test Bed – Power Switch Side

A motor and attached position encoder (Fig. 3, Item 1) provide rotary motion, which is translated to a linear motion by the worm gear (Fig. 3, Item 2) and the traveller (Fig. 3, Item 3). Needle Attachment Fixtures (Fig. 3, Item 4), with their sensors for position and force, are clamped into the traveller. Needles (Fig. 3, Item 5) are inserted through the needle support (Fig. 3, Item 6) and are threaded into the needle attachment fixture. When in operation, this needle will be driven into the material contained in the sample holder (Fig. 3, Item 7) and the forces will be recorded by the software. Two limit switches (Fig 4, Item 1) are provided to restrict motions of the traveller and the needle.

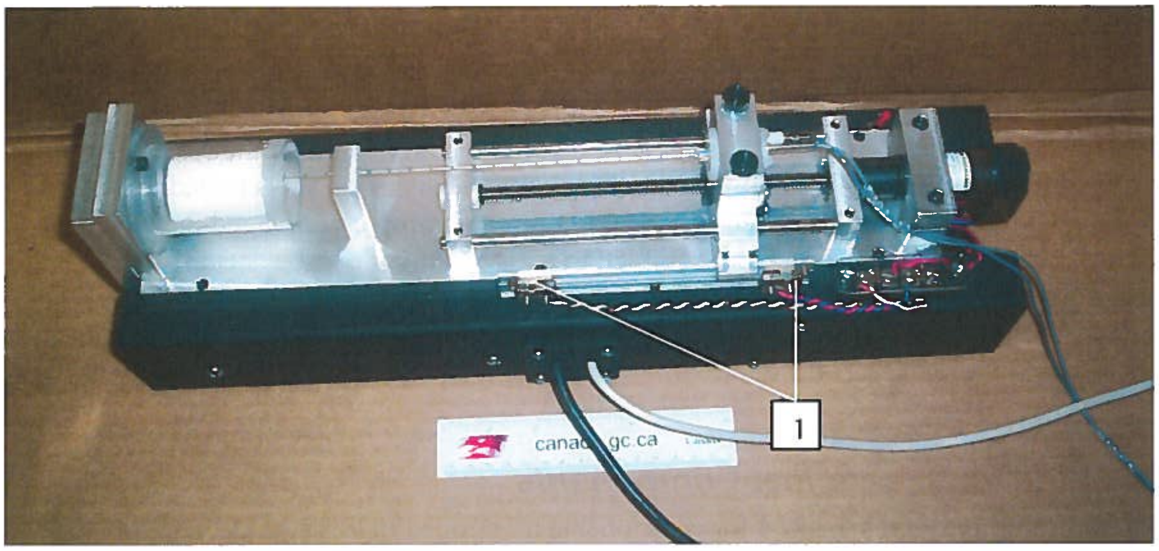


Figure 4 New Test Bed – Limit Switch Side

All motion components are mounted on a base plate, which is in turn mounted on a metal enclosure. This enclosure contains the main power switch (Fig 3, Item 8), the power indicator light, the electrical fuse, the power supply and the motor controller electronic module.

4.2. *Electrical Components*

Most of the electronics in the test bed are contained within the motor controller, and this is packaged by the vendor as one module. Connections to the motor and encoder are made to this module. The power supply and limit switches attach to this same module. A connector on this module is attached to the serial communication line of the computer.

The following diagram, Figure 5, shows all of these components and their connections. Switches, fuses and other miscellaneous components are also shown.

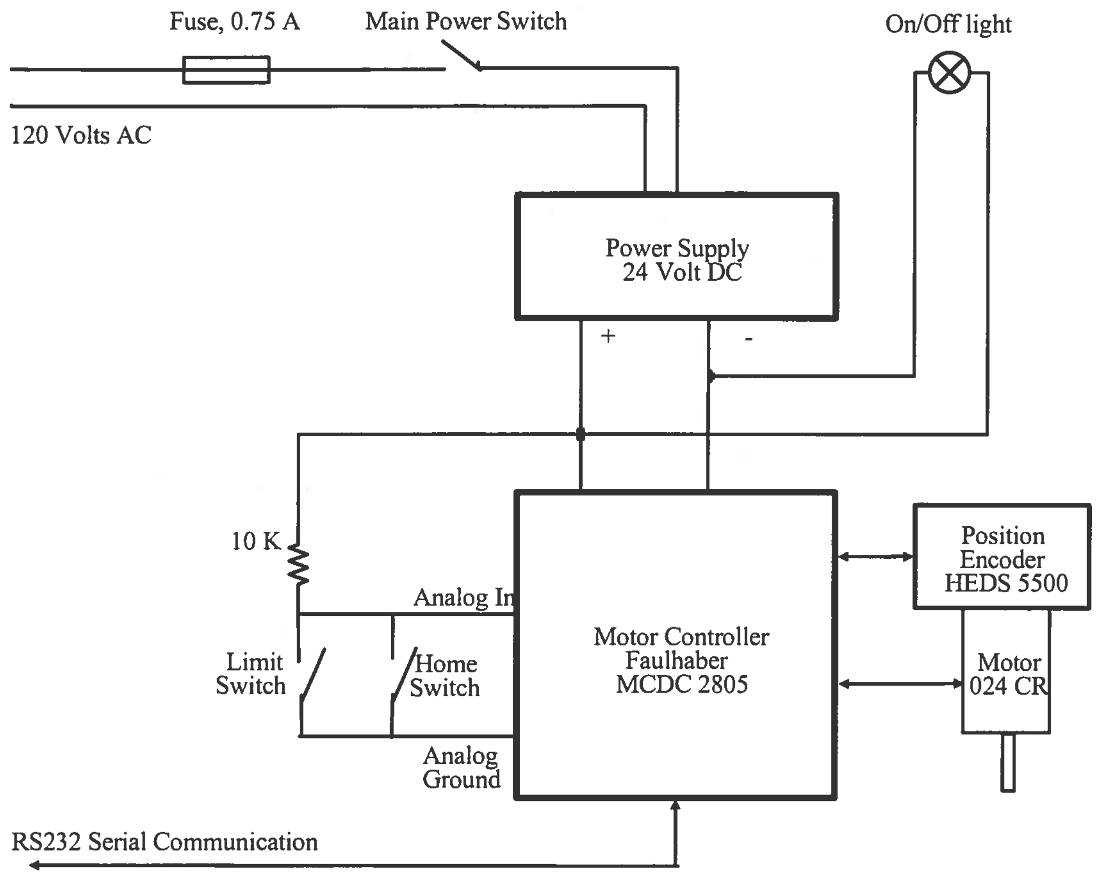


Figure 5 Electrical Block Diagram

4.3. *Software Components*

All the control software modules for the Needle Insertion Test Bed, and its interface for the user, are written using the LabView© graphical programming environment provided by National Instruments. This control software, called "NeedleInsertion", will only run on a computer which has the application and its sub-components installed and a full, licensed version of the LabView software. Or, the application can be installed as a "stand alone" version where it is packaged with all the support needed to run on a computer which does not have a full version of LabView.

The "NeedleInsertion" software can be considered to contain five (5) main sections as shown in Figure 6. All of these software functions are described in other sections of this document, and in detail in Appendix B

When started up, the 'initialize' section of the software will perform a set of tests to verify correct connections to the test bed, and to check motor and limit switch status. Then the motor will move the system to search for the 'zero' or 'home' position. Errors or timeouts in this search will halt the system. This software section provides a more secure and safe environment for the subsequent operation of the test bed.

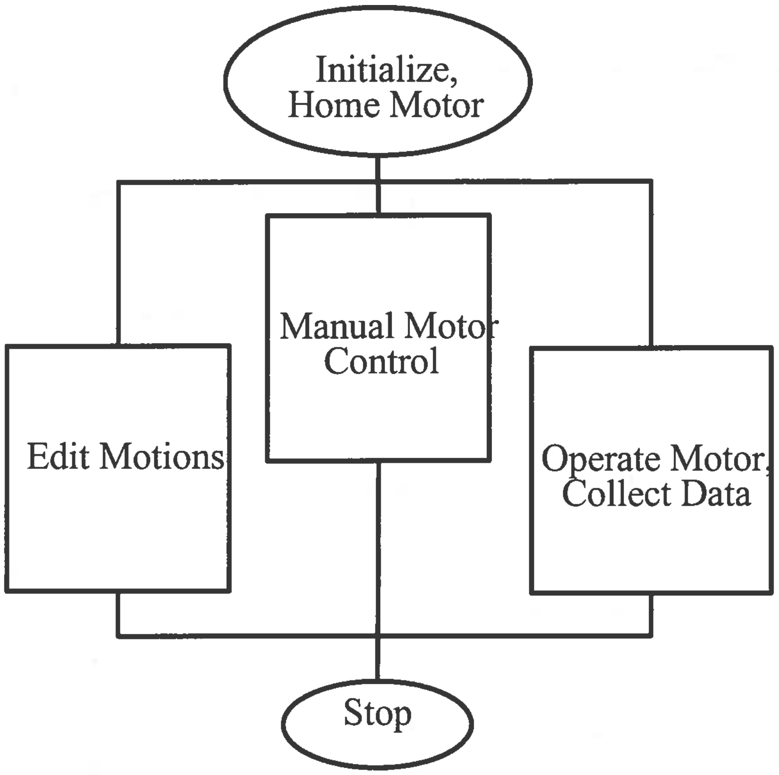


Figure 6 Control Software - Functions

Tabs on the software's main user interface allow the selection of one of the three main software sections.

- One section provides simple manual control over the motor. Users can verify system status, observe the current position, move the motor, and reset the home position.
- Another section displays a table of five motor movements that can be edited. Each line in the table defines a distance to move, a speed to use while travelling that distance, and a time to wait after this movement before proceeding to the next. A switch at the end of every line in the table will allow this movement to be changed from a constant motion to a fast set of 'jabs'. Users can modify one of these motion profiles, or create their own.
- The final, and very important, section of the software will start the motor so that it follows the currently defined array of motions. Force data will be measured from the needle, collected, and stored into a file during the entire insertion sequence at the sample rate defined by the user. This software section will allow the motion sequence to be repeated as many times as the user needs, with the force data from each insertion stored in its own data file.

The "Stop" software section will shut down and disable the motor to prevent any needle movement after the data collection is complete. Then the software application will halt. The software will also enter this section, with an error message displayed, if ever a limit switch is activated during a motor/needle motion.

5. User's Manual

This section provides a description of how to use measure needle insertion force using the test bed. Emphasis is placed on the features of the Needle Insertion Test Bed and the software that controls its operation. Descriptions of the NAF and separate data collection hardware system are found in their own documentation and reports.

5.1. Typical Operation

5.1.1. Start-up

Before operating the test bed, always perform a quick visual inspection. Power for the test bed should be initially switched off. Check that the motor rotates freely, that the traveller can move forward and backward easily, and that the traveller is positioned close to the 'home' limit switch, i.e. lose to the motor. Verify that the Needle Attachment Fixture (NAF) is securely attached to the traveller and that its sensor cables can move freely even if the traveller moves over its full range of motion. Now the test bed can be plugged in and powered 'ON'. The light next to this power switch should now be lit.

Connect the computer, i.e. the research computer with data collection hardware and the labview software, to the test bed and start the "NeedleInsertion" application. A user interface in the LabView style will be visible on the main computer screen. Start this application (arrow button at top left). A key feature of the control software is the initialization and 'find home' component. When first started the application performs a number of checks to verify that there is a motor controller of the correct type attached to the computer, that the status of the controller and limit switches are as expected, and that the motor can move. If any of these tests fail, the user will see a warning light displayed in the software's user interface and the screen will display an error message.

If the motor communication warning illuminates, check the test bed power and/or the computer connection cable. The application will loop at this check until it receives the expected response from the motor controller. Without any error, the software will try to move the traveller and search for the 'home' limit switch. If the limit switch is not found after a few attempts, the application will display an error message and then halt. The test bed must be examined and the problem fixed before restarting the application.

Correct start-up operation of the test bed is to move slowly towards the home position (towards the motor) until that limit switch is reached. Then the motor and traveller will move away from this limit until it is positioned just beside this switch. This location will be assigned as the 'zero' position. All motion will then stop and the system is ready for use.

Caution! Under normal software operation of the test bed the limit switches are used to set the 'home' position for the motor and as a 'stop if too far' limit switch for every programmed movement. It is possible for manual motor operation, or a software error, to drive the motor past the point where one of the switches is turned on. If this ever happens, turn off the main power switch, stop and exit the application software, manually move the traveler back to a mid-range position, restore power to the test bed and restart the application.

5.1.2. Preparation to collect data

Load the sample to be tested into the sample holder and secure the holder on the test bed. The holder can be rotated to allow for multiple needle penetration tests of the same sample. Insert needle tip through the small hole in the needle support. Screw the needle hub into the threaded opening of the Needle Attachment Fixture.

Decide where to store the force data on the computer, i.e. which folder (directory) and which name to use as the template for the set of data files[1]. Carefully record a description of the array of motions as this information is not stored with the force data. Also record the position of each of the 'jab' switches that can be used to modify each part of the motion.

5.1.3. Fine tune position

Based on the sample's size and the needle used, it may be necessary to modify the 'zero' or home position obtained during the start-up sequence. One option is to power everything off, move the physical location of the 'home' limit switch and to re-start the test bed and software application. This would be done if there were a large number of samples of the new size. For a faster and simpler adjustment, click on the "Manual" tab on the main user interface and use the manual control of the motor.

Use the 'jog' button to move the traveler in either direction, i.e. where this direction is selected by a switch provided in the software's display. Another switch will control the size of the motor 'jog' steps. Some status information is displayed on this tab panel, including the status of the limit switches.

Caution: It is possible to 'jog' the motor far enough to activate a limit switch - and beyond. Do not leave the test system in this condition or the remaining software will not work correctly!

Once the 'jog' movement has reached the desired position, use the 'Home' button to establish this motor location as the new 'home' or 'zero' for the test bed. If the "Go to Home" button is used instead, the system will move back to its current 'zero' position. Position information is converted to millimetres and displayed on the interface.

A "Help" button is provided for this user interface tab, and for the others. If pressed, it will display a brief description of the buttons and controls that are visible. Dismiss the help message by clicking on the "OK" button.

[1] If the template name is "Data", then the set of files will be automatically named "Data0", "Data1", "Data2", etc.

5.1.4. Edit motion sequence

If the mode switch on the front panel is set to 'edit', then the software option for defining the insertion motions is active. Decide which sequence of motions to use for the needle insertion. Several pre-defined sequences are provided. These range from a simple 'one motion, one speed' insert to several insert/withdraw operations each with its own speed and dwell time. Pre-defined sequences are designed to imitate the common techniques observed in medical practice. Any of the pre-defined sequences can be modified, or the user can define his/her own sequence.

For each line in the table the user can define a distance to travel in that motion (positive or negative), a speed for this motion, and a time to wait after the motion is completed. Total distance traveled over all five steps is checked against a limit, using a simple summation, and will cause a warning light if exceeded. If any speed value exceeds another pre-defined limit, a second warning light will be lit. If either the distance value or speed is zero, the motor will not move and the sequence will not delay, even if there is a time specified for that line of the table.

Larger positive motions, i.e. larger distance values, can be changed from a constant advance to a series of high speed jabs. Change the jab option to 'on' for the selected line in the table of motions. Jab motions are a series of fast forward and partial reverse movements similar to those used by some medical staff. This jab option switch will be ignored when the distance is negative or when the distance is too small.

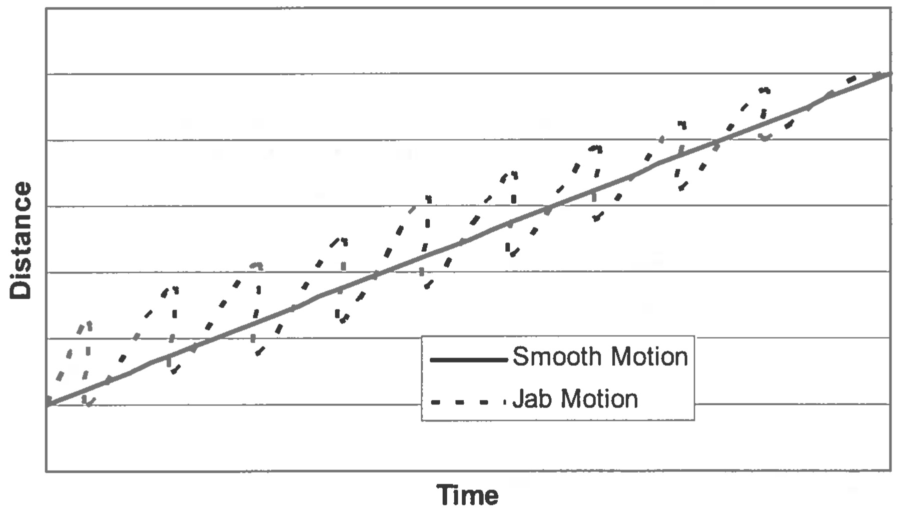


Figure 7 Constant Motion versus Jab Motion

5.1.5. Operate sequence

When the mode switch on the front panel is set to 'operate', then the software option for collecting data during needle insertions is active. Use the "(Re)Start" button to select a new of different location for the force data files, and to specify a 'template' name for the files. After this option the next data file will always have a "0" appended to the template name.

Modify the “rate” value in the software display to control how fast to record force data. In most cases the user will prefer the highest possible number of data samples per unit of time (smallest time per sample) since this will provide the most detail from the experiment. Slower rates will, however, reduce the size of the data files and speed up the post-collection analysis.

Use the “Next” button on the computer display to start the needle moving through the previously defined motion sequence. If either of the two warning lights from the ‘edit’ page are ‘on’, then the needle will not move. The sequence will proceed line by line, with delays if specified, until the end of the last delay, and then the needle will withdraw back to the home position.

Once the needle starts to move, the “Motor Ready” light will extinguish and the “Inserting” light will illuminate. The latter will stay illuminated until the system has completed the defined insertion motion sequence, then it will extinguish. Force data is only collected during this insertion sequence. The “Motor Ready” will remain extinguished until the needle is back at the home position and the motor is stopped.

If the motion sequence causes either the limit switch or the home switch to be tripped, then the software will stop the motor, stop the data collection, and display a message. Move the traveller away from the switch, exit and restart the application, and modify the motion sequence. It may be necessary to change the location of one or both of the limit switches to support the desired motions.

At the start of the first line in the motion sequence the system will also start to collect force data. The raw sensor voltage information is displayed on the application’s graph and also stored in the current data file. The graph will adjust its axis as required, but this behaviour can be modified by the user for the displayed graph while the application is running. Data collection will stop at the end of the last line of the sequence. The current data file will be closed and the system will prepare itself to record another set of data using the same template name but with a new appended number.

If the “(Re)Start” button is used instead of the “Next”, then the user will be able to provide a new template name and thus start a new series of data files.

5.1.6. Stop the application

Whenever the motor is at rest, i.e. at the end of the sequence after the needle is back to the home position, then the user can stop the application by using the large, red, button always visible in the user interface. The motor will be disabled, any open data files will be closed, and the software will shut down. There are other ways to stop the application, and these are described in the next part of this section of the document.

5.2. *Detailed Description of the User Interface*

This section will describe the user interface of the "NeedleInsertion" application in detail. Every button and control will be mentioned and its operation explained.

5.2.1. Application window decorations

The main window of the software application will look like Figure 8 when started by the user. Some of the controls are not created as part of the software development of the application, but are provided as part of every LabView application as a set of controls across the top of the application's display window.

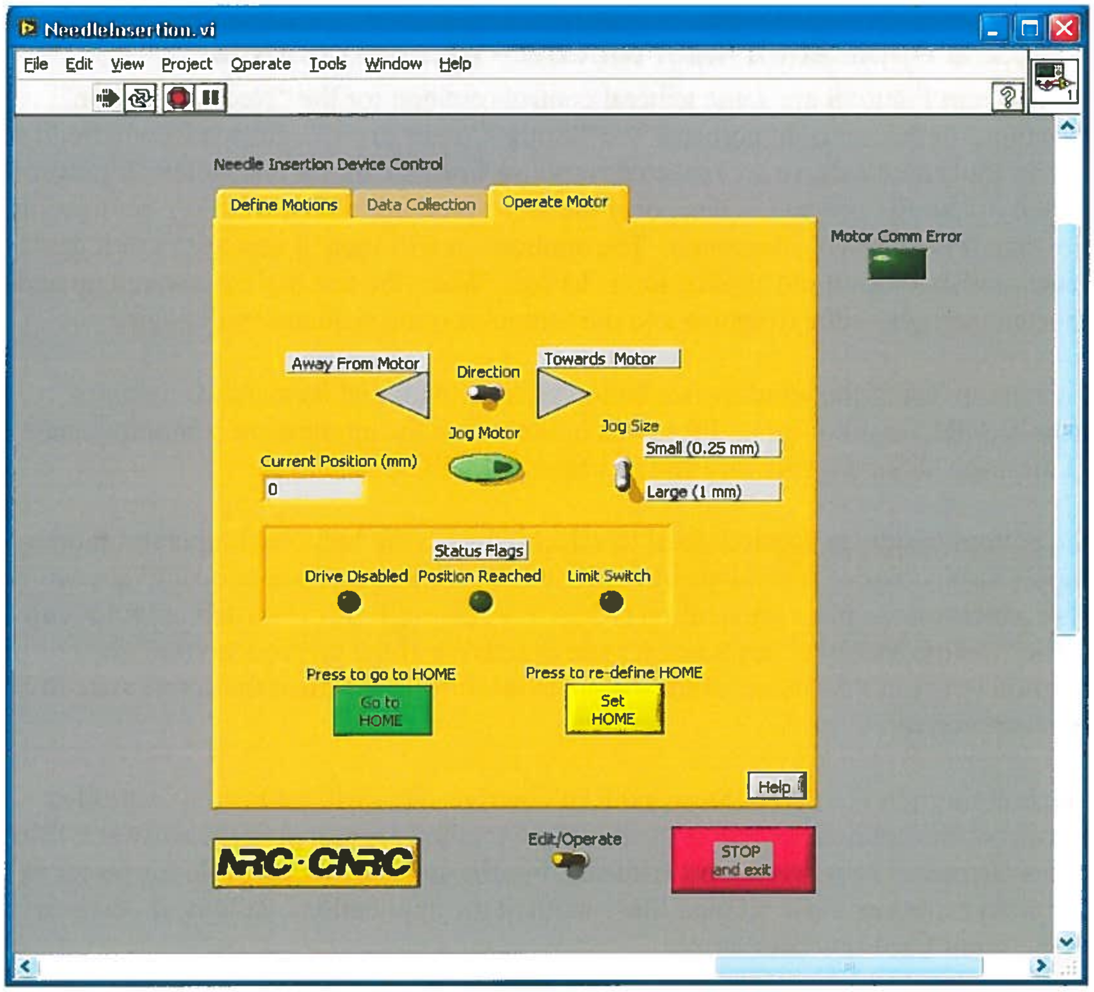


Figure 8 Main application window with decorations, Operate Motor panel

On a computer with full LabView installed, the user will find a row of menu options (File, Edit, Operate, … Help). These are used during the software development and testing and are not needed by anyone just using the application.

There is a row of buttons with small pictures in them. Only two of these are important to the application user; the 'run' button and the 'stop' button. The button on the left side with the right-pointing arrow is the 'run' button. This is used to start the application, i.e. the application does not start working as soon as the main window is visible but waits for the user to activate this control. Use this button when it is time to define a motion sequence and collect needle force data, or to restart the application after an error.

The button with the small, red hexagon represents a small stop sign. It is used to immediately stop the software application. Motion commands that have already be sent from this software to the motor controller will still complete. If this stop control is used, there is no option to allow the motor to be disabled and shut down. It provides an immediate stop.

5.2.2. Application main window – general controls

Also shown in Figure 8 are some general controls created for the "NeedleInsertion" application. In the top right corner is the "Motor Comm Error" light which will be lit if the software cannot receive an expected response from the motor controller. If power is switched off for the test bed device, or if there is a problem with the RS232 connection cable, then this light will illuminate. The application will loop at this test until it gets the expected result, or until the application is halted. When the test bed is powered up and connected the light will extinguish and the 'home' motion sequence will begin.

In the bottom left of the window is a button with the National Research Council of Canada's (NRC) symbol on it. Press this button when the application is running and a brief message about the software and its copyright will be displayed.

In the bottom middle is a switch used to select between the 'edit' and 'operate' modes of the application. Notice that the position of this switch will activate or deactivate two of the tab panels on the main application (when it is running). If the switch is set to 'edit', then the "Define Motions" application pane is active and the controls on the "Data Collection" pane are disabled. If the switch is set to 'operate' then the active state of both pane's is reversed.

At the bottom right is a large "Stop and Exit" button. This will perform a controlled shutdown of the application and is the preferred method for stopping the software. Motor motions already in progress are not affected, but the software will disable the motor at the end of active motion, close all data files, and halt the application. Restart, if necessary, with the small LabView 'run' arrow.

In the center of the main window is the largest part of the application, with three tabs shown on the top. Select a tab to display the application panes associated with each of the three operation modes of the software.

5.2.3. Application main window – "Operate Motor"

For this description, look at the main application pane as shown in the center of Figure 8. Use the "Direction" switch to select whether subsequent manual motions will move the traveller toward the sample or away from the sample. The size of each motion can be adjusted with the "Jog Size" switch. In "Large" mode, each jog step will be 1 mm. With "Small" selected each jog step will be ¼ mm.

Click on the "Jog" button and hold to cause the motor to perform a manual move. Motion will continue until the button is released. Manual operation will allow the motor to move until a limit switch is activated, and even beyond. A limit switch status indication will illuminate. Do not try to move the traveller in the same direction once the mechanical system has reached its end of travel. Do not leave this application pane until the traveller has been moved off the limit switches.

A few of the motor status values will be displayed while in this application pane. "Drive Disabled" should be on only when the motor has been disabled by the software, such as after hitting a limit switch during a programmed motion. "Position Reached" will flash on at the end of every jog step – because the motor has reached the position it was asked to move to. "Limit Switch" will illuminate whenever either one of the two limit switches is "ON". Limit switches would be triggered if the traveller moved over one of them, or the user could simply be operating these switches as part of system testing.

The position of the needle and traveller is shown on a display as the distance from the current 'home' location. This will be updated after every motor movement. Notice how the position value oscillates at the end of every motion before obtaining the final value. This position "hunting" is caused by the large mechanical inertia of the test bed. Motion 'overshoot' is automatically corrected by the motion controller.

A "Go to Home" button will cause the traveller to reposition itself to the current 'zero' location. It has no affect if already at 'zero'.

Also under the status lights is a large "Set HOME" switch. Whenever this button is pressed the current position of the motor is set as the new 'home' or 'zero' location. The traveller will return to this position after every sequence of motions. Do not set a position as 'home' if either one of the limit switches is activated. Do not set a 'home' position too close to the sample holder as this will cause standard insertion sequences to hit the "end of travel" limit switch.

In the bottom right of this application pane is a small "Help" button. Activate this control to see a small description of the controls shown on the screen and how to use them.

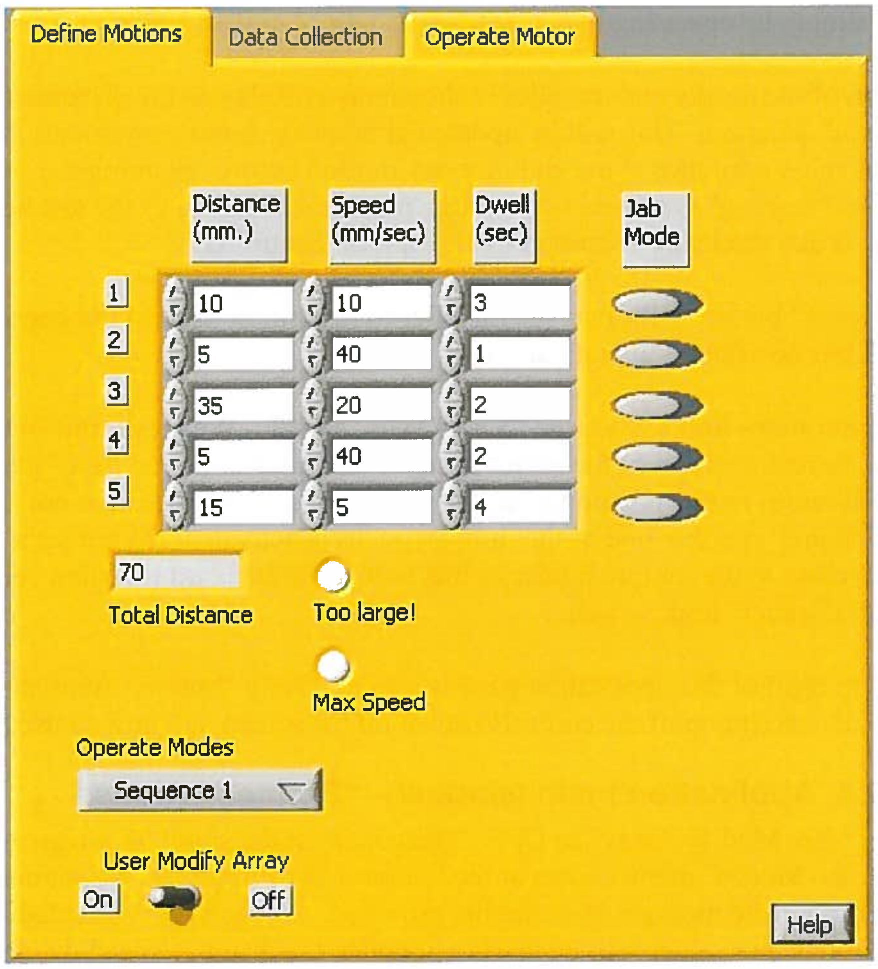
5.2.4. Application main window – "Define Motions"

Switch the "User Modify Array" to OFF. Then click on the small 'down arrow' control in the "Operate Modes" menu shown at the bottom left of Figure 9. Select one of the pre-defined sets of needle motions from the list provided. As each set is selected from the control's options, the values will be used to populate the distance/speed/dwell array shown in the middle of this pane. One of the options from this "Operate Modes" menu will clear all the existing values in the array to zero.

Change the "User Modify Array" switch to ON and the user can modify any value in the array. If any speed value is set to faster than the limit (60 mm/sec), then the "Max Speed" warning light will illuminate. Reduce the large speed value to clear this warning light.

If the total of the distances defined in the table is greater than the travel limit (80 mm.), then the "Too Large" warning light will illuminate. Reduce the total distance to clear this warning light. Note that this check is a simple sum of the distance values, i.e. a negative distance is subtracted from the total. The current total distance will be shown in the "Total Distance" indicator. It is possible for a user to define a single distance greater than the travel limit, as long as there are negative distances to reduce the total. This would result in a motion error, i.e. the traveller would activate a limit switch.

The test device and motor will not move to execute the motion sequence if either one of the two warning lights is still active.



	Distance (mm.)	Speed (mm/sec)	Dwell (sec)	Jab Mode
1	10	10	3	
2	5	40	1	
3	35	20	2	
4	5	40	2	
5	15	5	4	

70
Total Distance

Too large!

Max Speed

Operate Modes
Sequence 1

User Modify Array
On Off

Figure 9 Define Motions application pane

For each row in the table, there is a switch to select 'Jab' mode for this movement. These switches are not set or reset by the predefined motion selections provided from the

"Operate Modes" menu, and must always be set by the user. Any motion changed to a 'jab' will operate to move the same distance, but the motion speed will be increased and the motion will be modified to be a series of forward movements followed by partial withdrawals, as shown in Figure 7. 'Jab' mode is not allowed for movements with a negative distance. The faster 'jab' speed is based on the base motion speed, but the new speed cannot exceed 100 mm/sec and will be limited to that value. If the distance to travel is less than 10 mm, then the 'jab' motion option is ignored.

5.2.5. Application main window – "Data Collection"

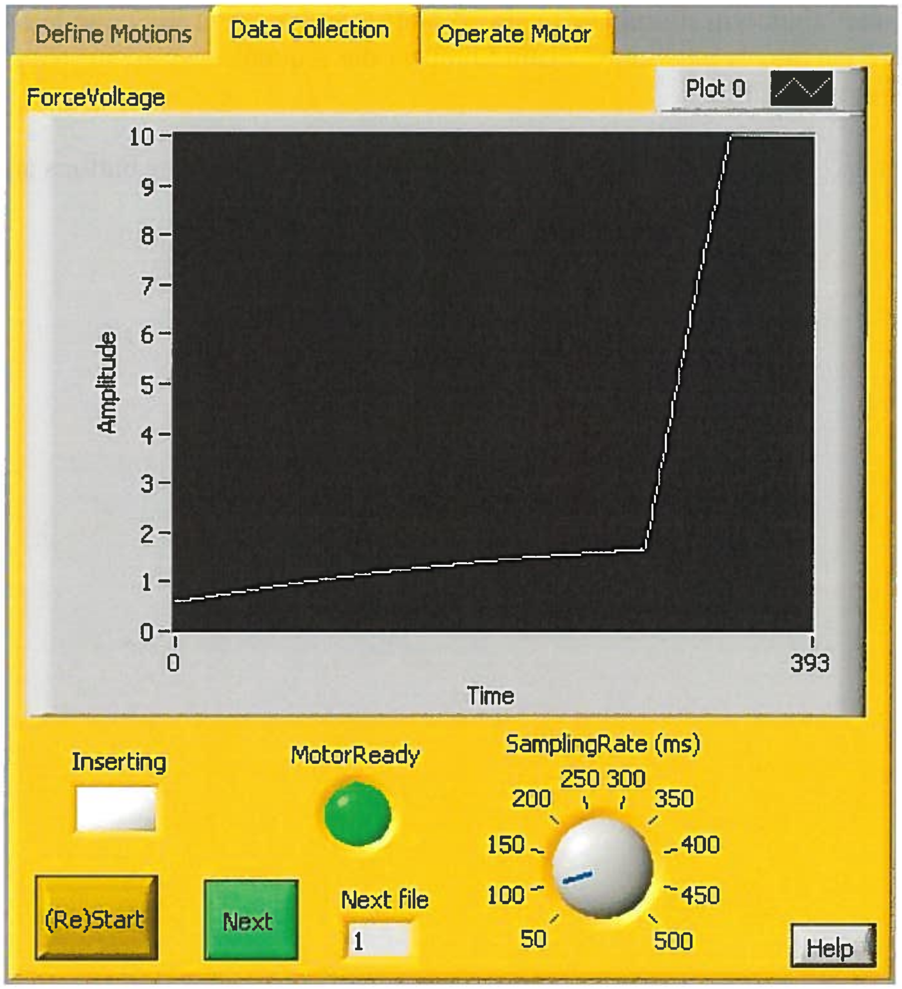


Figure 10 Data collection application pane

Use the "(Re)Start" button on the lower left corner to set the folder to store data files, and to define the template name to use for this next set of data files. When these have been defined the "MotorReady" light will illuminate. In the "Next file" display is the number which will be appended to the template data file name. Use this number in the description of the current data file when making notes during the experiment.

Set the desired sampling rate (time between samples) for the needle force data. The default rate is the fastest one as this provides the most information. Slower rates may miss fine details in the force data, but would result in much smaller data files.

Start the needle moving, and the data collection, by pressing the "Next" button. The "Inserting" light will illuminate and the "MotorReady" light will extinguish. Needle force data will be displayed on the chart as it is collected. "Right click" on the chart to change the auto scale operation of the chart's axis. When the motion sequence is complete the "Inserting" light will extinguish and the data collection will stop. The needle will then automatically, slowly, return to the 'home' position. After the motor is stopped the "MotorReady" light will illuminate again. Notice that the "Next file" number will increment. Use the "Next" button again to repeat the sequence again, but with the data stored in a new data file.

Use the "Help" button to see a message that will remind users of the buttons and their operation.

6. Design – Software

LabView was used to create the control and data collection software. This graphical programming environment is designed to support rapid development of applications, especially those which use data acquisition hardware, such as this system to collect needle force data.

The latest version of software to control the Needle Insertion Test Bed was designed with two main features;

- The motions are more complicated than just a simple, one speed, insertion and withdraw. Doctors often insert, insert at different speed, withdraw, and insert again. Some doctors will use a series of rapid "jabs" instead of a constant insertion motion. The motion control system for the test bed must support these different motion techniques.

- The use of the test bed with early versions of motor control, and with manually created motor commands, caused several minor accidents. Unexpected motions caused the needle to contact the base of the sample holder, bending and destroying the needle. Fortunately, no fingers were pinched or motors damaged. The current software was designed to use safety stitches to limit motions and provide a more secure system for data collection.

Software was created in LabView to implement the two consideration listed above, and to provide the different functions of motor motions and data collection. This section of the document records the design ideas and assumptions used during this software development. Diagrams of the final graphical programs are provided in Appendix B.

6.1.1. Initialize and home functions

Motion commands for the low cost motor and controller used for this device only support a limited response to safety switches. There is only one signal dedicated to limit switches, so it is not possible for this motor controller to distinguish between the switch indicating the lower limit of travel, and another indicating the upper limit. Higher level software, and a few assumptions, must provide this understanding.

When first started the software will attempt to obtain status information from the motor controller. Since no control is possible without this communication, the software will pause at this first step, illuminate an error indication, and wait for the communication to be restored.

When communication is working, the status of the limit switches is determined. If the status is 'Off' then the assumption is made that the traveller is located between the two limit switches. Commands are issued to move the traveller slowly away from the sample holder and towards the limit switch located closest to the motor. If this switch is not activated after a time suitable for a full motion travel, then there must be a problem with the switches, motor, coupling, worm screw or traveller. The software will stop sending movement commands, display an error message, and halt the software.

Normal operation is for the traveller to continue moving until limit switch close to the motor is activated. This position, with the switch now active cannot be the home position. The software must move the traveller off the switch using a set of small reverse movements. If it is not possible to move off the active switch, then the software must have started the system with the traveller already activating the limit switch closest to the sample holder. The software's reverse movements are subsequently driving the traveler closer to the sample holder, and this limit switch cannot deactivate. The code must halt and allow the operator to manually correct this problem.

Once the traveller does move off the limit switch close to the motor, the traveller must still be moved another small distance. Mechanical inertia and overshoot will occur even on slow movements. The traveller will be moved, by the software, a small distance (2 mm) away from the lower limit switch. This final location is defined as the zero or 'home' position.

6.1.2. Motion and limit switches

All programmed motions will include a request to the controller to halt immediately if there is a change in either one of the limit switches. The operator can adjust the locations of the limit switches to insure safe motions for the operator and the equipment. Limit switch reaction is active on all constant insertion motions, will work with some restrictions on 'jab' motions, and will also work on withdrawals back to the home location.

6.1.3. Error response

Whenever there is an error due to a limit switch, the motor will be disabled, an error message is displayed to the user, and the application software will stop. The operator is expected to reposition the hardware so that neither of the two limit switches is active. Any other problems, e.g. a loose motor coupler, broken switch, etc. must also be corrected before the software is restarted.

The motor is disabled after any error or when the software is halted, so that the operator can freely turn the motor and thus reposition the traveller and needle.

6.1.4. Application's constants and limits

Parameters used in the application are listed in the table below. If these must be modified, i.e. if the hardware is changed, then the parameter values must be changed in the software.

Parameter	Value	Description
TotalDistance	80 millimetres	The maximum total distance from all 5 lines in the motion array. The check is simply the sum of all the distance values

MaxSpeed	70 mm/sec	The base speed (speed of any single constant motion) cannot exceed this value
MaxJabSpeed	100 mm/sec	Jab motions move, over small distances, faster (up to 1.5 x) than the basic speed defined for an insertion. However, jab motion speed cannot exceed this higher limit
Home Retract	2 mm	Distance away from home switch to 'zero' position, to allow for motor overshoot
acceleration	0.312 rev/sec/sec	All motor accelerations and decelerations will use this value. The motor will ramp up to its defined speed value at the beginning of a motion, and slow down as it approaches the end.
Command delay	50 milliseconds	The delay after every motor command before allowing the software to issue another command.
Encoder Count	2048 per rev	One motor rotation will create 2048 pulses from the encoder
Worm pitch	2 mm per rev	Each revolution of the motor will cause the worm gear to rotate one revolution and the traveller to move 2 mm

7. Future Plans

The following is a short list of potential changes that could be made to the Needle Insertion Test Bed. There would be significant change required to the mechanical design if either of these two changes was implemented.

- A common technique used by medical practitioners during needle insertion is to rotate the needle in order to achieve a straight line track for a bevelled needle tip.. Typically this is done by rotating the hub of the needle either with a continuous rotation or the common technique of rotating in alternating directions (“bi-rotation”) while inserting. The rotation technique should reduce the force needed for insertion. A second motor and controller would add needle rotation, but would require significant changes to the mechanical design of the traveller. Software would have to be modified to support the second motor’s motion.

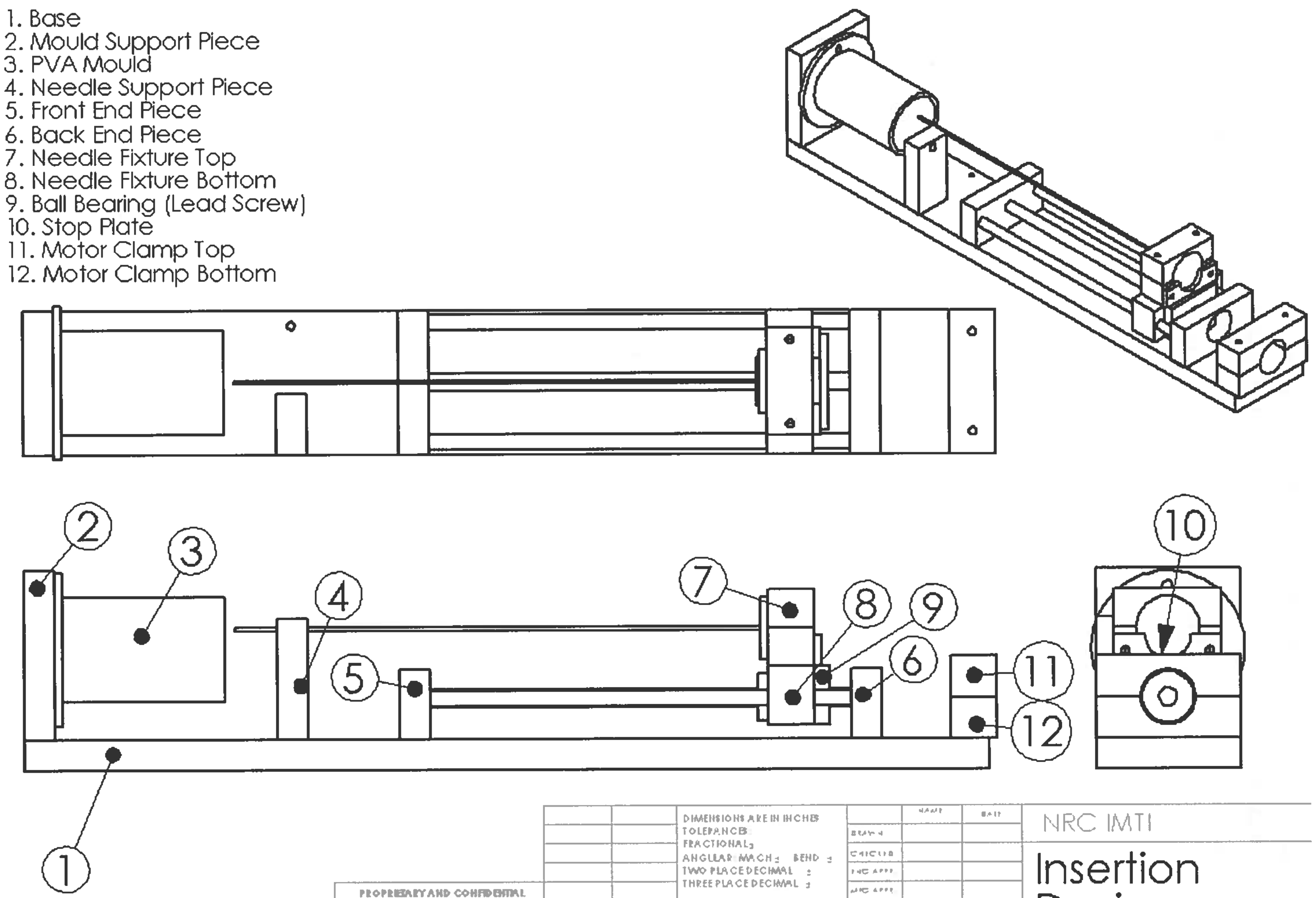
- If this test bed is ever used with other needles and therefore other needle holders, then the traveller must be modified to suit.

Appendix A – Mechanical Drawings

The drawings shown in this appendix were created from CAD models of the original mechanical components of the Needle Insertion Test Bed. These drawings have been captured in this document since they have not been documented elsewhere.

There have been a few minor changes since the drawings were created. All smooth holes shown in the drawings are actually threaded, tapped attachment points. There has been the addition of an enclosure for the power supply and controller. Limit switches have been added. There are no drawings for these modifications.

1. Base
2. Mould Support Piece
3. PVA Mould
4. Needle Support Piece
5. Front End Piece
6. Back End Piece
7. Needle Fixture Top
8. Needle Fixture Bottom
9. Ball Bearing (Lead Screw)
10. Stop Plate
11. Motor Clamp Top
12. Motor Clamp Bottom



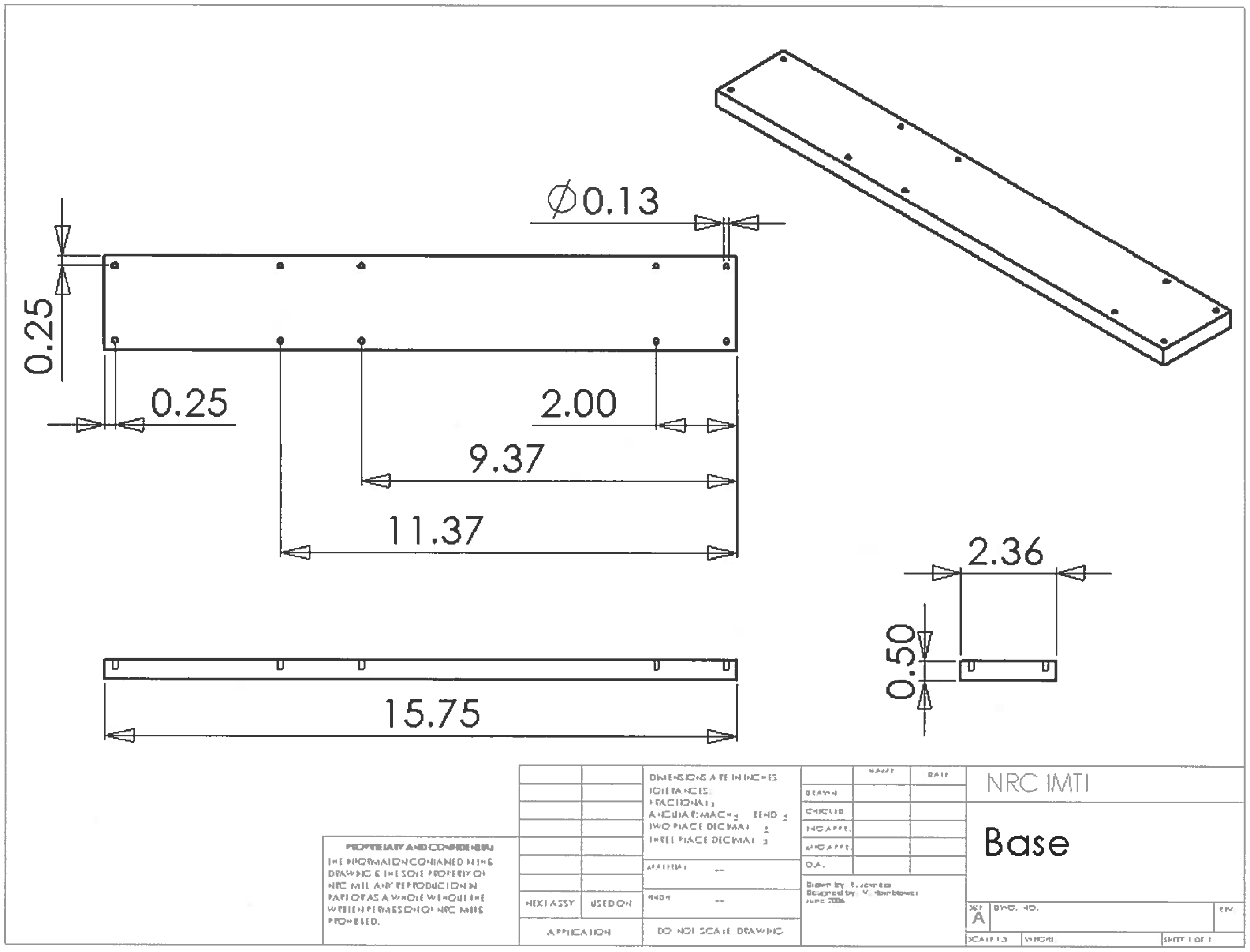
NEXT ASSY	USED ON	DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±		NAME	DATE
			DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		MATERIAL --	Q.A.		
		FINISH --	Drawn by: [illegible] Designed by: V. [illegible] June 2004		
APPLICATION		DO NOT SCALE DRAWING			

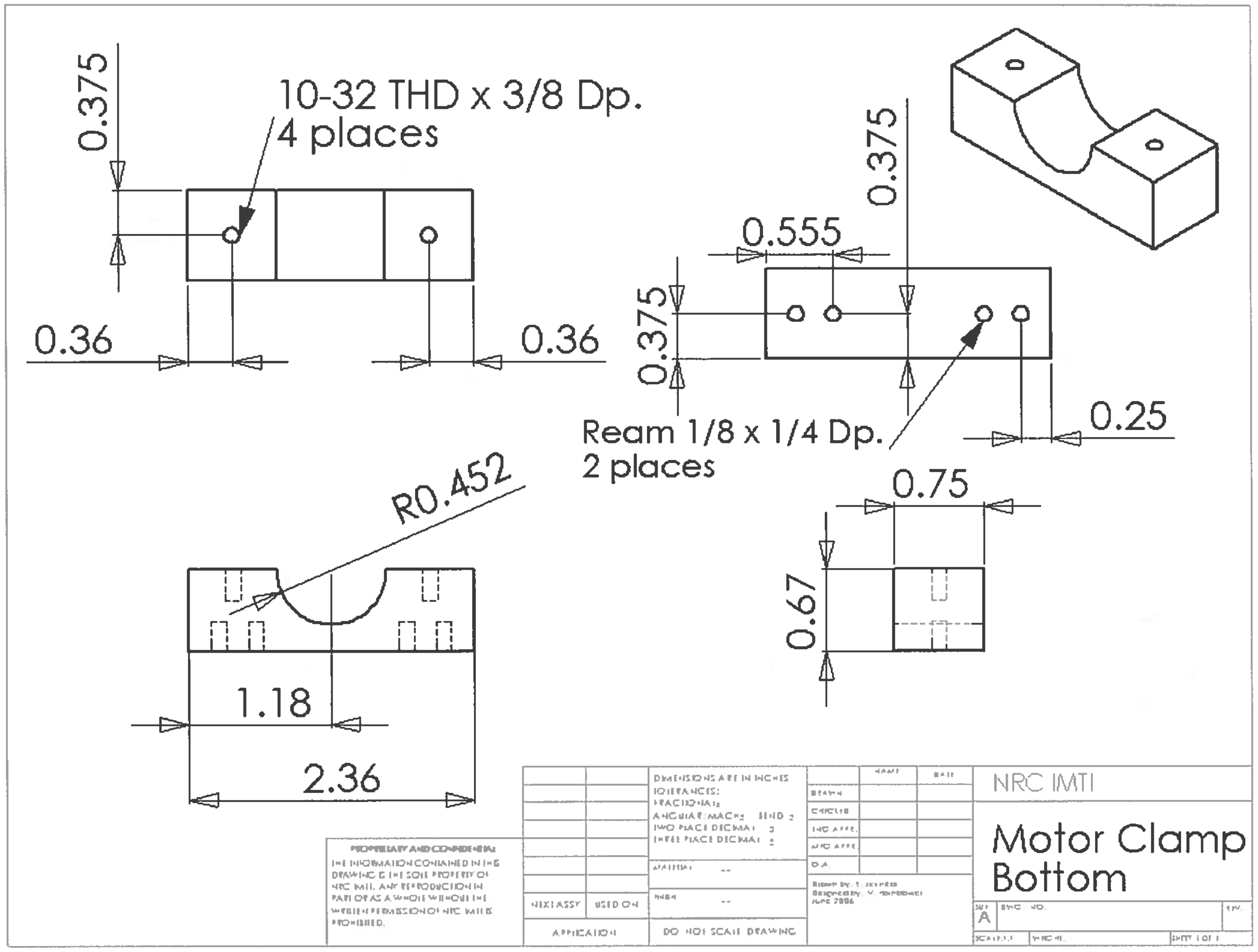
NRC IMTI

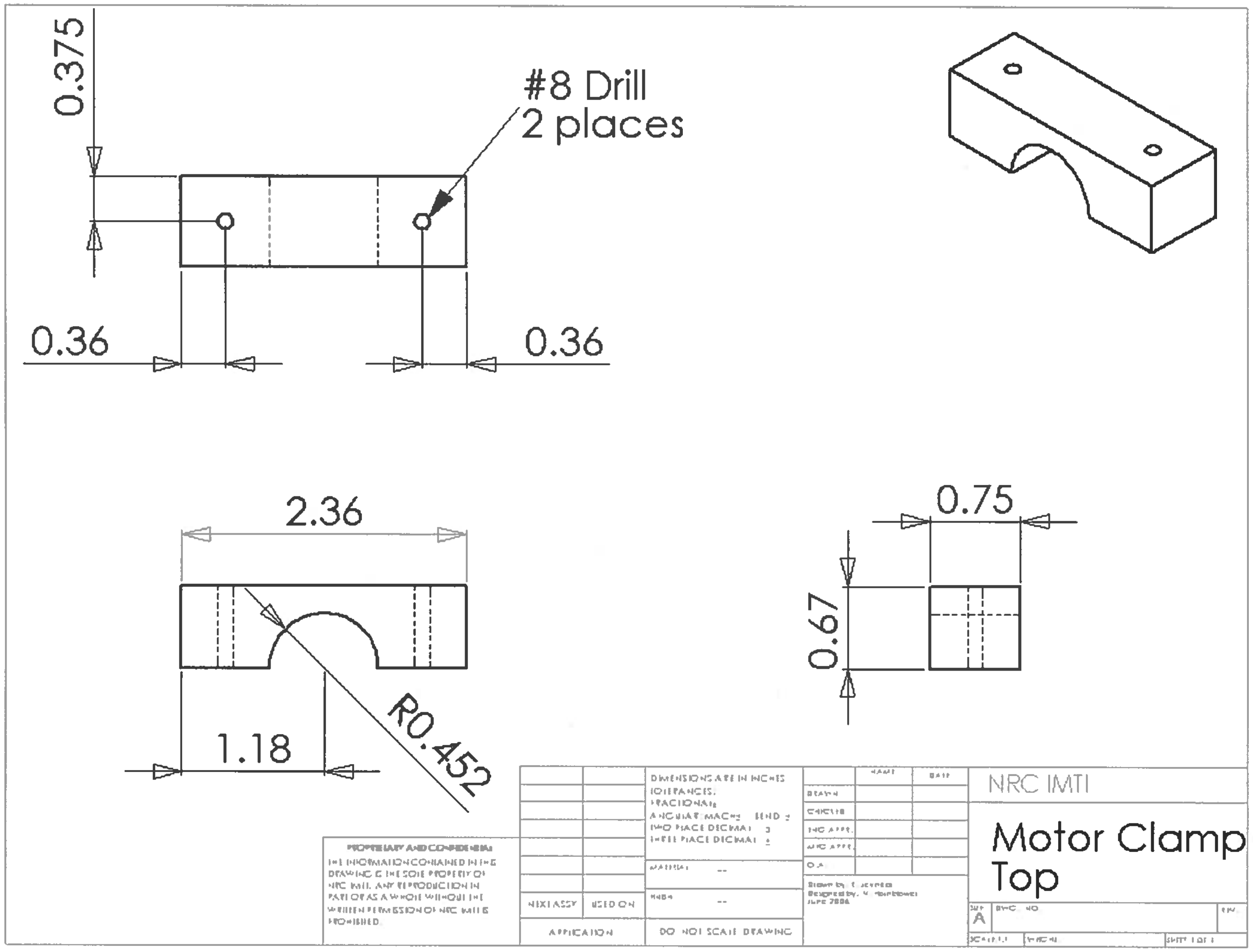
Insertion Device

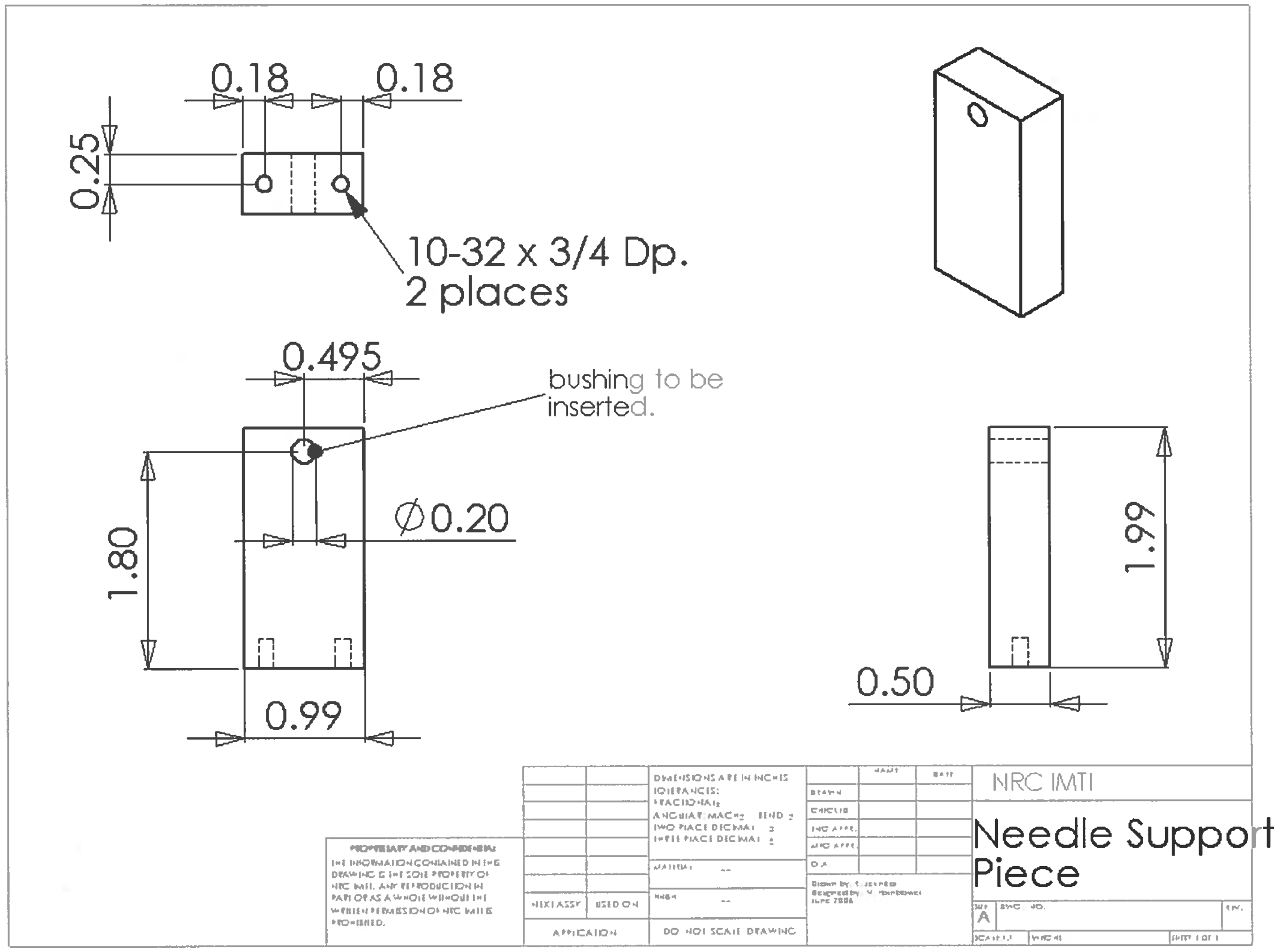
SIZE A DWG. NO. REV

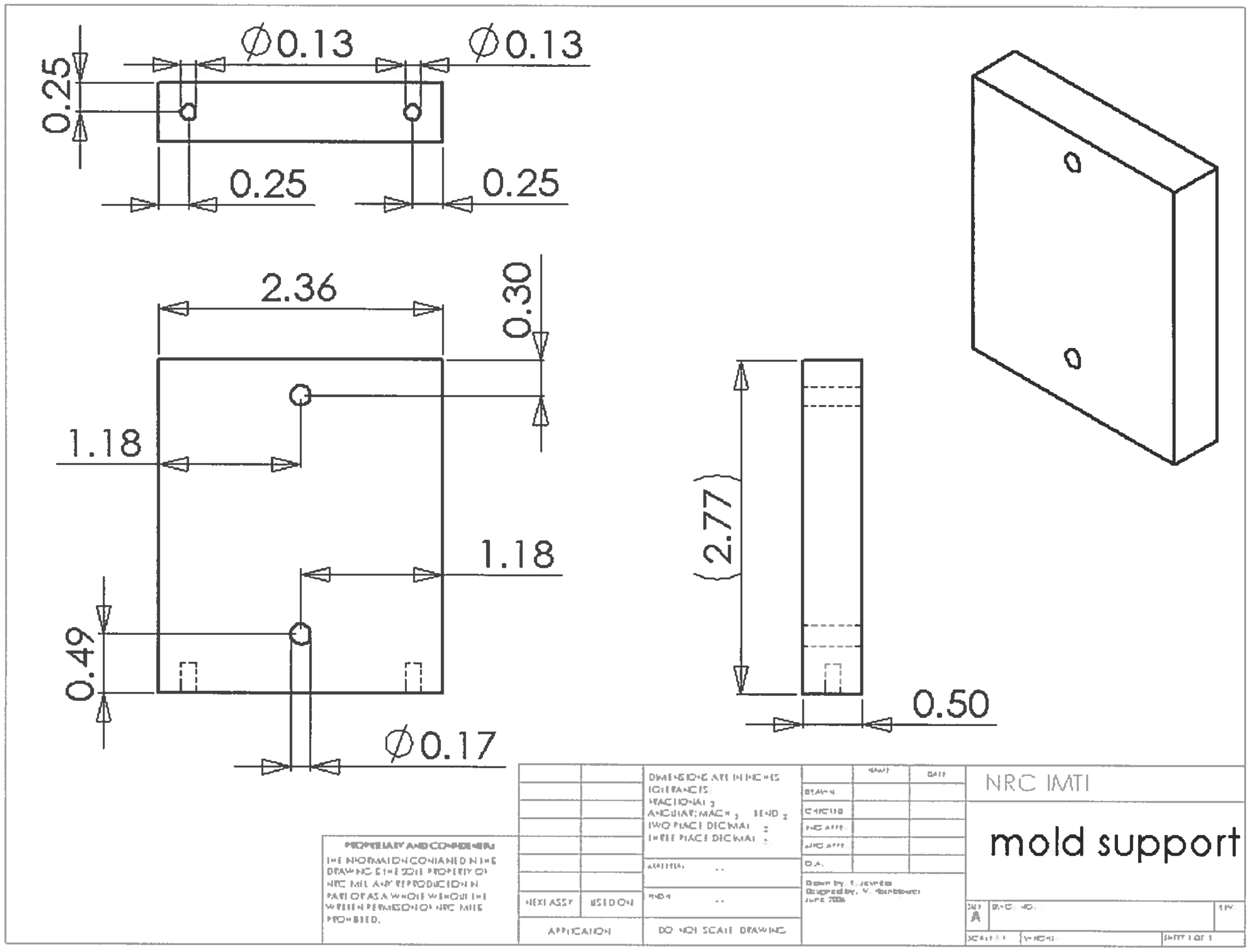
SCALE 1:3 WEIGHT SHEET 1 OF 1

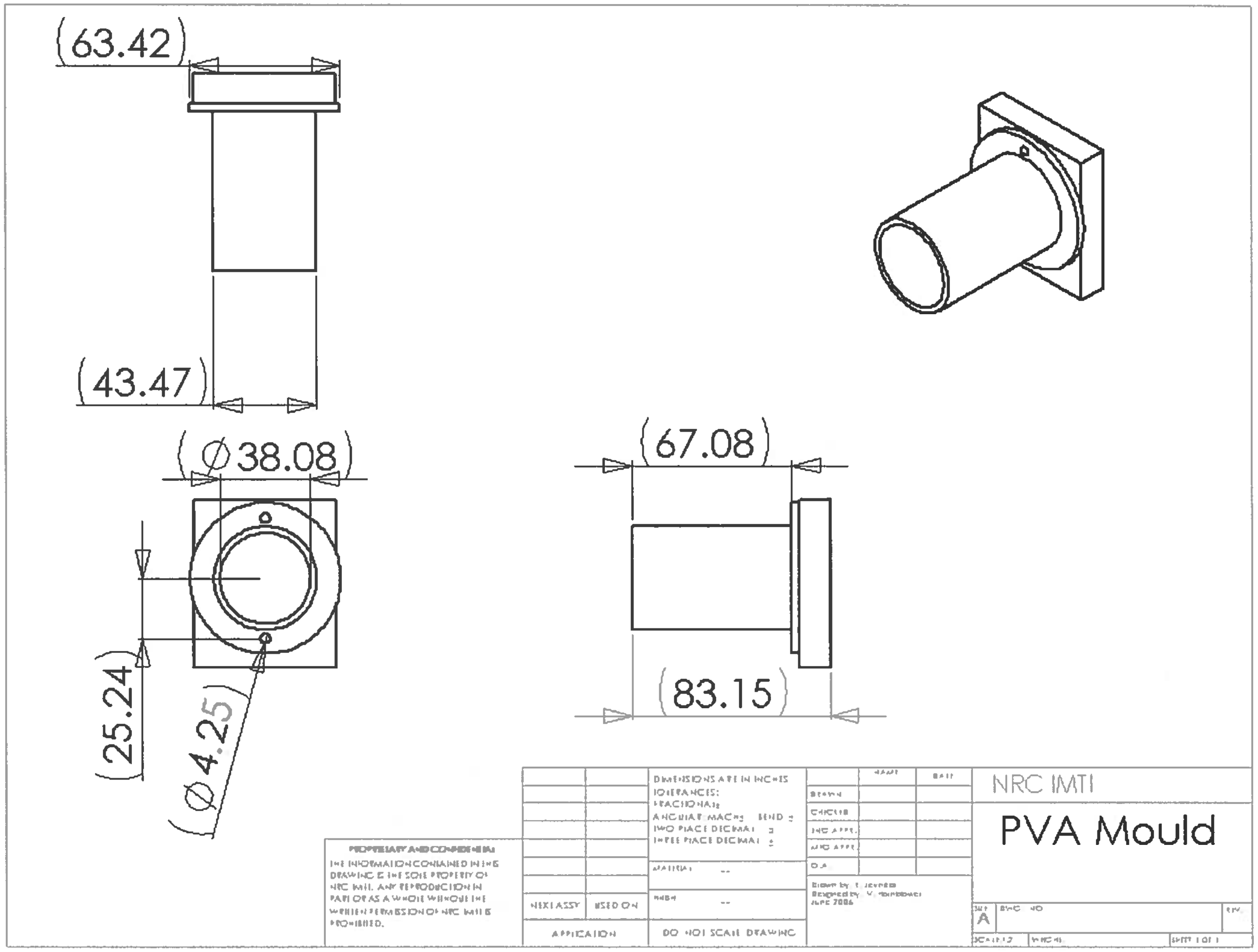

∅0.13
0.25
0.25
2.00
9.37
11.37
15.75
2.36
0.50
NRC IMTI
Base
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by: T. Jewers
Designed by: V. Hornblower
June 2006
SIZE A
DWG. NO.
REV
SCALE 1:5
WEIGHT
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.

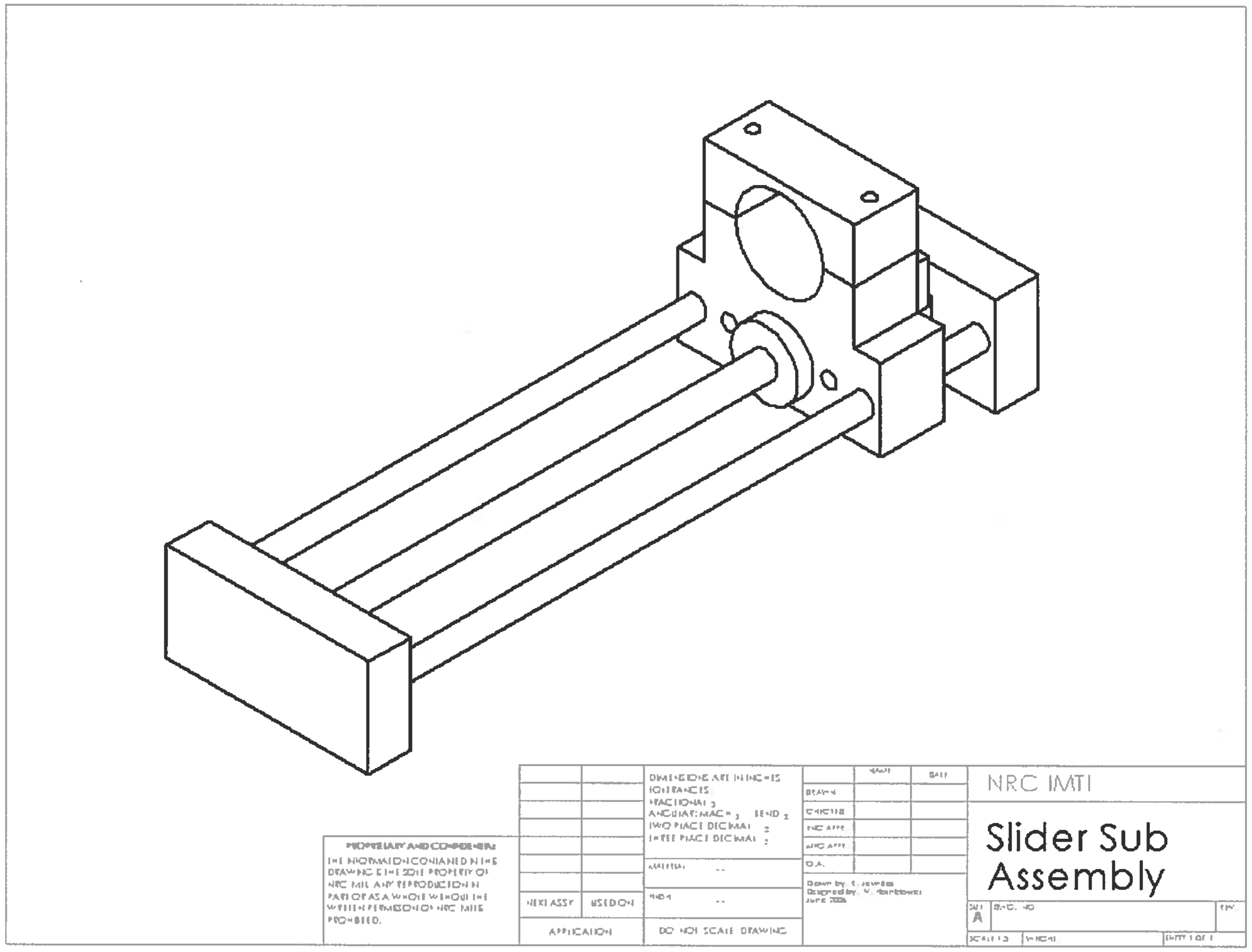

10-32 THD x 3/8 Dp.
4 places
0.375
0.36
0.36
0.555
0.375
0.375
0.25
Ream 1/8 x 1/4 Dp.
2 places
R0.452
1.18
2.36
0.75
0.67
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by:
Designed by:
June 2006
NRC IMTI
Motor Clamp Bottom
SIZE
A
DWG. NO.
REV
SCALE
WEIGHT
SHEET 1 OF 1

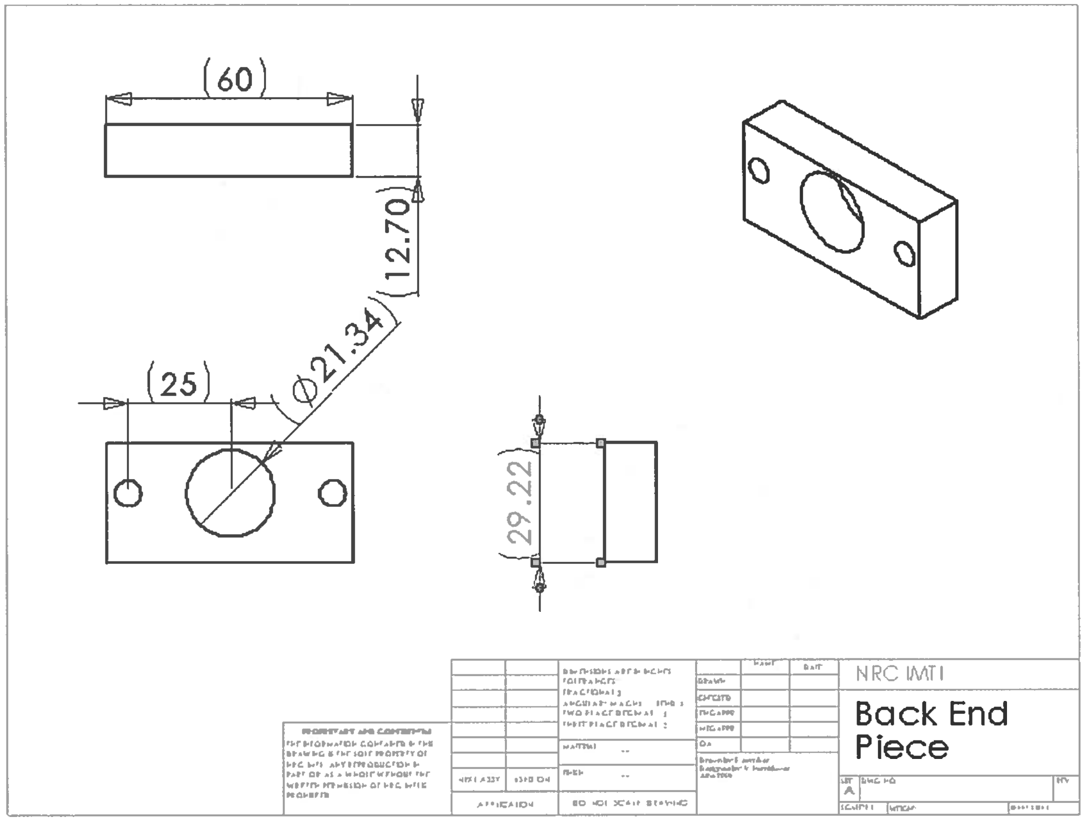

0.375
#8 Drill
2 places
0.36
0.36
2.36
R0.452
1.18
0.75
0.67
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by:
Designed by:
June 2006
NRC IMTI
Motor Clamp Top
SIZE
A
DWG. NO.
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1

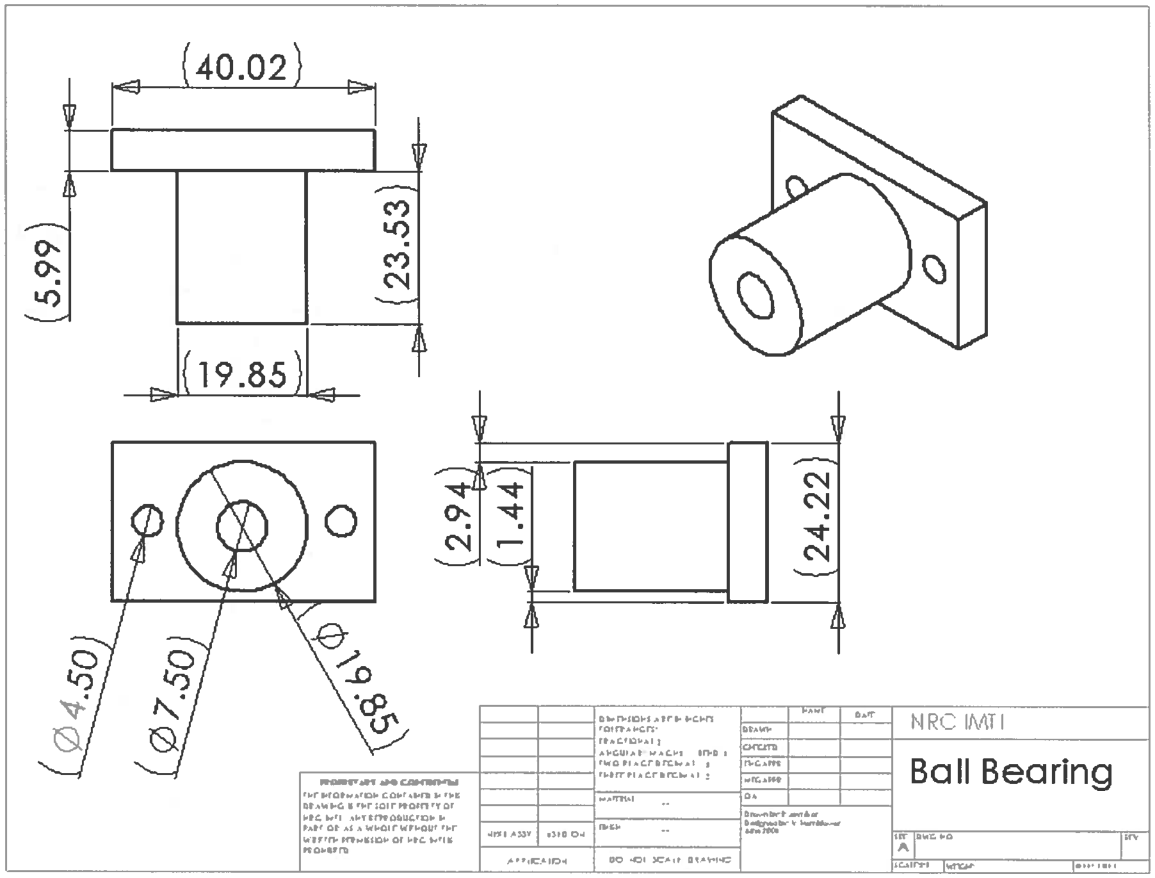

0.18
0.18
0.25
10-32 x 3/4 Dp.
2 places
0.495
bushing to be inserted.
∅0.20
1.80
0.99
1.99
0.50
NRC IMTI
Needle Support Piece
DO NOT SCALE DRAWING
APPLICATION
NEXT ASSY
USED ON
A

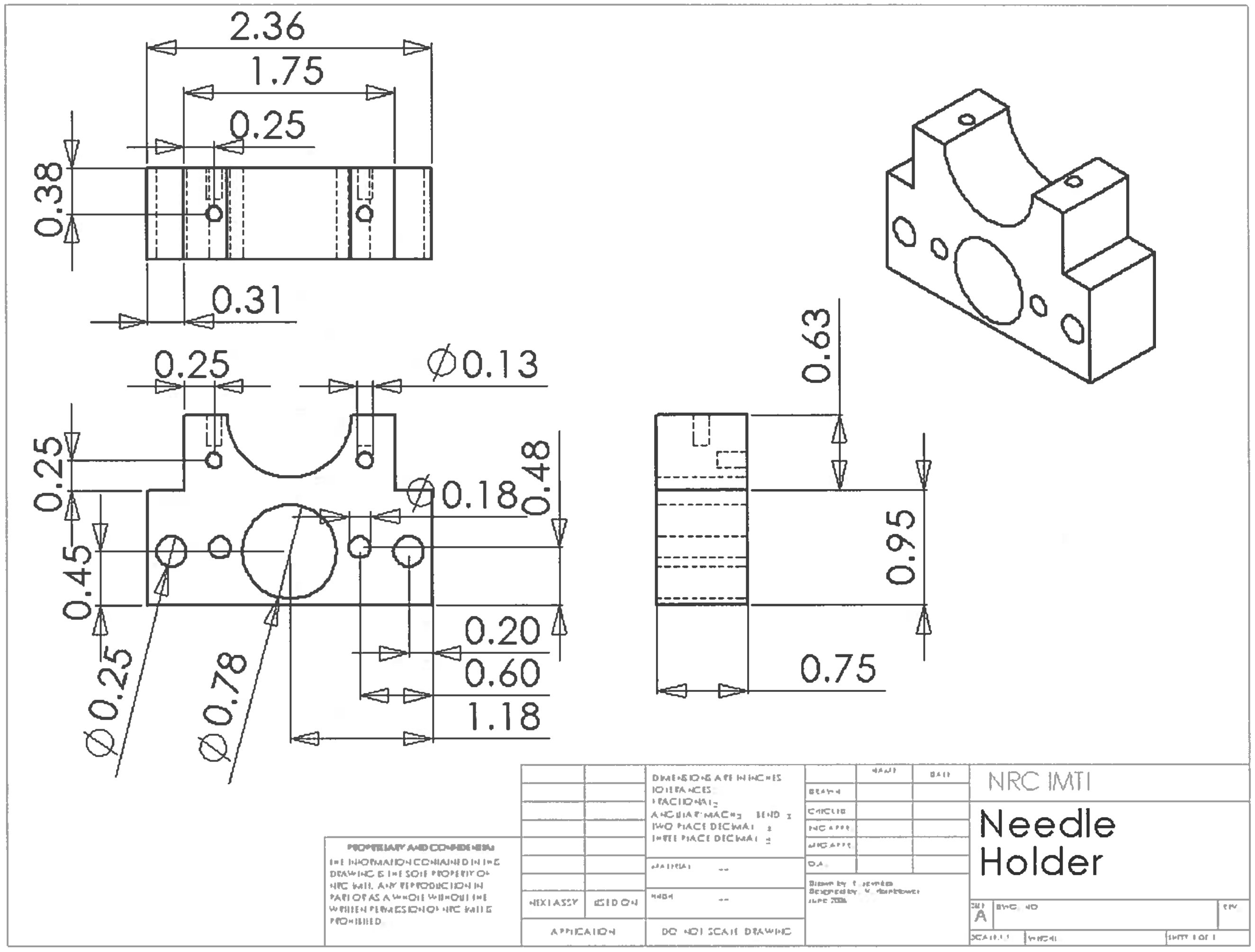

Ø0.13
Ø0.13
0.25
0.25
0.25
2.36
0.30
1.18
1.18
0.49
Ø0.17
(2.77)
0.50
NRC IMTI
mold support
DIMENSIONS ARE IN INCHES
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL

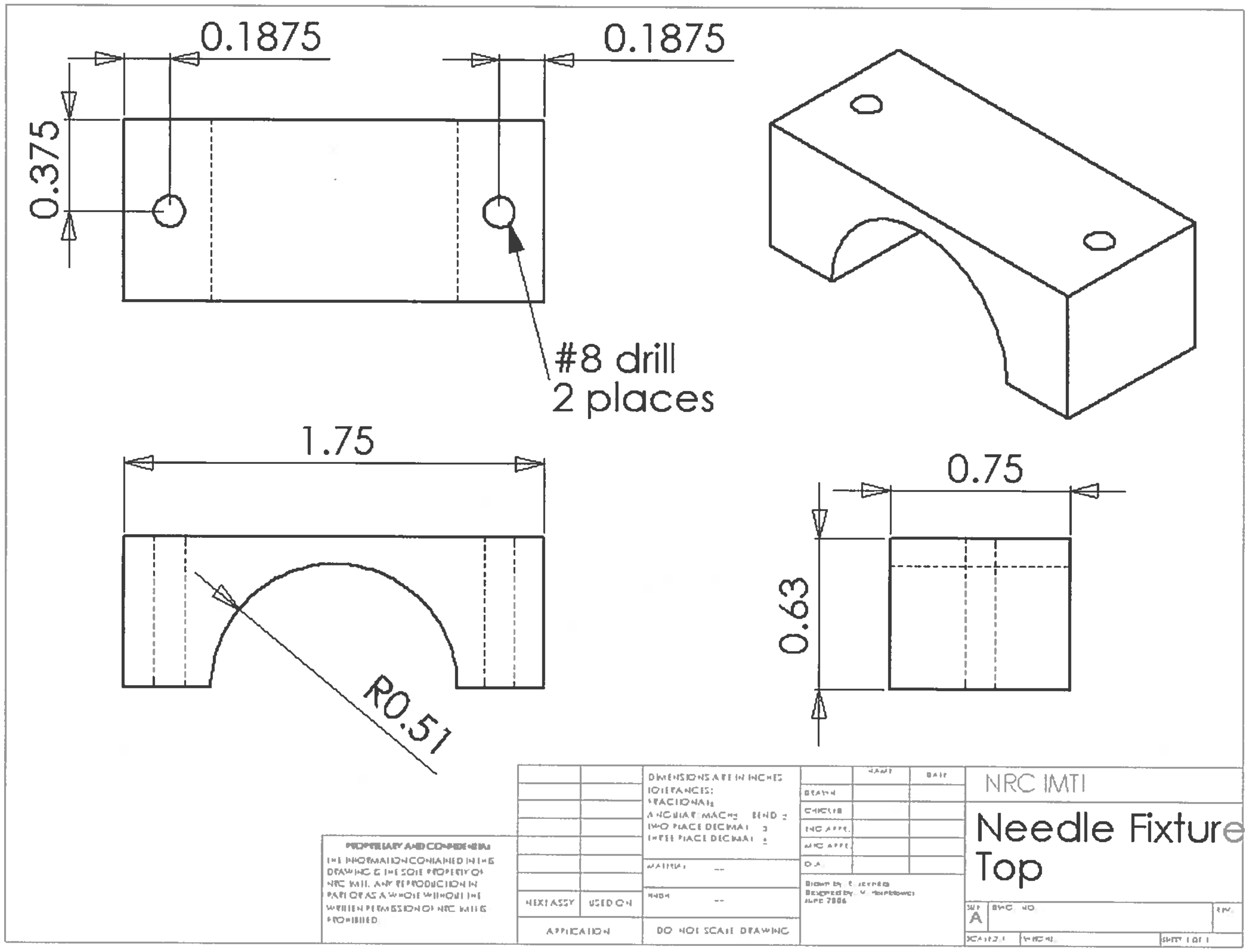

(63.42)
(43.47)
(Ø38.08)
(25.24)
(Ø4.25)
(67.08)
(83.15)
NRC IMTI
PVA Mould
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
June 2006
SIZE
A
DWG. NO.
REV.
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1

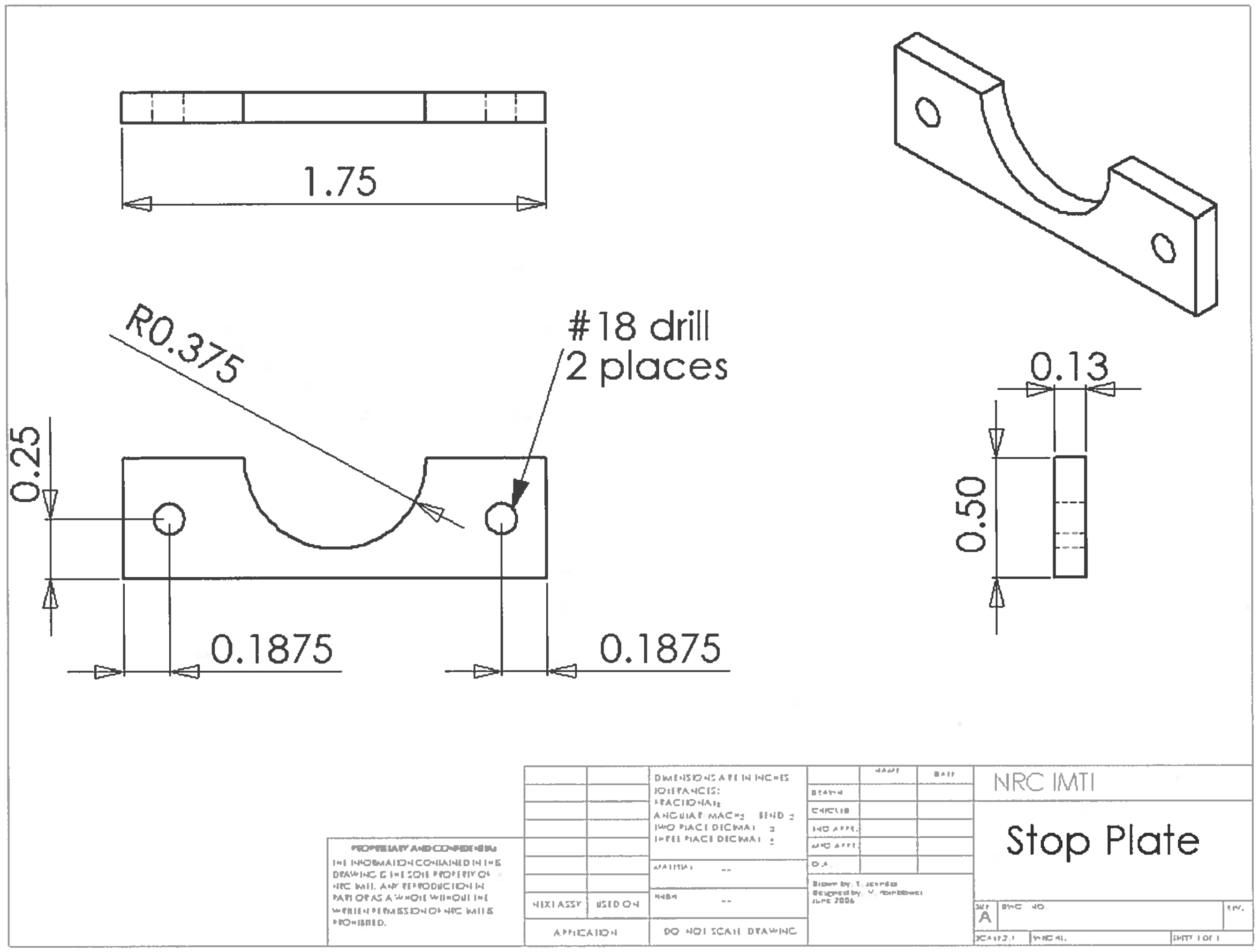

PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by: [illegible]
Designed by: [illegible]
June 2006
NRC IMTI
Slider Sub Assembly
SIZE A
DWG. NO.
REV
SCALE: 1:3
WEIGHT:
SHEET 1 OF 1

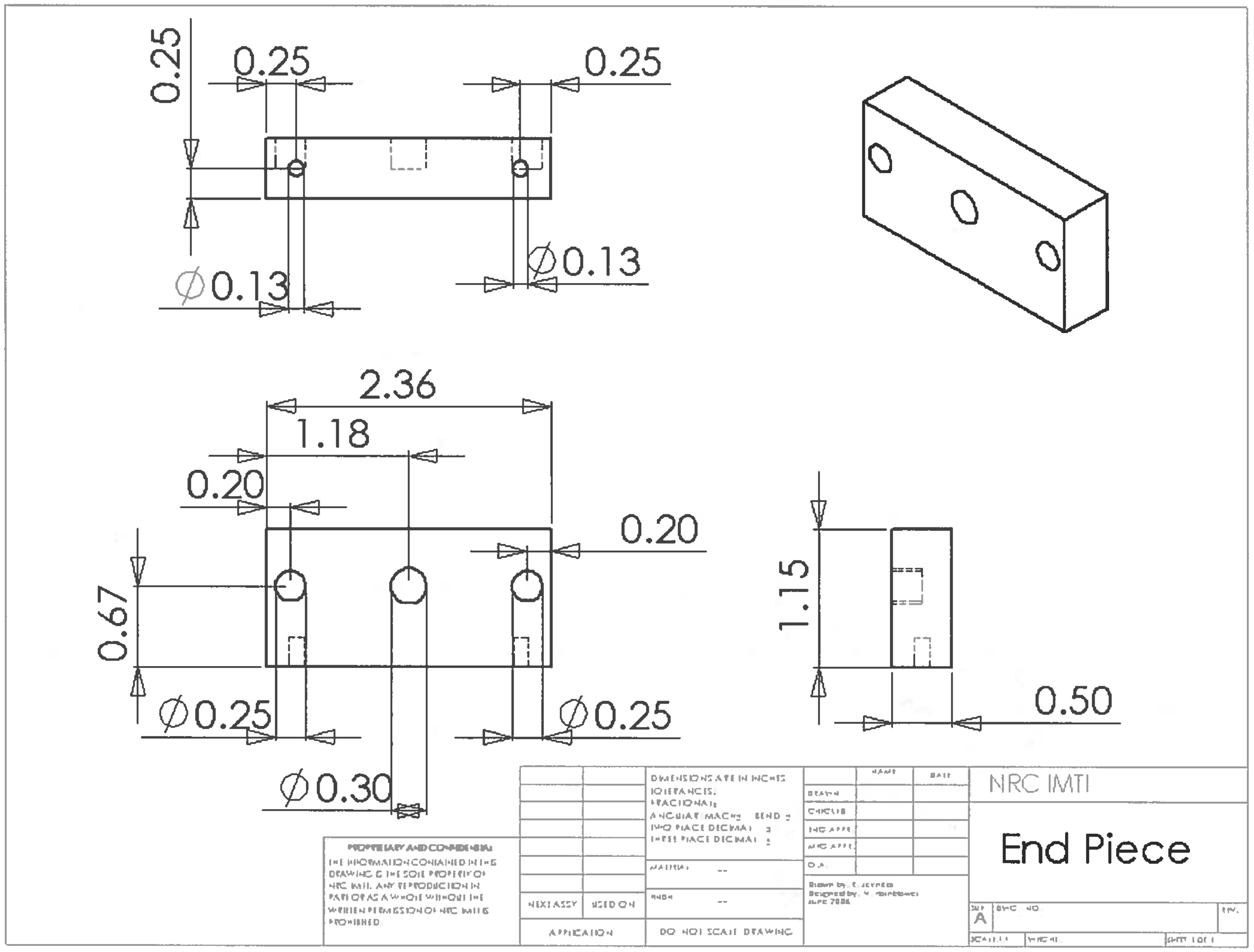

(60)
(12.70)
(25)
(Ø21.34)
(29.22)
NRC IMTI
Back End Piece
A

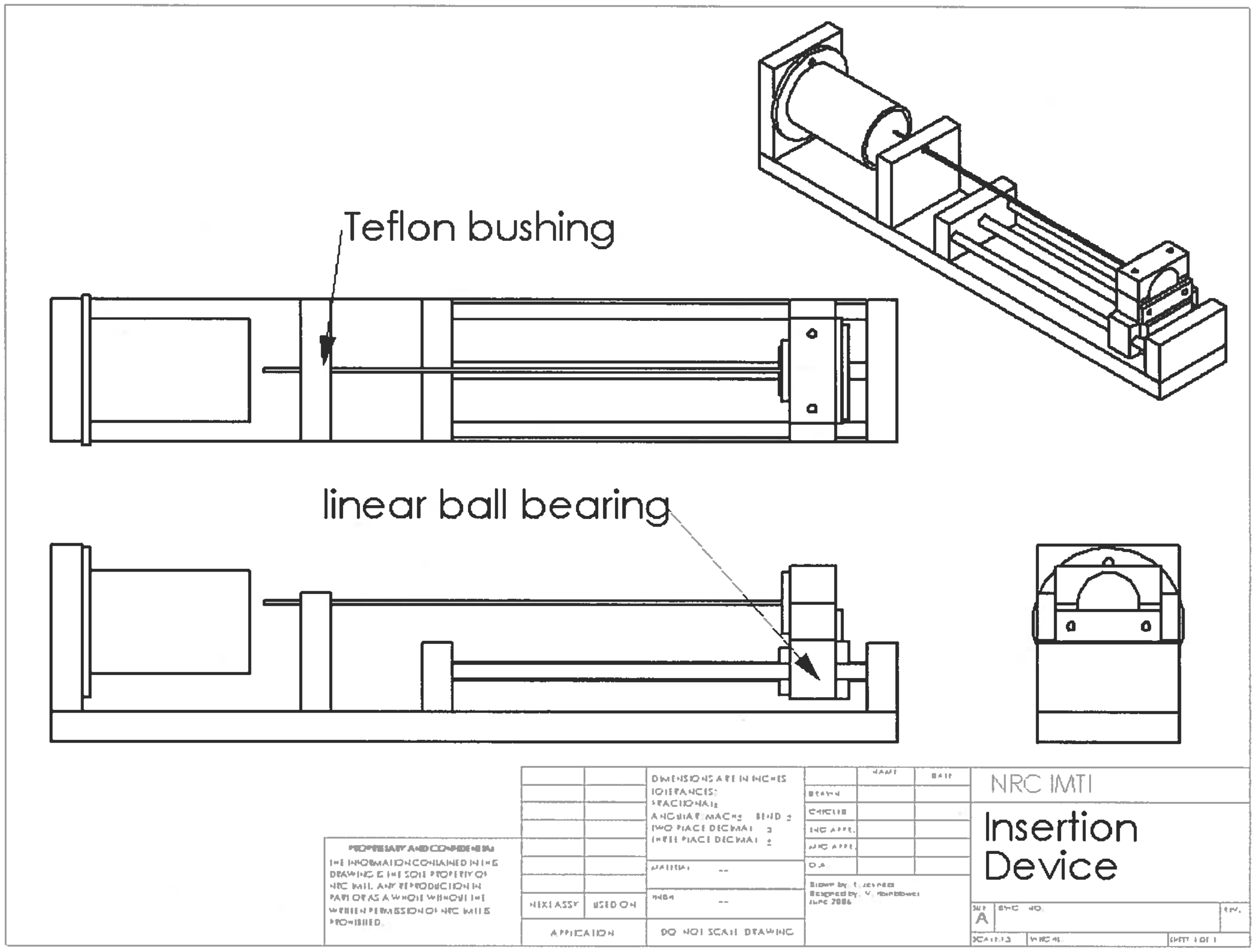

40.02
5.99
23.53
19.85
Ø4.50
Ø7.50
Ø19.85
2.94
1.44
24.22
NRC IMTI
Ball Bearing
A


2.36
1.75
0.25
0.38
0.31
0.25
∅0.13
0.25
0.45
∅0.18
0.48
0.20
0.60
1.18
∅0.25
∅0.78
0.63
0.95
0.75
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by [illegible]
Designed by [illegible]
June 2006
NRC IMTI
Needle Holder
SIZE A
DWG. NO.
REV
SCALE 1:1
WEIGHT
SHEET 1 OF 1


0.1875
0.1875
0.375
#8 drill
2 places
1.75
R0.51
0.75
0.63
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
Drawn by
Designed by
June 2006
NRC IMTI
Needle Fixture
Top
SIZE
A
DWG. NO.
REV
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1


1.75
R0.375
#18 drill
2 places
0.25
0.1875
0.1875
0.13
0.50
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
MATERIAL --
FINISH --
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
June 2006
NRC IMTI
Stop Plate
SIZE
A
DWG. NO.
REV.
SCALE
WEIGHT
SHEET 1 OF 1


0.25
0.25
0.25
Ø0.13
Ø0.13
2.36
1.18
0.20
0.20
0.67
Ø0.25
Ø0.25
Ø0.30
1.15
0.50
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
NRC IMTI
End Piece
SIZE
A
DWG. NO.
REV.
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1


Teflon bushing
linear ball bearing
NRC IMTI
Insertion Device
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
MATERIAL --
FINISH --
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF NRC IMTI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF NRC IMTI IS PROHIBITED.
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
June 2006
SIZE
A
DWG. NO.
REV.
SCALE:1:3
WEIGHT:
SHEET 1 OF 1

Appendix B – LabView Software Diagrams

Control and data collection software for the Needle Insertion Test bed was created using the LabView graphical programming environment. All the logic of the software is captured in the diagrams shown in the following pages.

Some software functions cannot be shown in a single diagram, i.e. the components and labels would be too small to read. These are instead shown as a sequence of diagrams, usually panning the whole diagram from left to right.

Some software contains "case" components to designate options. If the other case(s) is (are) non-trivial, it will be shown in a subsequent diagram.

One software function, i.e. the display of the NRC copyright, is so simple that the diagram for this function is not shown.

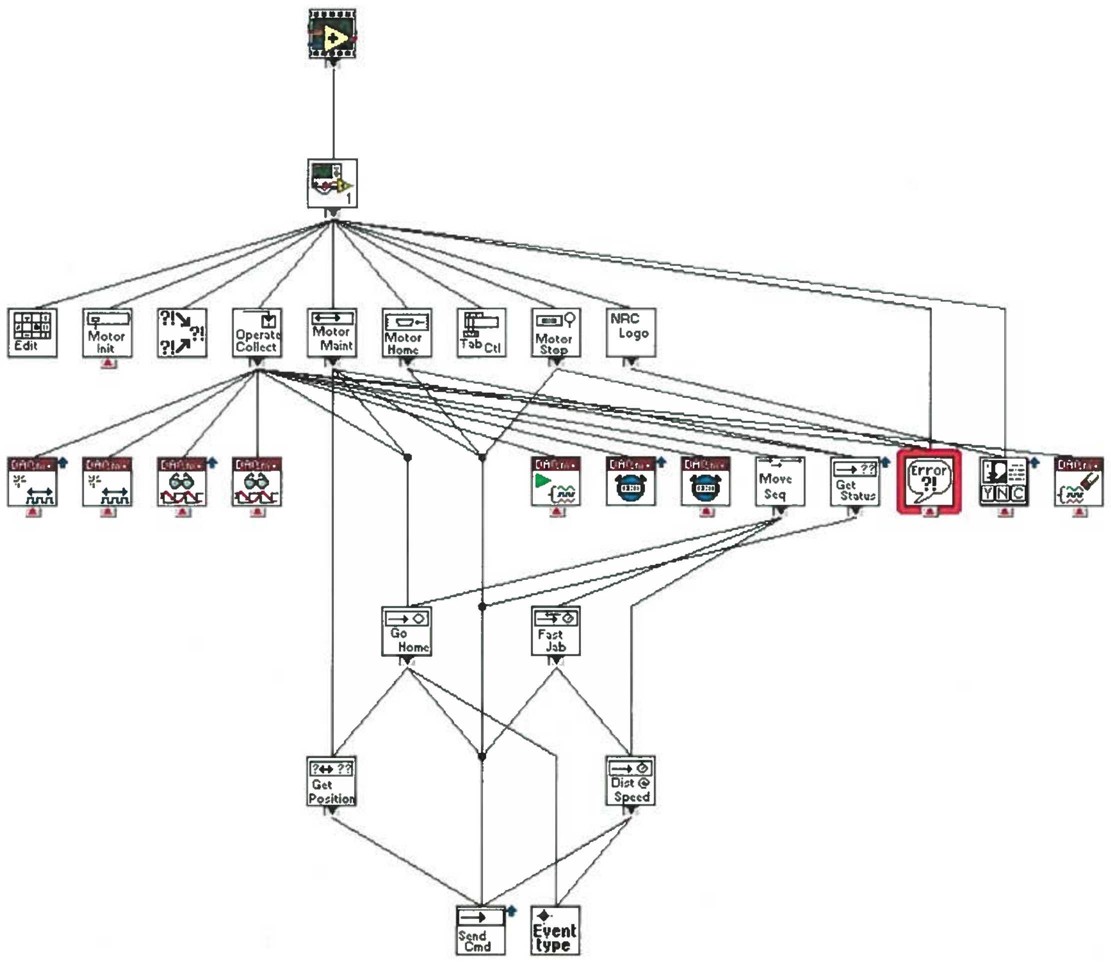


Figure 11 Hierarchy of the Needle Insertion software (main app. is 1)

Shown in Figure12 is the start of the main sequence of the application. The application will execute a couple of steps of start up, then loop while the operator selects between the three tabbed options, and then it will stop because of an error or because the user pressed the button.

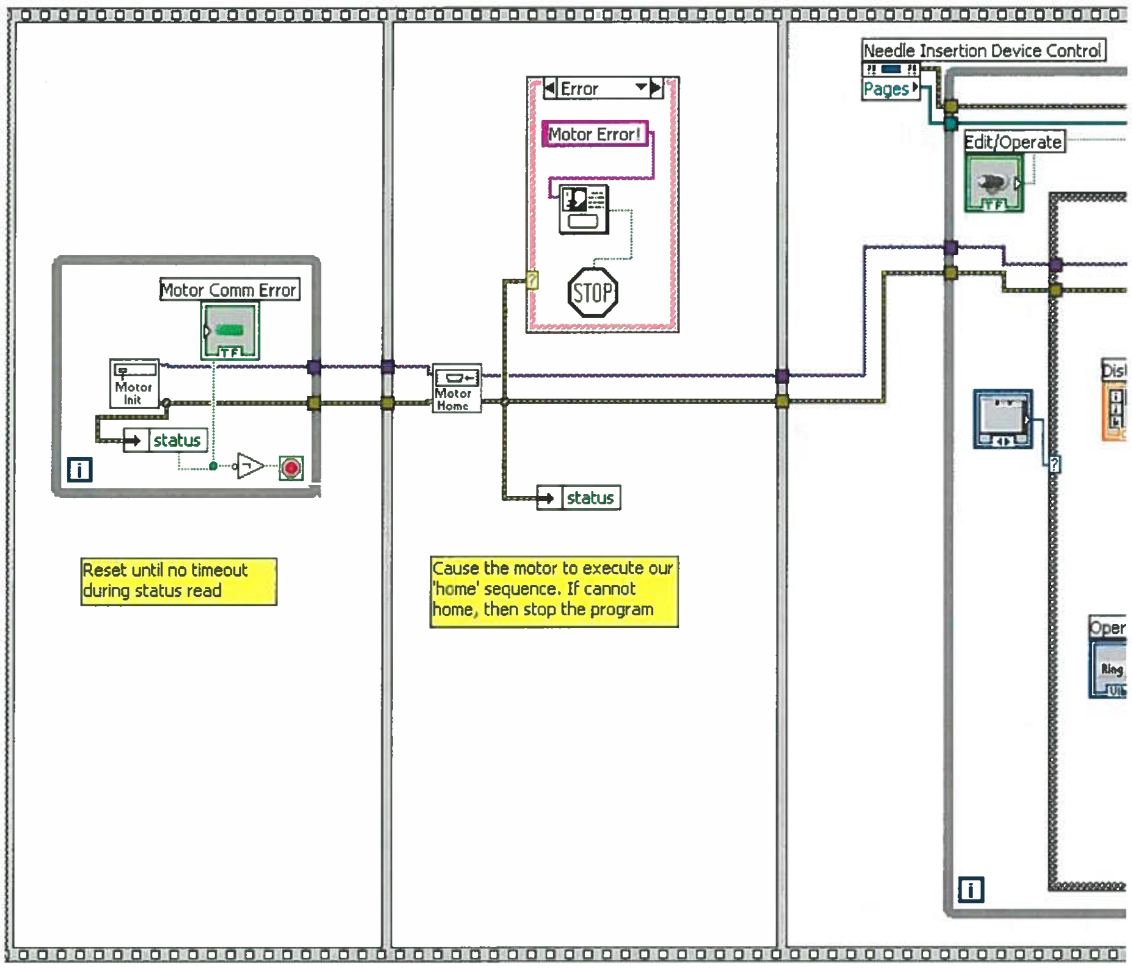

Motor Comm Error
Motor Init
status
Reset until no timeout during status read
Error
Motor Error!
STOP
Motor Home
status
Cause the motor to execute our 'home' sequence. If cannot home, then stop the program
Needle Insertion Device Control
Pages
Edit/Operate
Disl
Oper
Ring

Figure 12 Main program - Start up

The first step in the sequence is the check for motor status. This check will be part of the "Motor Init" subprogram (which will be described later). If the initialize subprogram returns an error, then the "Motor Comm Error" light will illuminate and the software will continue the loop and continue to try to establish motor communications. If there is no error, the software can proceed to the second frame of the sequence.

The complex "Motor Home" subprogram will now execute. If there is an error return, the program will display a message and halt the application. If there is no error, the program will proceed to the next frame of the sequence, which will be the tab selection and one of the three modes of the application.

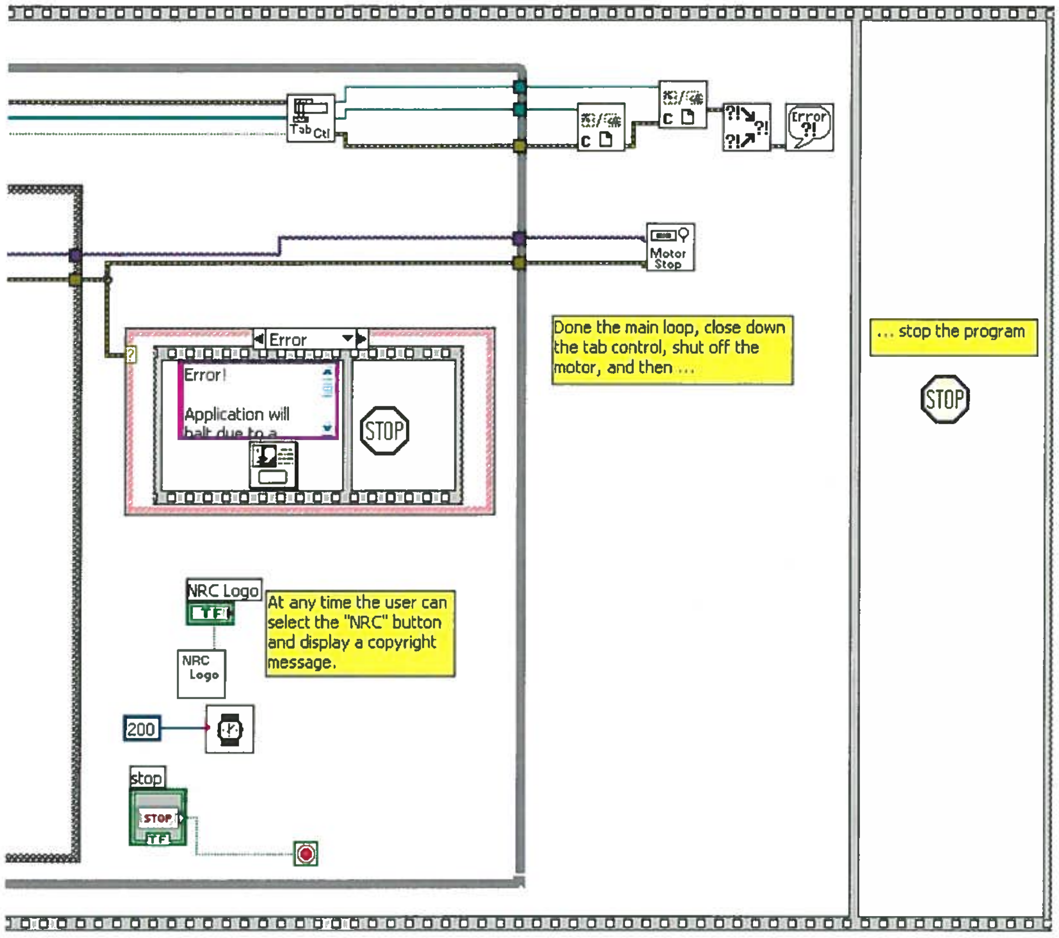

Tab Ctl
Motor Stop
Error
Error!
Application will halt due to a
STOP
Done the main loop, close down the tab control, shut off the motor, and then ...
... stop the program
NRC Logo
NRC Logo
At any time the user can select the "NRC" button and display a copyright message.
200
stop
STOP
TF

Figure 13 Main program – Stop

Within the main application loop, checked every 200 ms., the operator can press the "NRC" button. This will cause the "NRC Logo" subprogram to run, which simply displays a copyright message. The user will "OK" the message to have it cleared from the display.

A subprogram called "Tab Ctl" (tab control) will execute every cycle of the loop. This sub-function hides the logic for enabling and disabling two of the tab options. A full description of this function is provided in this appendix.

Any error from the motor in any one of the three modes of operation of the program will cause the display of a message for the user. Once this message is acknowledged, this error-handling mini-sequence will halt the application. The motor will already be disabled by the motor function which first detected the error.

If the "Stop" button is pressed in the main loop, the loop will terminate. This will cause the "Motor Stop" subprogram to halt and disable the motor. The tab control processing will be shut down (and any error message displayed) and then the application can precede to the last frame of the sequence and halt the application.

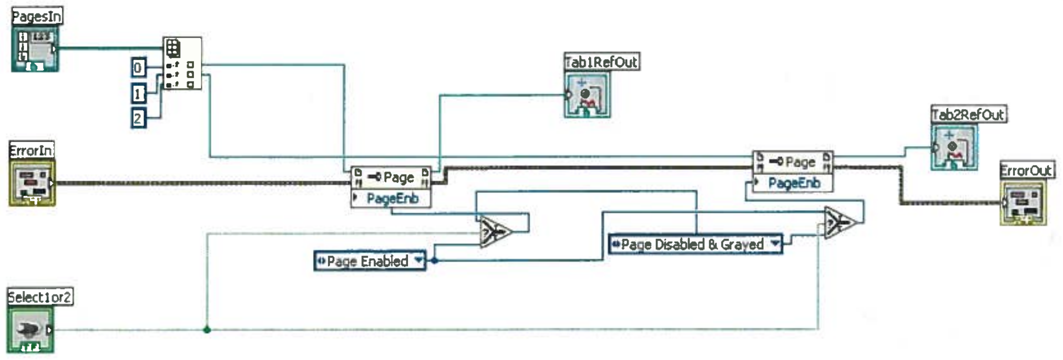


Figure 14 Main program – Tab control subprogram

This subprogram will expect an input which is a set of pages grouped as a 'tab control'. Each 'page' is an independent collection of user interface controls and indicators. The software will separate the three (in this application) pages and then extract a 'property node' of the application's tab control. Specifically, this is the property which controls if the individual tab is enabled or not. The selector switch input will set the 'enabled' state for tab options 0 and 1. Tab option 2 (the manual control of the motor) is not modified and is therefore always enabled. The remaining two tab options will alternate between 'enabled' and 'disabled and grayed'.

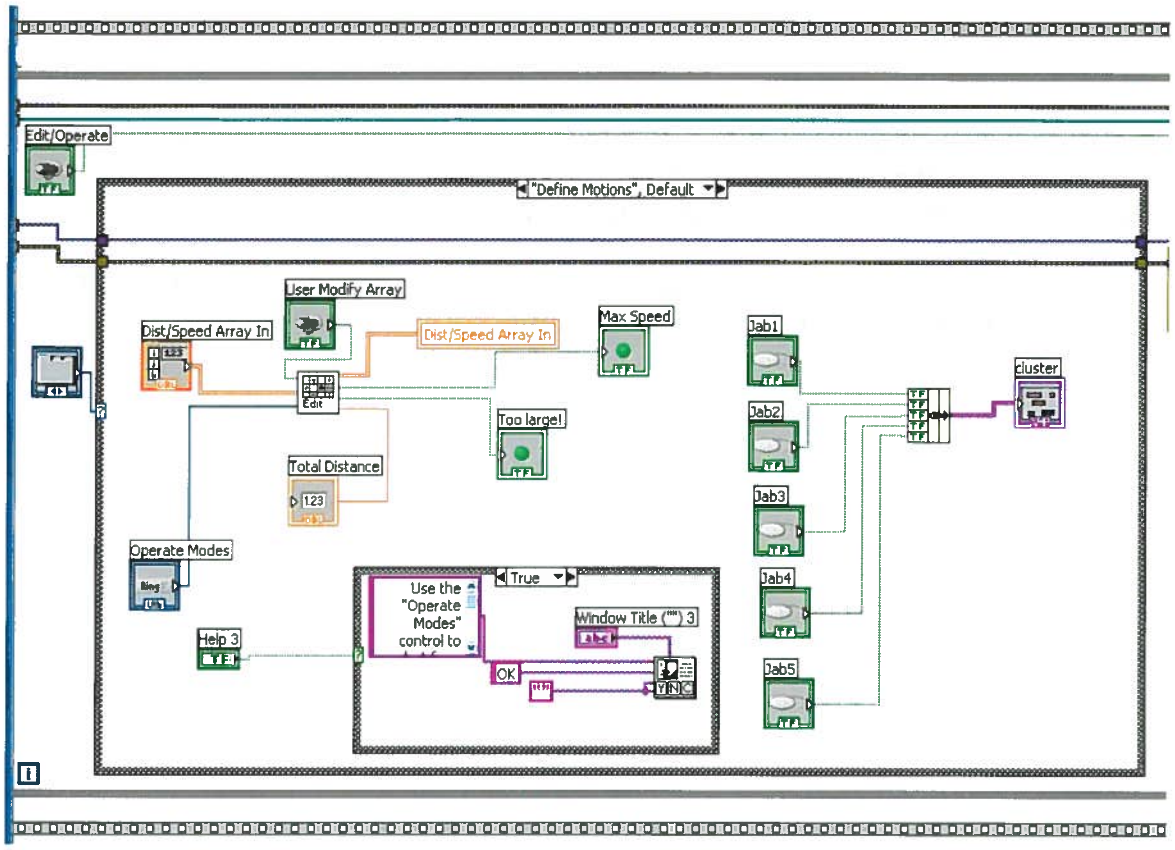


Figure 15 Main program – Define motions tab

Inside the main program's loop is the case (a choice or selection) function driven by the tab selection of the user. When the "Define Motions" tab is selected, and the related controls are shown and enabled, then the logic shown in this diagram will operate.

On the bottom is the help button ("Help 3"). If the help button is pressed the message will be displayed using a standard LabView dialog. The user selects "OK" on the dialog to dismiss the message. If the help button is not pressed, i.e. the other case option, then nothing happens.

The column of five (5) jab motion switches is collected into a cluster of Boolean values. This makes it easier to send the information to subprograms as a single input. A local variable is formed from this cluster and will be shown and used in other parts of the main program.

The "Edit" subprogram will use the selection from the predefined "Operate Modes" selection to place values in the five lines of the distance/speed/dwell array. The array is used as both an input to the subprograms and (a 'local variable' of the array) as an output. This will allow changes in the selected array to be forced into the array displayed on the user interface of the program. Values in this array cannot be changed.

Switching the "User Modify Array" control switch will disconnect the displayed array from a fixed, pre-defined set of values to the current values from the software display.

Now these values can be modified and user changes will not be reset by the software loop.

The displayed set of array values is incorporated in the "Dist/Speed Array In" component. Another local variable copy of this component was made and is used by the insertion and data collection function of the main program.

If the array values are not valid then the subprogram will turn on Boolean outputs. These outputs will illuminate light displays for this part of the user interface.

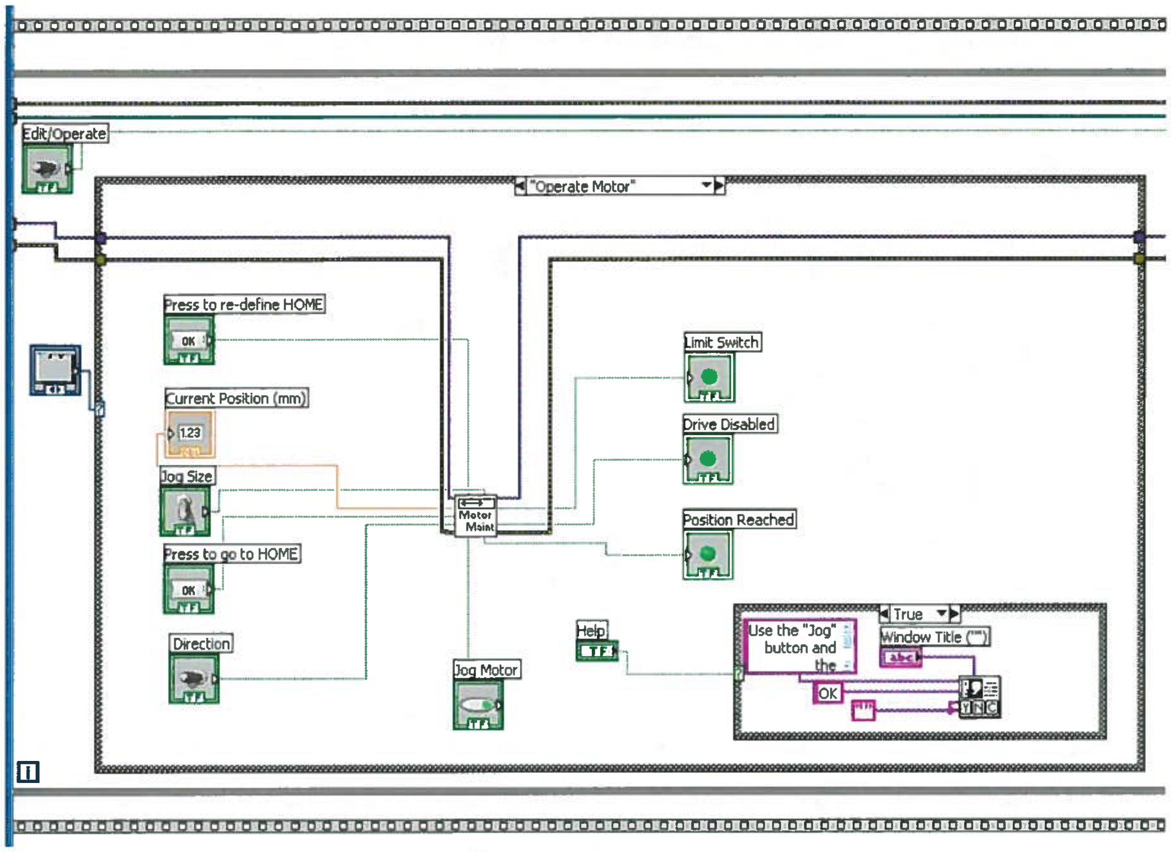


Figure 16 Main program – Manually operate motor

This tab option contains only simple components (buttons, switches, and lights) without any references or local variables. One button will cause the help message for this option to be displayed.

The other buttons and displays are connected to the subprogram. A subprogram was created in this case to simplify the complexity of the main program and reduce the size of the program's connection diagram.

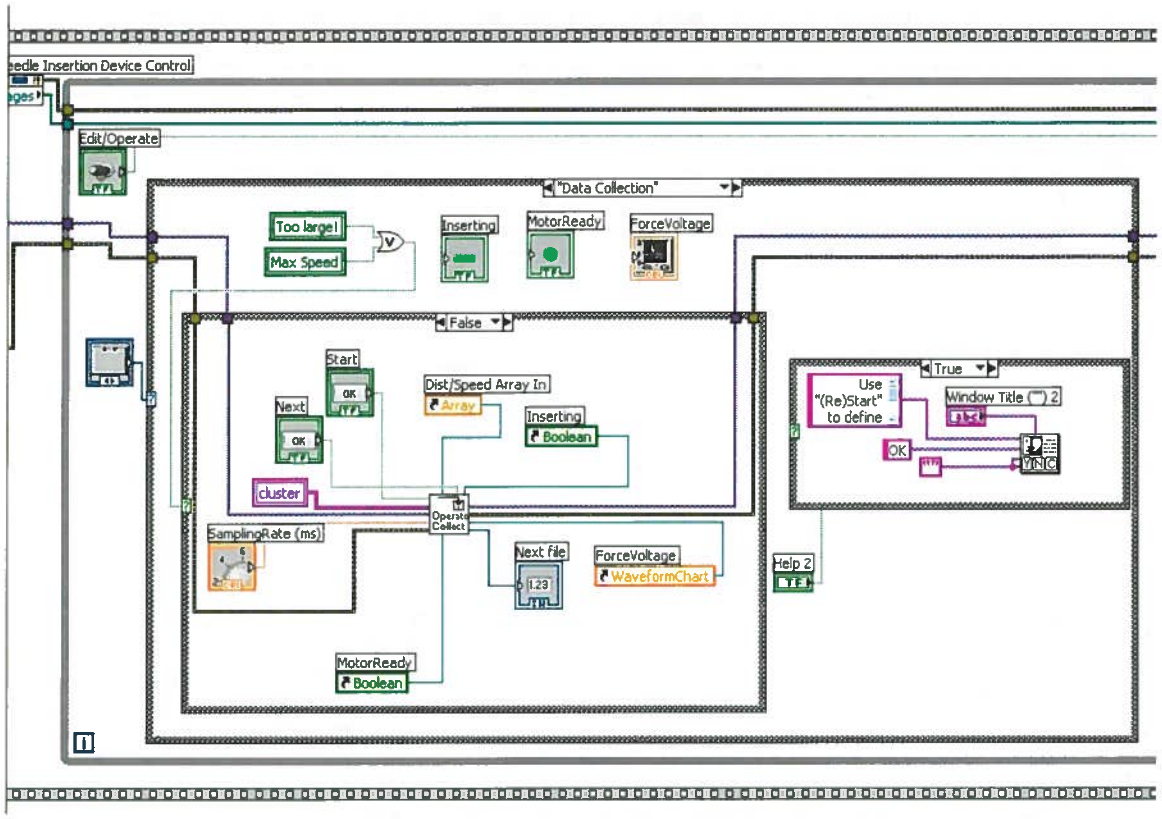


Figure 17 Main program – Sequence motor and data collection

Figure 17 shows the logic for the main purpose of the application, which is to move the needle as defined by the "Dist/Speed Array In", and to collect needle force data while the insertion proceeds.

The motion array and the 'jab' Boolean cluster are passed to the subprogram as inputs. There are also buttons whose state is passed to this subprogram, i.e. the "Start" and "Next" buttons. The data sampling rate is read from the user interface dial and also passed to the subprogram. The number of the next data file is passed back so that it can be displayed.

Three components of this part of the program have no visible connection to other components. Two indicator lights, and the displayed chart for force data, have been copied as a 'reference' or pointer to the respective component. These 'references' are passed down to the subprogram so that it can directly set the state of the lights and write data to the chart. Without this technique the lights could only be changed after the entire subprogram was complete. Without a reference to the chart a large stream of force data would be passed up from the subprogram, and chart's display would change all at once at the completion of the subprogram without any intermediate update.

The "Help 2" button will display an informative message to the user, i.e. similar in operation to the help button in the 'edit array' tab option.

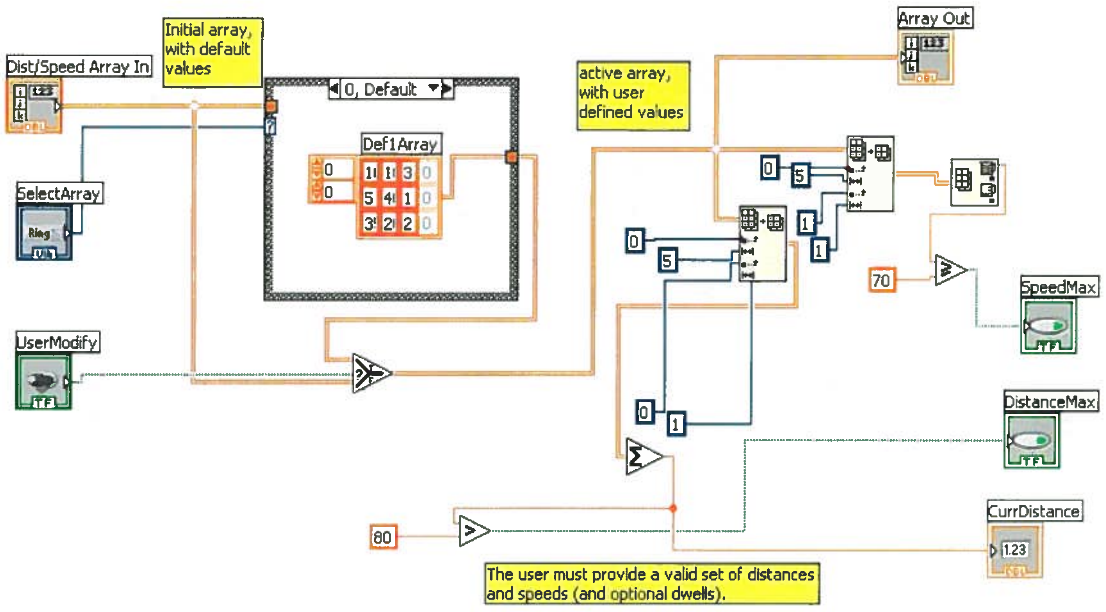

Initial array, with default values
Dist/Speed Array In
0, Default
Def1Array
SelectArray
UserModify
active array, with user defined values
Array Out
SpeedMax
DistanceMax
CurrDistance
70
80
The user must provide a valid set of distances and speeds (and optional dwells).

Figure 18 Main functions – Edit array subprogram

A central component of this software is the case statement box. There is one 'case' for each option in the "SelectArray" list, but each option is similar to the one shown. A predefined array of distances, speeds and dwell times is selected by the case statement to be sent to the selection module. As long as the "UserModify" Boolean is "True" then the output of the selection and the displayed array values are always set to the constant values provided by the case option, i.e. the values cannot be changed.

When the Boolean input is switched to "False", then the new set of values for the array will depend only upon the current values in the display. Thus the user can select and edit any value(s).

The current set of values in the array is checked against limits. If any speed (column 1) in the array exceeds the 70 mm/sec. limit in the diagram, then the output "SpeedMax" will be set "True". If the simple sum of the distance values (column 0) exceeds the 80 mm limit, then the "DistanceMax" output will be set as "True". This simple sum is also provided to the calling program for it to display.

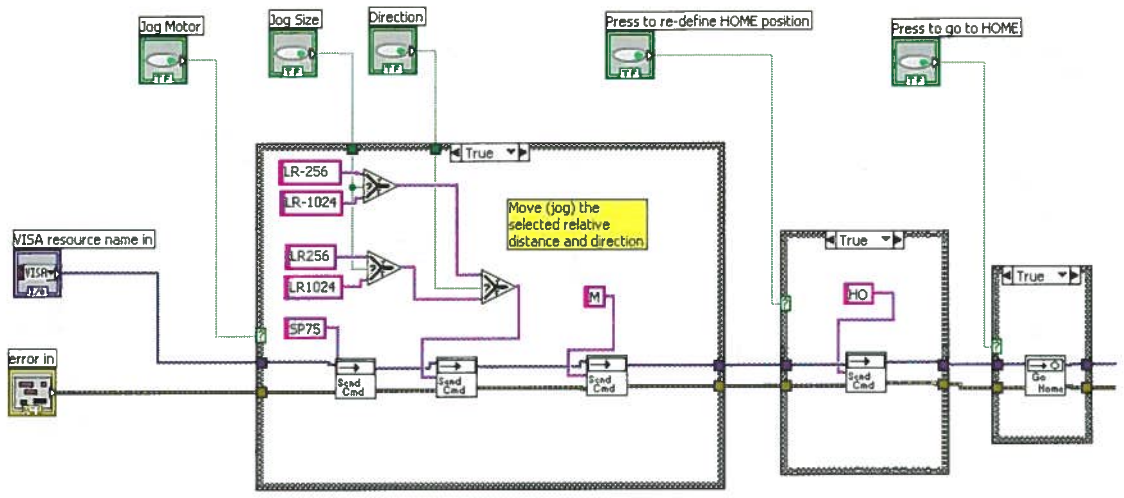


Figure 19 Main functions – Manual motor control – Part 1

This function will execute a sequence of checks and updates each time it is called by the loop in the main program.

The first frame in the sequence will execute a motor movement, but only if the "JogMotor" input is set to "True", i.e. the button is pressed by the user. If the input is "False", the there is no reaction and the signals are passed to the next frame. Inputs for size and direction will, respectively, control the value and the sign of the relative movement.

If the function has the "Home" input set as "True" the subprogram will issue the command to reset the 'home' location for the motor.

Another input to the subprogram controls whether the "go to home" motor function is called.

Subprogram execution is then continued by the next sequence frames, shown in the following figure.

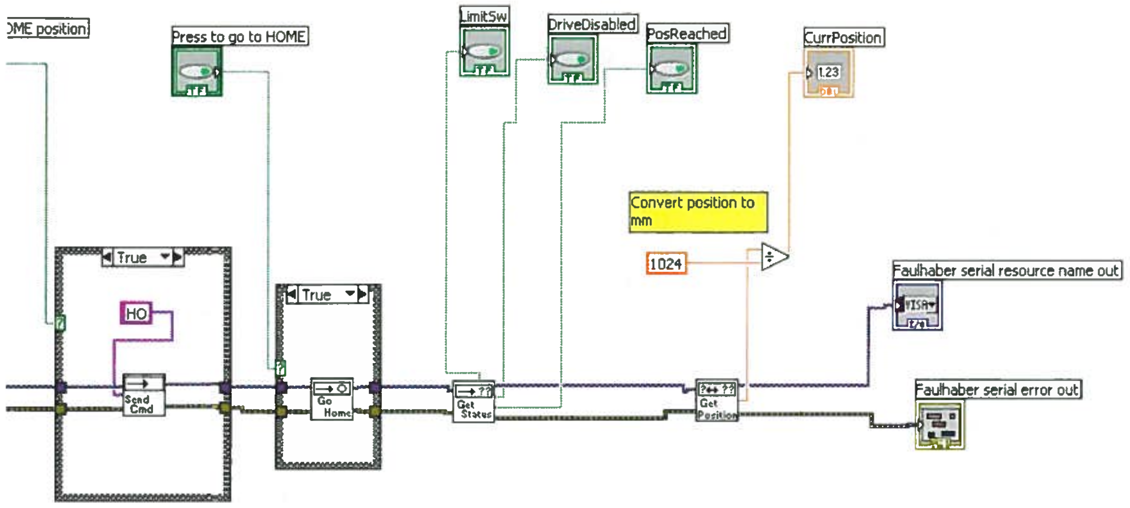


Figure 20 Main functions – Manual motor control – Part 2

The next subprogram sequence to execute would be the "Get Status". This routine will set three output values (Booleans) based on the current motor status as read from the controller.

The final sequence frame in this subprogram will read the current position information from the motor controller and convert the value into millimetres of distance from the zero location. Note: The application uses a conversion factor of 2048 position values for one rotation of the motor, and 2 millimetres of linear distance for the traveller for one motor rotation.

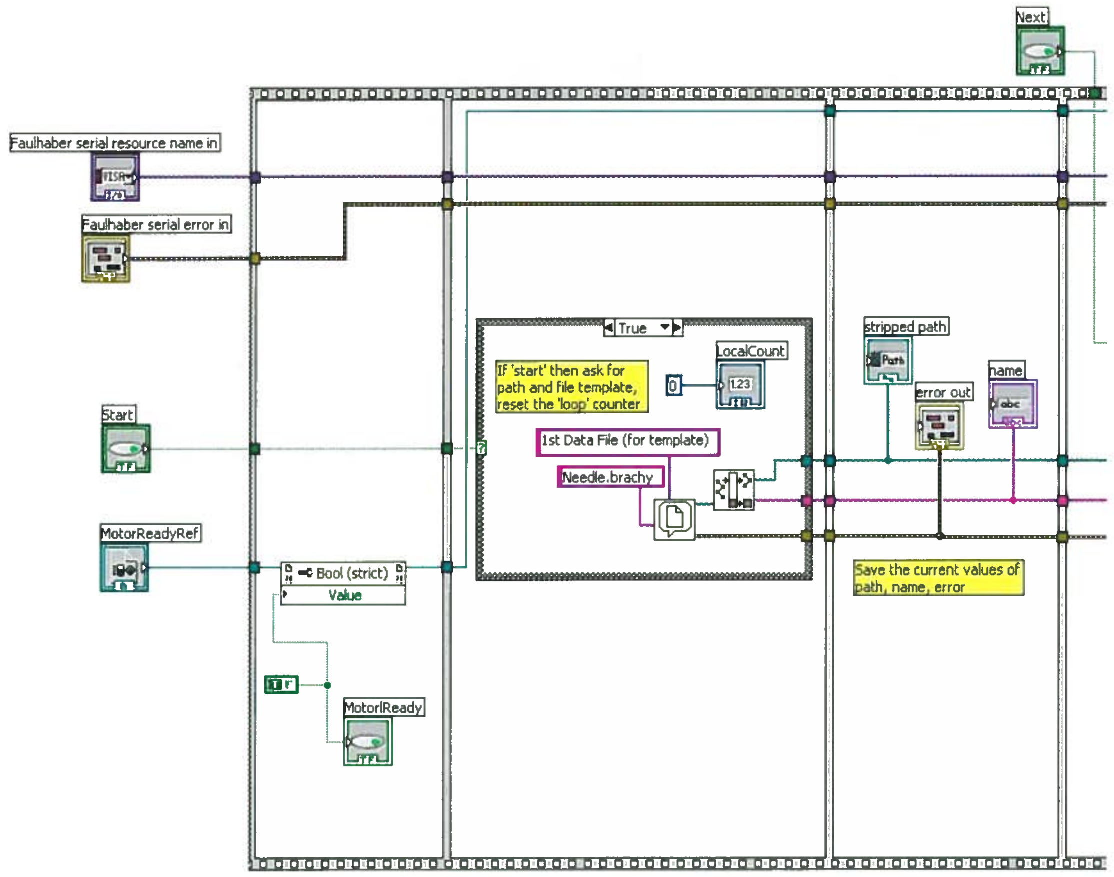

Next
Faulhaber serial resource name in
Faulhaber serial error in
Start
MotorReadyRef
Bool (strict)
Value
MotorlReady
True
If 'start' then ask for path and file template, reset the 'loop' counter
LocalCount
1st Data File (for template)
Needle.brachy
stripped path
error out
name
Save the current values of path, name, error

Figure 21 Main functions – Manual motor control – Part 2

This function will cause the motor to execute the current array of motions and collect data during the needle insertion part. Again, the program is defined as a sequence to force an order on the operation and to make sure that multiple operations inside one sequence box all complete before the operations in the next box are started.

The first frame in the sequence will simply set a local Boolean display to indicate "MotorReady". Since this "True" value is also wired to the "value" property of the reference input, the main display's motor ready light will also be illuminated.

In the second frame of the sequence the "Start" input is checked. If set "True" then the subprogram will execute a dialog to ask the user for a folder location and a template name for the data file. File location (path) and file name are separated so that the name can have a file number appended to it. File information is displayed in the local interface for the subprogram so that local variables can be formed for each piece. If the "Start" input is not set, then the other case, shown in Figure 22, will be executed and the current (local variable) values of these three components will be used for the remainder of the subprogram's execution.

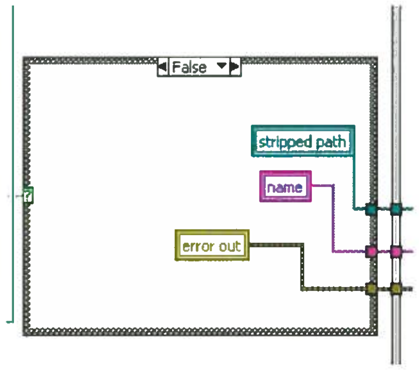


Figure 22 Start button - False case

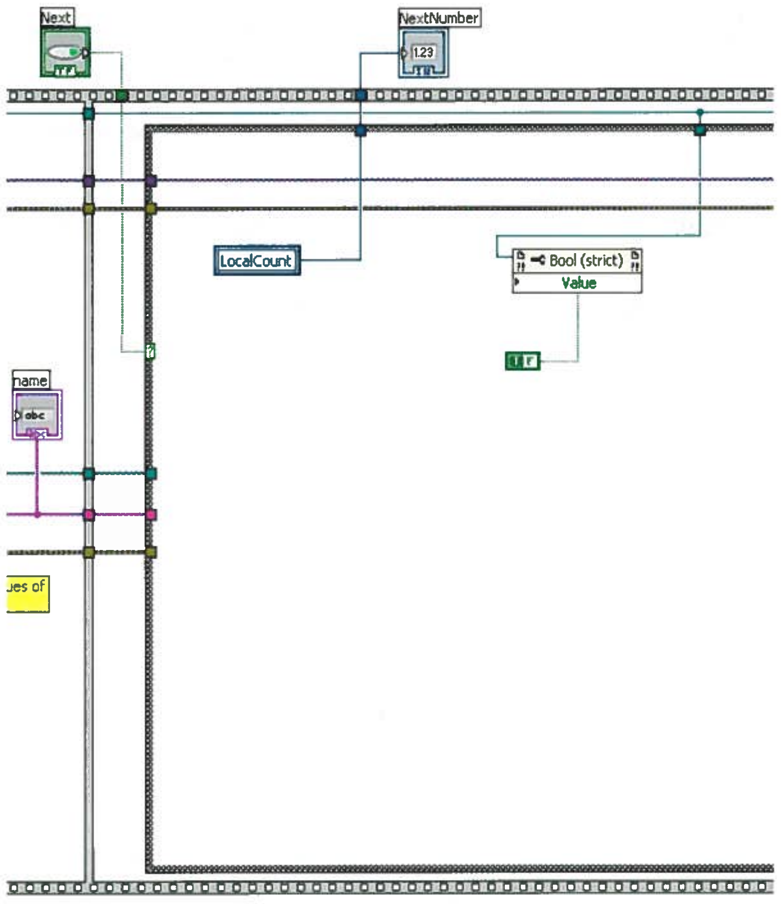


Figure 23 Main functions – Manual motor control – Part 3 - False case

If the input “Next” is false, then there is not much for this subprogram to do. The current value of the ‘next data file’ counter is sent as an output, and the value of the main programs “MotorReady” light, obtained from the reference input, is maintained as “True”.

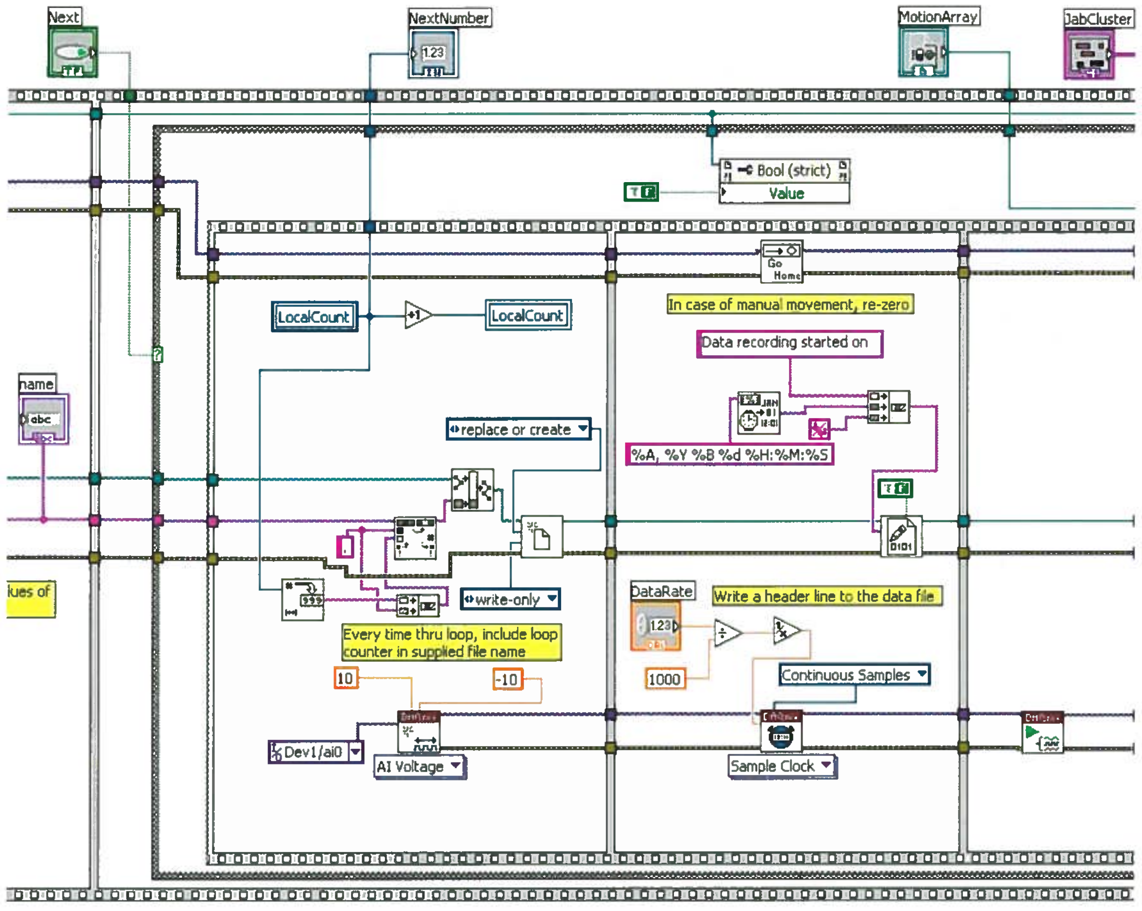


Figure 24 Main functions – Manual motor control – Part 3 - True case

When the "Next" control has been selected to request another insertion and data collection, the case box will switch to the more complex "True" state. The reference to the "MotorReady" light is used to extinguish the light, since the motor is about to move.

Another sequence structure is used to ensure that several parts of the program are all complete before the following operations can be started. In the first frame of this internal sequence the number attached to the template file name will be increased by one. The old, current, number will be appended to the file name and the new name will be merged with the folder path. This complete file location is used to open a new data file. This file will be opened "write only", and will replace any existing file of the same name.

Finally in this first fame of the sub-sequence the data acquisition hardware is activated and set to sense analogue voltage input.

When execution of the subprogram passes to the second frame of the sub-sequence, the motor will execute a move back to the home or zero location. If the motor is already at this location, the motor will do nothing.

A “header” line is written as the first line in the force data file. It contains a message about the date and time the file was created.

Finally this frame will use the data sampling “Rate” input value to define the data collection rate for the analogue force (voltage) input. Since the input value is provided as integer values of seconds, the subprogram must divide by 1000 to obtain a sampling rate in milliseconds.

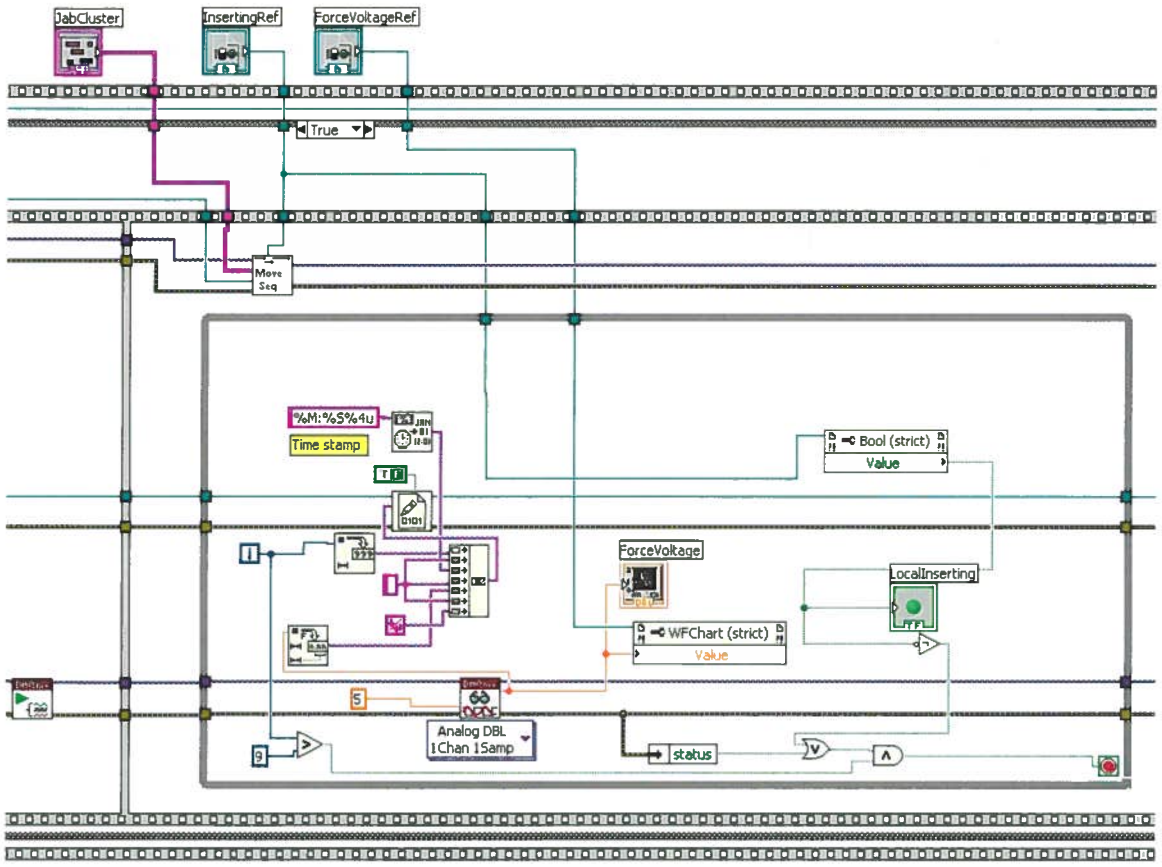


Figure 25 Main functions – Manual motor control – Part 4

The reference to the distance, speed, and dwell array is passed down to another subprogram. The cluster of 'jab' switch values is also provided to this new subprogram, and the reference to the "Inserting" light in the main program is passed through so that this new subprogram can control it. It will only be "True" while the motor and needle are executing the motions defined in the array. It will be "False" when the needle starts to withdraw.

The data collection loop can make use of this 'needle insertion' information, so the Boolean value of the indicator is extracted. One value of analogue force/voltage will be collected for each time this loop executes, and it will execute at the sampling rate defined in the previous sequence frame. At least ten (0 to 9) samples will be collected into the data file. Each line in the file will be written with a time stamp obtained from the computer, a sample number starting from zero (0), the value of the analogue input, and finally an "end of line" character combination.

As analogue data is obtained from the sensor, it is also used to update the chart in the main display via this chart's provided 'reference' value and 'value property'.

Data collection, i.e. the data acquisition loop, will continue beyond the minimum ten samples if there is no measurement error, and if the state of the "LocalInserting" signal remains “True”. Typically there will be hundreds of data samples collected as the motor executes the complex motion sequence requested by the operator.

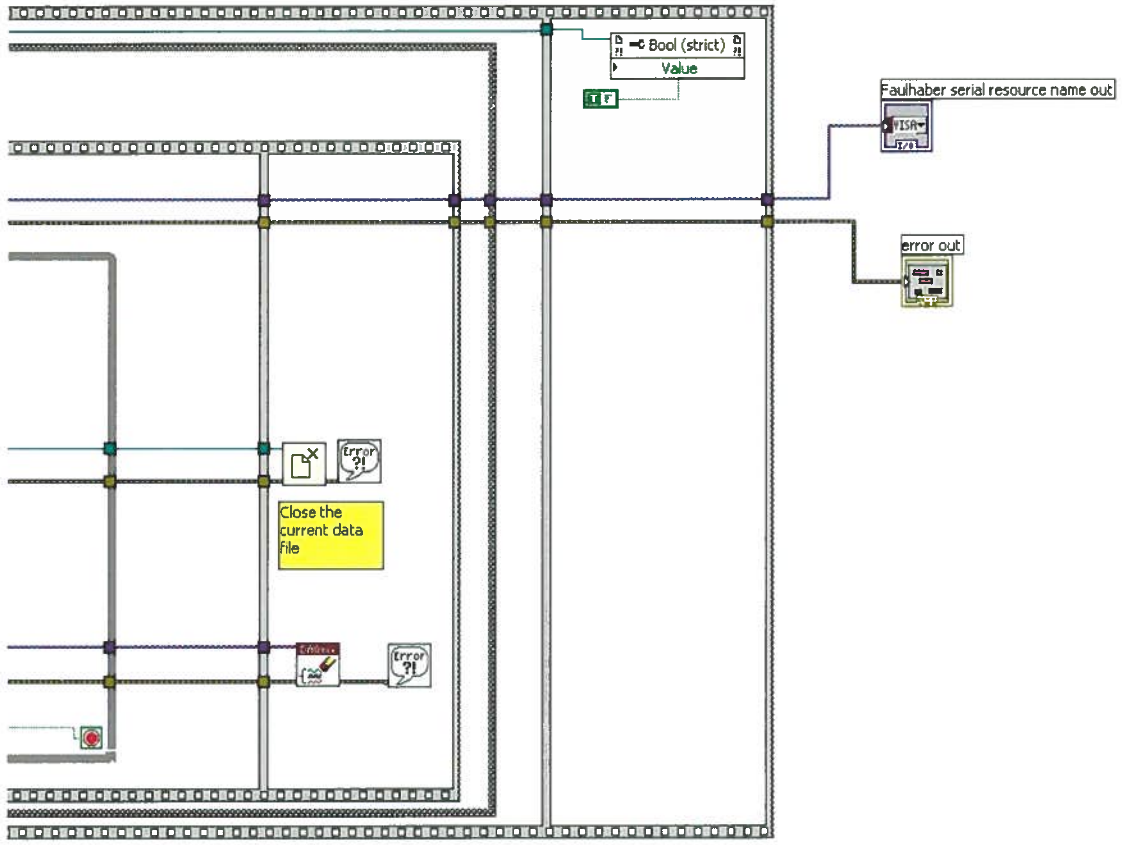

Bool (strict)
Value
T F
Faulhaber serial resource name out
VISA
error out
Close the current data file
Error ?!
Error ?!

Figure 26 Main functions – Manual motor control – Part 5

After the data collection loop terminates, the sub-sequence also terminates by shutting down the data acquisition hardware and closing the data file. Errors from either of these operations would be displayed.

The final frame in the main subprogram sequence will illuminate the "MotorReady" light since the motor is correctly located back at the home position and is ready for another data collection cycle.

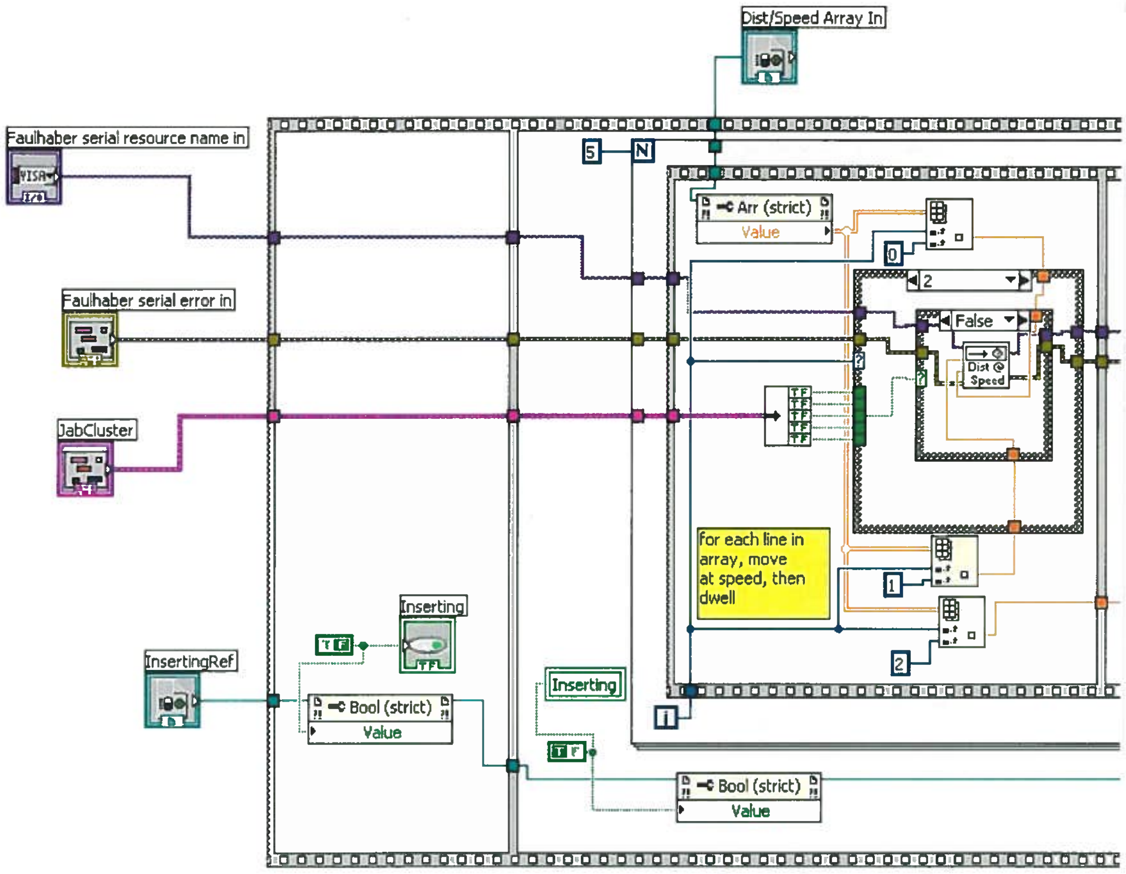


Figure 27 Main sub-functions – Move Sequence – Part 1

To move the needle through its defined motion sequence, the software will start the function in Figure 27. As the needle insertion motion has not started, the software will make sure that these indicators (the local light and the one referenced from the front panel) are extinguished.

In the second panel the logic will illuminate (turn "On") these indicators to identify that an insertion motion is running. A 'for' loop will process each line of the motion array, and this loop is assigned a count of five (5) to match the number of lines in the array.

Inside the 'for' loop is a sequence to support each part of the line of motion values. In the first panel of this sequence the software will extract the 0, 1, and 2 values from the array, which correspond to the distance, the speed, and the dwell time. The cluster of 'Jab' option switches has been passed down to this function, and it be used to define which type of motion to use. When the 'jab' option is "False", the standard constant motion is used. The needle will move over the defined distance at the defined speed. When the 'jab' option is "True", as shown in Figure 28, the software will use the same distance and base speed, but instead call a function to implement an alternative motion.

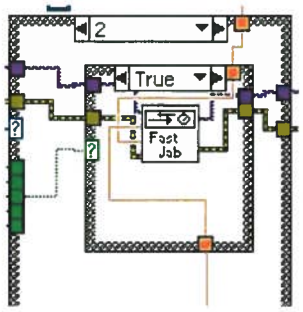


Figure 28 Fast jab motion option

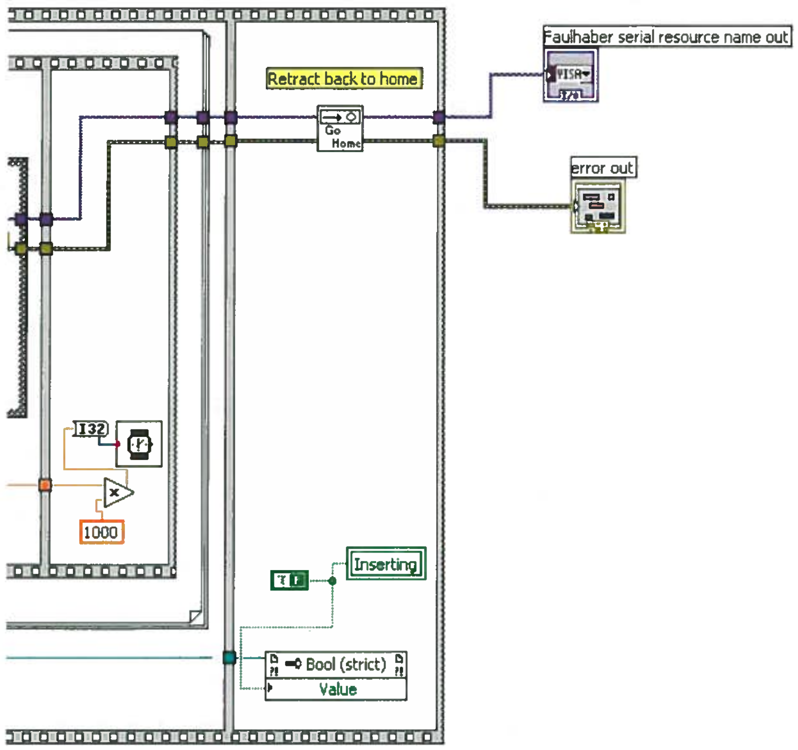


Figure 29 Main sub-functions – Move Sequence – Part 2

To complete the motion sequence function, the dwell time obtained from the array is used, in the second sequence frame of the 'for' loop, to wait the specified time. The array provided dwell times in seconds, so this must be converted to milliseconds before it can be used by the built-in LabView timer function.

The internal sequence and the 'for' loop are now complete, i.e. each line of the array has been processed.

One of the final actions of the function is to extinguish the "Inserting" indicators as the needle has completed its sequence and is about to withdraw.

The second final action of this function is to call a motor function to return the needle and motor to the home position.

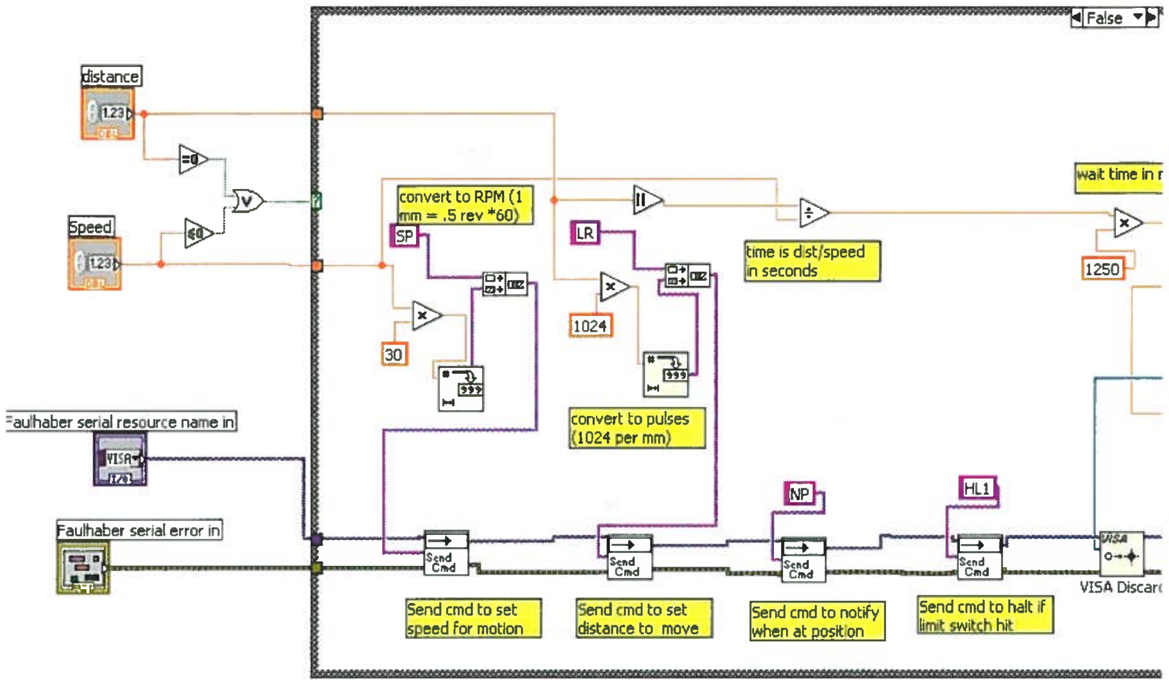

False
distance
Speed
Faulhaber serial resource name in
Faulhaber serial error in
convert to RPM (1 mm = .5 rev *60)
SP
30
LR
1024
convert to pulses (1024 per mm)
time is dist/speed in seconds
wait time in r
1250
NP
HL1
Send Cmd
Send cmd to set speed for motion
Send cmd to set distance to move
Send cmd to notify when at position
Send cmd to halt if limit switch hit
VISA Discard

Figure 30 Main sub-functions – Move a Distance at Speed – Part 1

A constant needle motion will result from a call to the function diagrammed in Figure 30. If either the distance input is zero or the speed input is less than or equal to zero, then there will not be any motion. The large case box which encompasses the whole function will switch to the "True" option (not shown), and the function will simply return.

The "False" option shown in the figure is the general case where the motor will move. The input distance must be converted to the number of motor (encoder) pulses, since this is the only distance that the motor controller will understand. Similarly, the needle speed must be converted to RPM. These values are sent to the motor as commands.

Also sent to the motor is a command to request a notification to the software when the final position is reached, i.e. to verify that the motion has been completed correctly.

The software will also include in every motion the command to halt if any limit switch is activated. This provides some safety during needle motions.

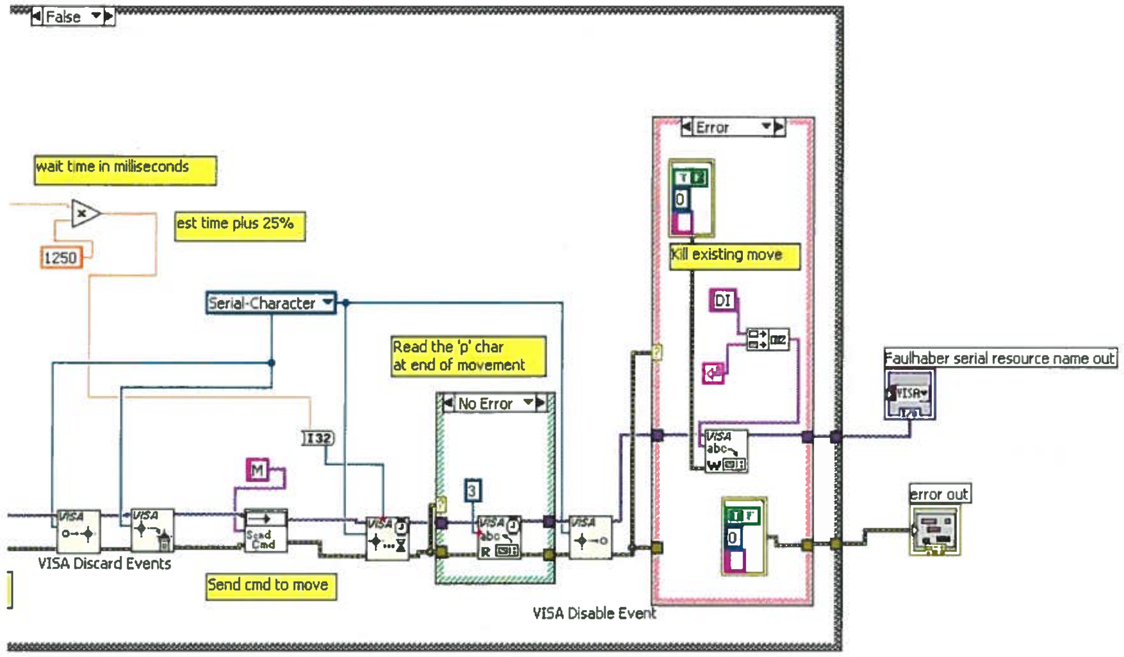


Figure 31 Main sub-functions – Move a Distance at Speed – Part 2

Before the command to start moving is sent, the software will estimate how long the motion should take (plus 25%) and define this as a timeout to wait for the 'at position' notification.

The motion is started and the timeout is set. If there has been any communication error then the code will simply pass this error through. Without any previous error, the code will perform a 'read with timeout' looking for the single character (plus carriage return and line feed) notification.

After the characters arrive, or after the timeout, the code will turn off the checking for timeout events. The function will simply complete if there was no timeout. A timeout, or previous, error will execute the case shown in the figure. First the code will temporarily force a 'no error' value so that it can issue a command to disable the motor. Once the motor has been disabled, the code will force a 'yes error' condition that will be passed back to the main program.

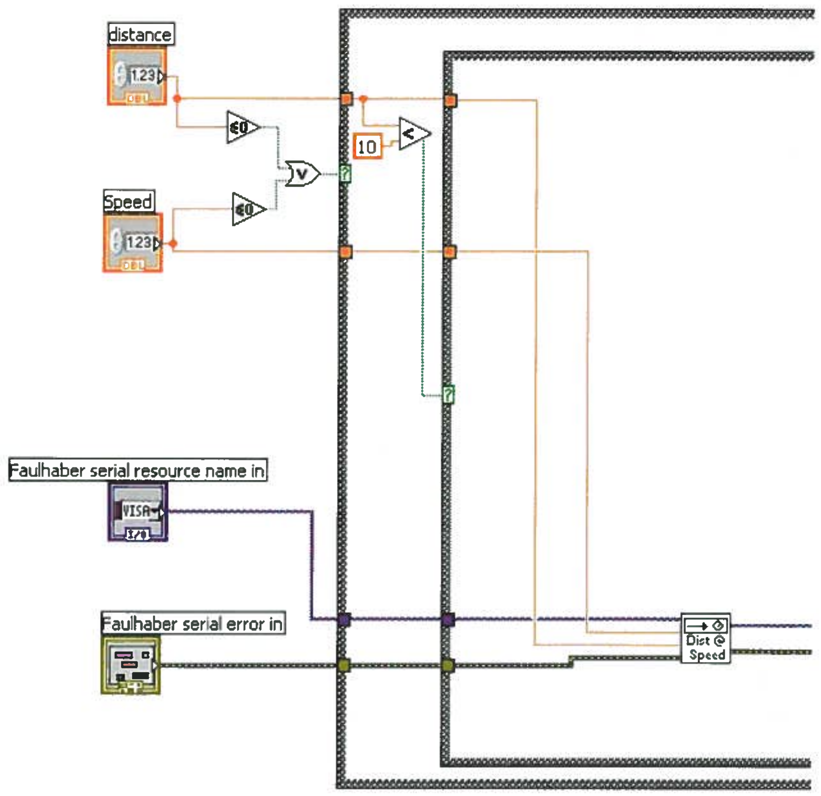


Figure 32 Main sub-functions – Fast Jab move – Part 1 – Less than 10 mm

A call to move with a 'jab' motion may not have any effect. Like the standard, constant motion function, a zero value or less for speed will cause to motor to do nothing. Different from the previous function is a check of zero or negative distance values. An extra check on the value of the distance will cause small movements to behave as if the 'jab' option had not been selected.

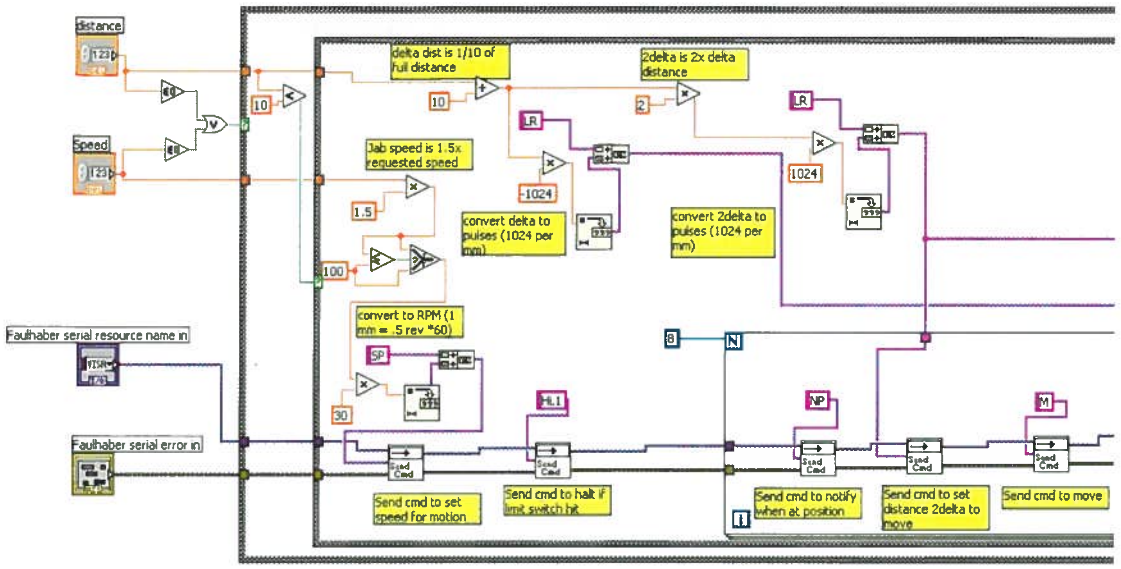

distance
Speed
Faulhaber serial resource name in
Faulhaber serial error in
delta dist is 1/10 of full distance
2delta is 2x delta distance
Jab speed is 1.5x requested speed
convert delta to pulses (1024 per mm)
convert 2delta to pulses (1024 per mm)
convert to RPM (1 mm = .5 rev *60)
Send cmd to set speed for motion
Send cmd to halt if limit switch hit
Send cmd to notify when at position
Send cmd to set distance 2delta to move
Send cmd to move
10
1.5
100
30
1024
-1024
8
LR
SP
HL1
NP
M

Figure 33 Main sub-functions – Fast Jab move– Part 1 – Greater than 10 mm

To implement a 'jab' insertion motion, the code must calculate two sub-movements. One is one tenth of the desired distance, i.e. the total desired motion will be implemented as roughly ten 'jab' motions. The second sub-movement is equal to twice this one tenth or 'delta' distance. A 'jab' will consist of a forward move of this 'twice delta' distance, followed by a withdrawal of a single 'delta' distance. Eight of these dual motions, followed by a final forward 'twice delta' movement, will complete the distance.

The code will also modify the speed of the motion. In general the 'jab' motions will be at 1.5 times the speed defined by the user in the array. There is a limit, based in the system's mechanical components. If the new speed exceeds the value it will be reduced to the limit to prevent damage to the hardware of the test bed.

Note: A version of this 'jab' sequence function was also implemented as a series of calls to the standard, constant motion, LabView function. Software overhead made this method too slow, with annoying delays between each of the sub-movements.

Motion speed is sent to the motor controller via a command. The motor is also told to halt movement if a limit switch is triggered.

The code will start the loop of eight small insertion/withdrawal motions. A request for notification at the end of each sub-move is made as a check of valid operation.

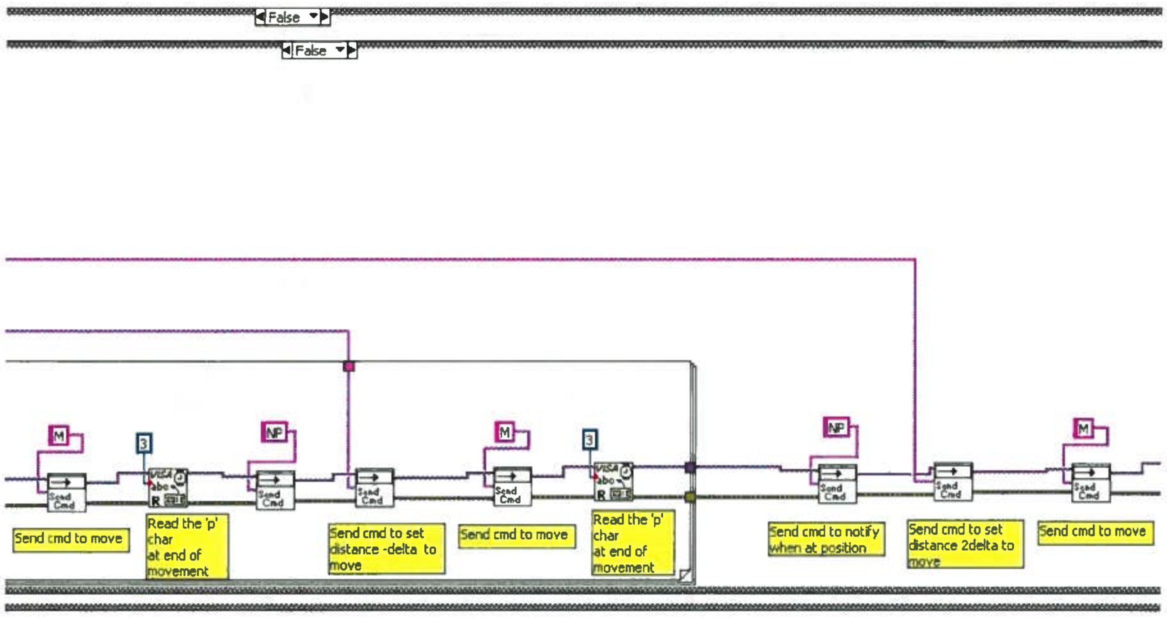

False
False
M
3
NP
M
3
NP
M
Send cmd to move
Read the 'p' char at end of movement
Send cmd to set distance -delta to move
Send cmd to move
Read the 'p' char at end of movement
Send cmd to notify when at position
Send cmd to set distance 2delta to move
Send cmd to move

Figure 34 Main sub-functions – Fast Jab move– Part 2

Figure 34 provides the remainder of the code for the 'eight times' loop. After moving 'twice delta' forward, the code will wait for the notification. A timeout error will propagate forward and cancel all other operations. If there is no timeout the code will issue the commands to withdraw 'one delta', with timeout checking. The entire 'for' loop will repeat until all 8 cycles are complete.

When the 'eight times' loop is done, there is still one more 'twice delta' forward move to complete the desired distance.

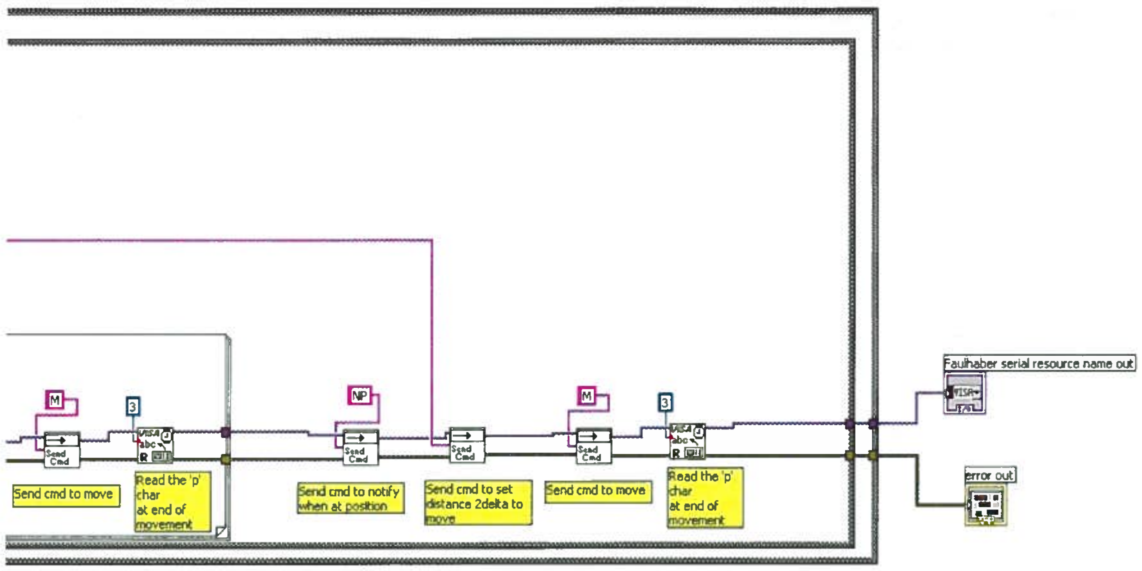
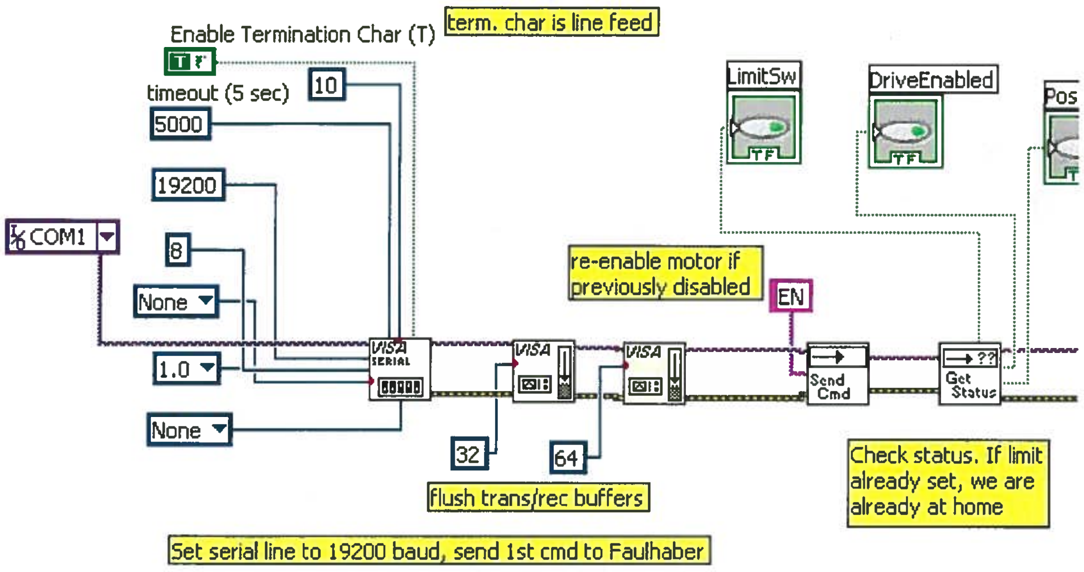

M
3
NP
M
3
Send cmd to move
Read the 'p' char at end of movement
Send cmd to notify when at position
Send cmd to set distance 2delta to move
Send cmd to move
Read the 'p' char at end of movement
Faulhaber serial resource name out
error out

Figure 35 Main sub-functions – Fast Jab move– Part 3

The last "twice delta" forward motion is executed to complete the fast jab function. Code will still check for timeout on the last forward sub-motion. As the function is now complete, any error conditions will be passed back up to the main program.



Enable Termination Char (T)
term. char is line feed
timeout (5 sec)
10
5000
19200
COM1
8
None
1.0
None
VISA SERIAL
VISA
VISA
32
64
flush trans/rec buffers
Set serial line to 19200 baud, send 1st cmd to Faulhaber
re-enable motor if previously disabled
EN
Send Cmd
Get Status
LimitSw
DriveEnabled
Pos
Check status. If limit already set, we are already at home

Figure 36 Motor functions – Initialize motor communication – Part 1

All needle motions are executed by the motor controller in the test bed. This electronics is connected to the computer and software via an RS-232 communication connection. Before any commands can be sent, this communication line must be set to the correct port and communication speed.

Communication buffers are flushed to remove any unexpected characters. Commands can now be sent to the motor controller. Since the motor may be disabled from a previous error, the motor is re-enabled. Then the current motor status is requested.

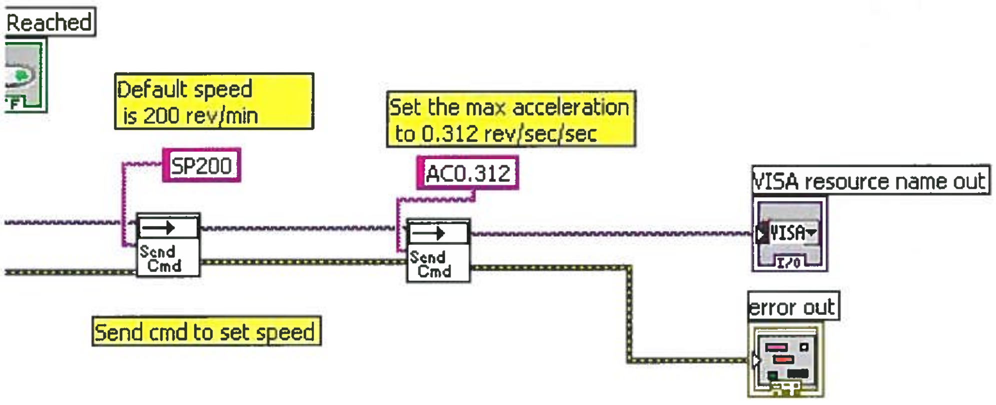


Figure 37 Motor functions – Initialize motor communication – Part 2

To complete the motor initialization, the code will define default values for motions speeds and acceleration.

Any error condition from any part of this function will be passed back to the calling program, so the error return can be used to indicate a communication failure.

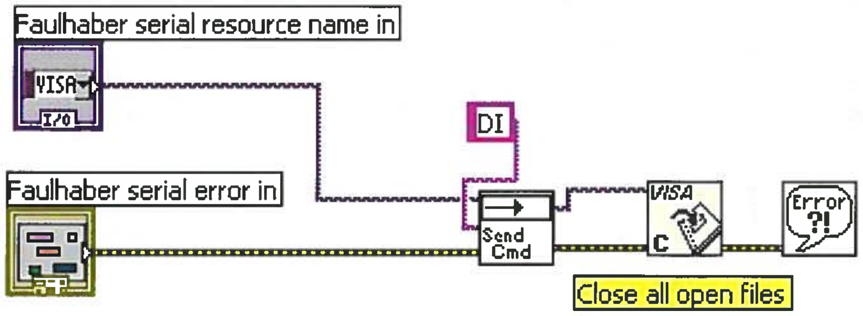


Figure 38 Motor functions – Stop motor

To stop the motor, the code will simply disable it, close down the communication channel, and display any new or existing error message.

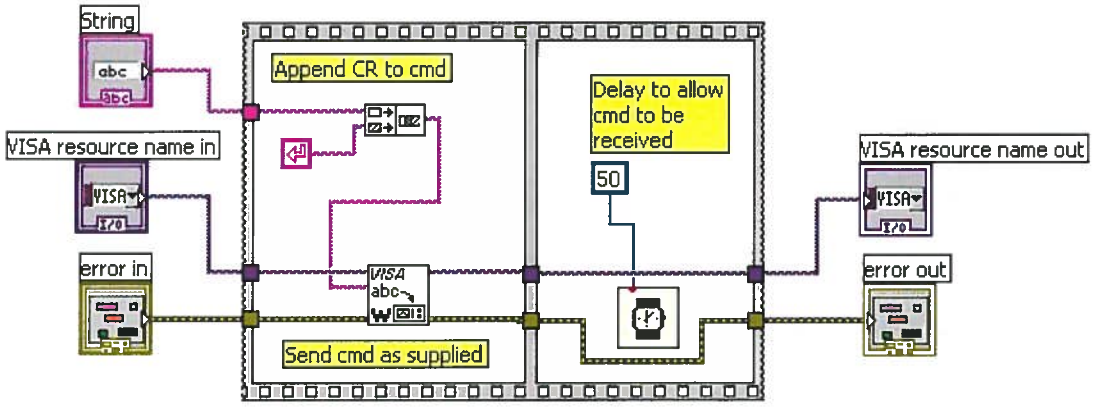


Figure 39 Motor functions – Send command

A key motor function is to send a command to the controller. When a command string is supplied it will have a carriage return appended and the modified string is communicated over the serial line connection to the controller. The function concludes with a short delay so that multiple commands sent one after the other will not overlap and confuse the controller's software.

The function is implemented as a sequence so that the delay will happen after the command is sent. Otherwise, the delay would start at the same time as the string was being appended.

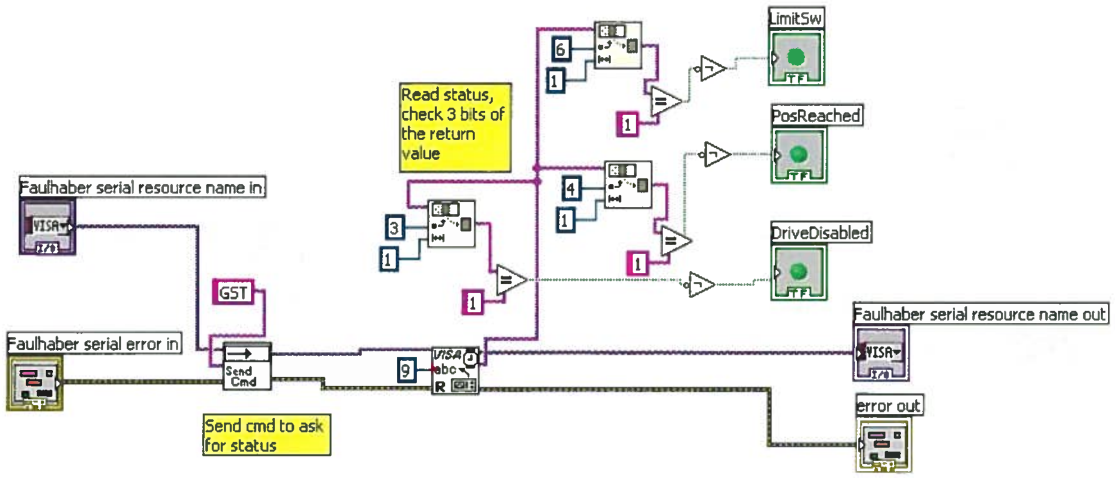

Faulhaber serial resource name in
Faulhaber serial error in
GST
Send Cmd
Send cmd to ask for status
Read status, check 3 bits of the return value
LimitSw
PosReached
DriveDisabled
Faulhaber serial resource name out
error out

Figure 40 Motor functions – Get Status

This motor function will send a command to ask for the basic motor status. The reply will be a string of eight (8) characters, followed by a carriage return. The reply string will consist of characters which will either be a "0" or a "1". Three of these characters (bit numbers 3, 4 and 6) are extracted and compared to a character "1" to obtain a Boolean value that is passed back to the calling program. See the command reference documentation supplied with the motor controller for the description of all the remaining status bits.

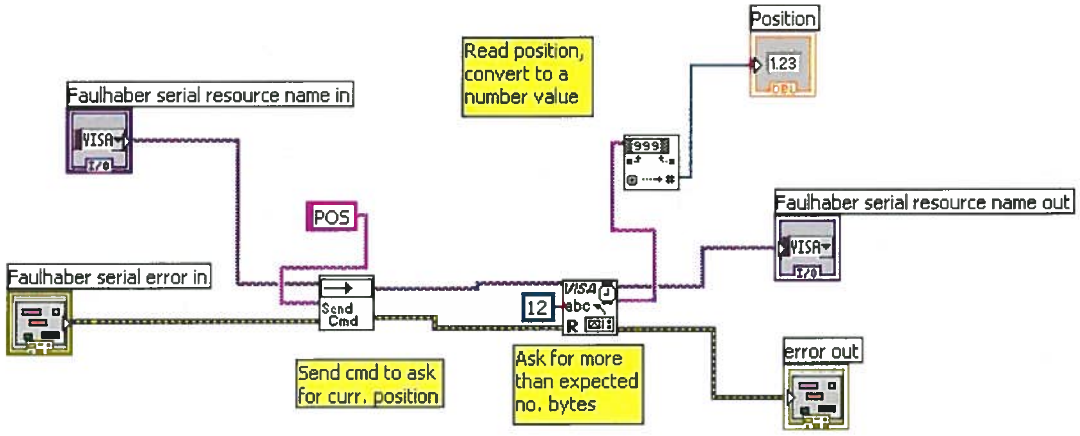

Position
1.23
Read position, convert to a number value
Faulhaber serial resource name in
VISA
999
POS
Faulhaber serial resource name out
Faulhaber serial error in
Send Cmd
12
VISA abc R
error out
Send cmd to ask for curr. position
Ask for more than expected no. bytes

Figure 41 Motor functions – Get position

Another useful motor function is to ask for the current position of the motor with respect to its current home location. The reply will be a string of number characters with sign, where the string length may be short or long depending upon the distance value. A LabView function is used to convert this character string into a number value that can be passed back to the calling program. Note that the number is always an integer number of encoder pulses away from home position.

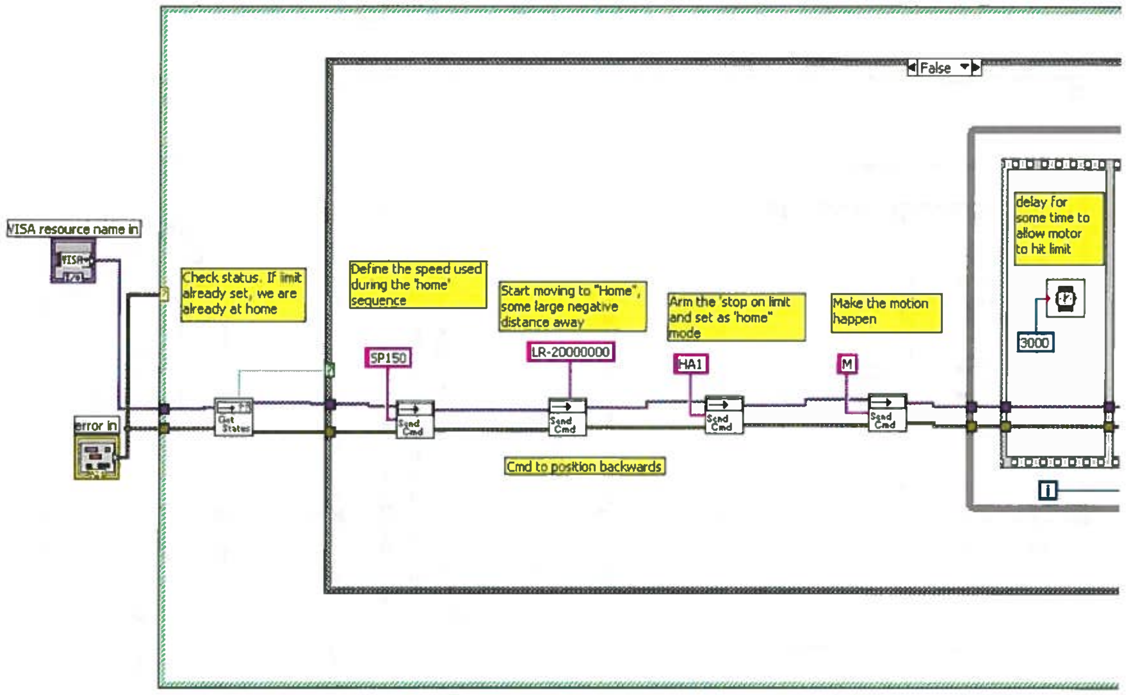

VISA resource name in
error in
Check status. If limit already set, we are already at home
Get Status
Define the speed used during the 'home' sequence
SP150
Send Cmd
Start moving to "Home", some large negative distance away
LR-20000000
Cmd to position backwards
Arm the 'stop on limit and set as 'home" mode
HA1
Make the motion happen
M
False
delay for some time to allow motor to hit limit
3000

Figure 42 Motor functions – Find home position – Part 1

Whenever the motor and needle insertion system is started, i.e. the application software is started, it is very important to insure that the motor is in a known, safe position. This function will perform the checking and the motor motions. If there is an existing motor error then the function will do nothing.

First the code will request the status of the motor, looking for the current value of the limit switch(s). If at least one limit switch is "On" then the code will execute the "True" option of the inside case or decision structure (not shown) and this can proceed to the logic to move off the switch again (see Figure 44). If no limit switch is currently active, i.e. the "False" case shown in the diagram, then the code will start a long movement in the direction towards the 'home' limit switch. Motions will be slow, and the controller will be asked to halt as soon as the home switch is activated.

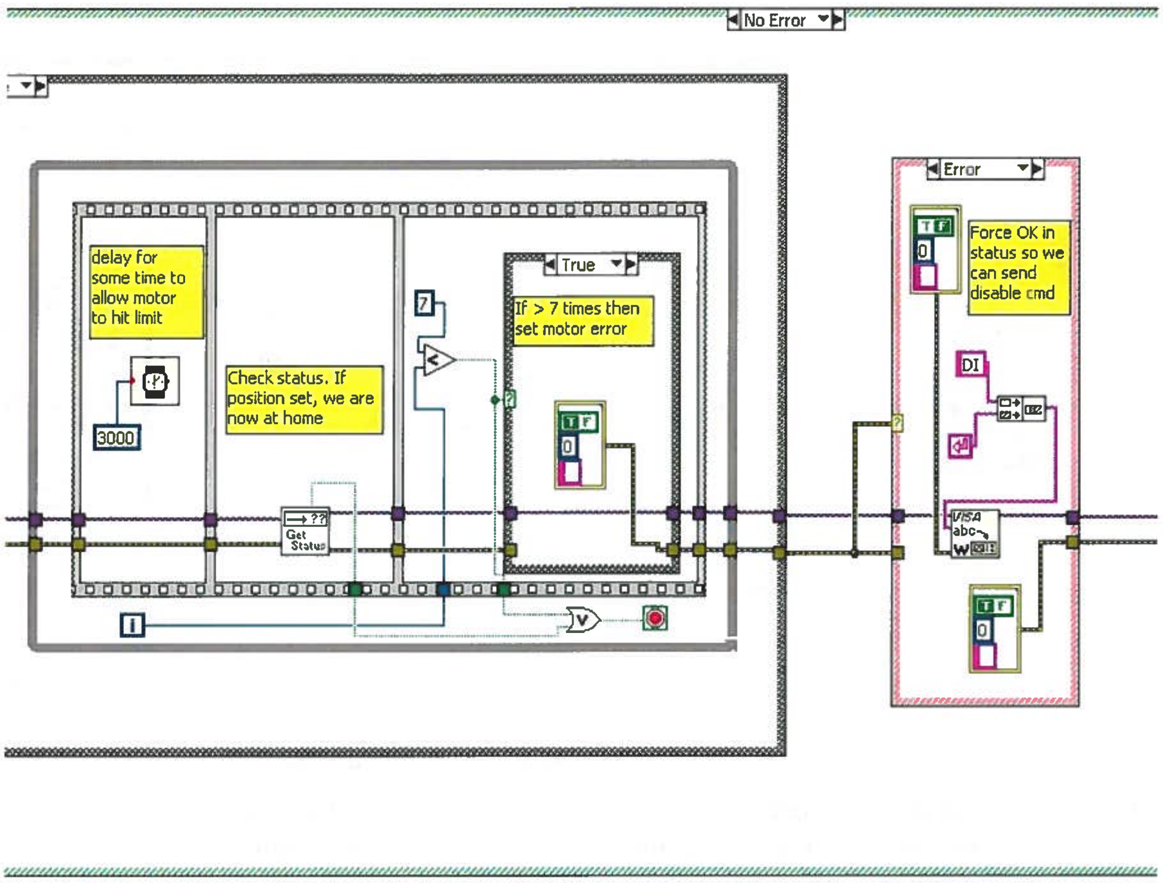

No Error
Error
delay for some time to allow motor to hit limit
3000
Check status. If position set, we are now at home
Get Status
7
True
If > 7 times then set motor error
Force OK in status so we can send disable cmd
DI
VISA abc W
i

Figure 43 Motor functions – Find home position – Part 2

The motor will have already started the search for home position in the code shown in part 1 of this function diagram. The motor will now be moving. In the remainder of the "False" case shown for the decision box the code will be waiting for the limit status signal to turn on. Since this may take some time, i.e. the motor may be very far from the home limit switch, the code will delay and loop several times waiting for this switch to be activated.

If a limit switch is not activated after the code waits for the several time delays, then an error status will be set for the motor. This will cause the error handling case box to restore a 'not error' motor status, issue a command to disable the motor, and force an error status once again.

When the status message indicates that the limit switch has been activated, the code will break out of the loop and simply bypass the error handling case box.

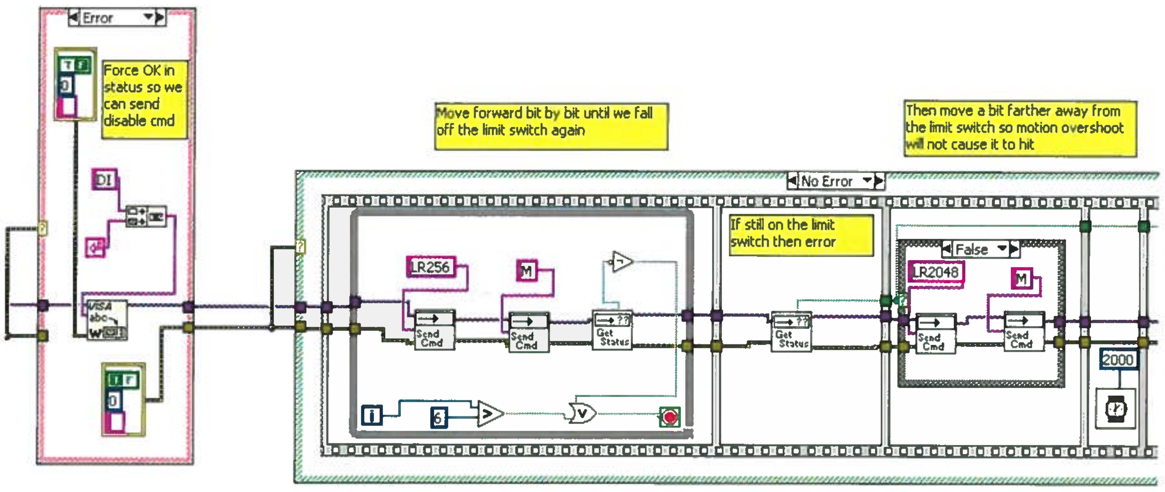
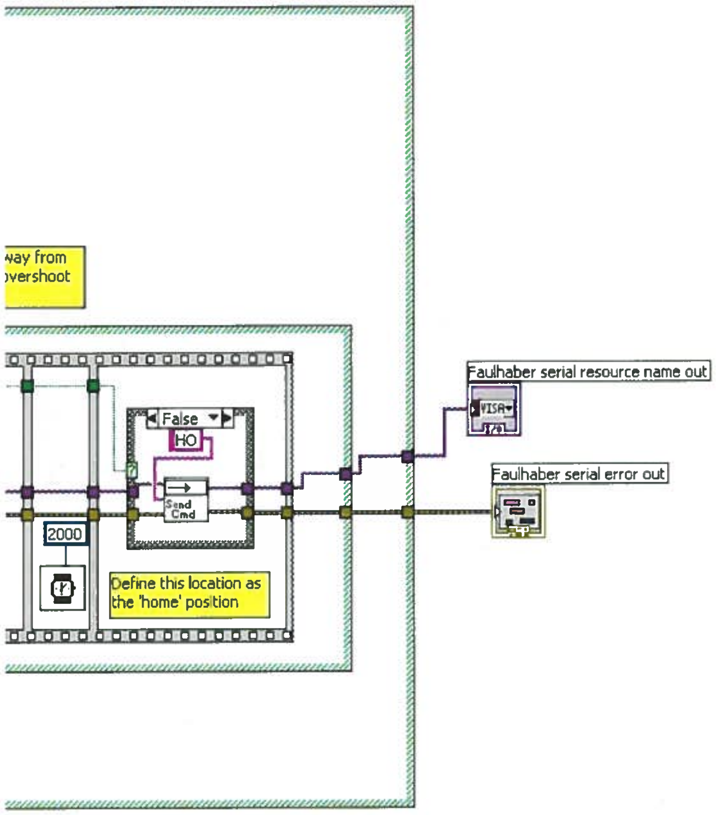

Error
Force OK in status so we can send disable cmd
DI
VISA abc W
Move forward bit by bit until we fall off the limit switch again
LR256
M
Send Cmd
Send Cmd
Get Status
6
No Error
If still on the limit switch then error
Get Status
Then move a bit farther away from the limit switch so motion overshoot will not cause it to hit
False
LR2048
M
Send Cmd
Send Cmd
2000

Figure 44 Motor functions – Find home position – Part 3

Now that the motor is on top of the 'home' limit switch, or if it was already on top of a limit switch when the function was started, it is necessary to move it off the switch. A loop of up to six (6) attempts will try small movements away from the home limit switch until the status indicates that the limit switch turns "Off". Once the status bit is off, it is necessary to move the system a bit farther away. Since the mechanical system may overshoot as it moves towards home position, it is necessary to have some distance so that the overshoot does not trigger this limit switch again.

If the code cannot move the motor off the limit switch, then perhaps the software was started on top of the limit switch close to the sample holder, not the close to the motor. All attempts to move the system off the switch would actually drive it further on to the wrong limit switch. In this case the loop count would be exceeded and the code will set a motor error status in the case statement instead of the extra "move farther away" command.



way from
overshoot
False
HO
Send Cmd
2000
Define this location as the 'home' position
Faulhaber serial resource name out
VISA
Faulhaber serial error out

Figure 45 Motor functions – Find home position – Part 4

When all the function movements are completed, and if there is no error status, then the code will issue the command to set the current position as the home position. If there was a motor error, the case option box will select an option which does not issue the "Home" command.

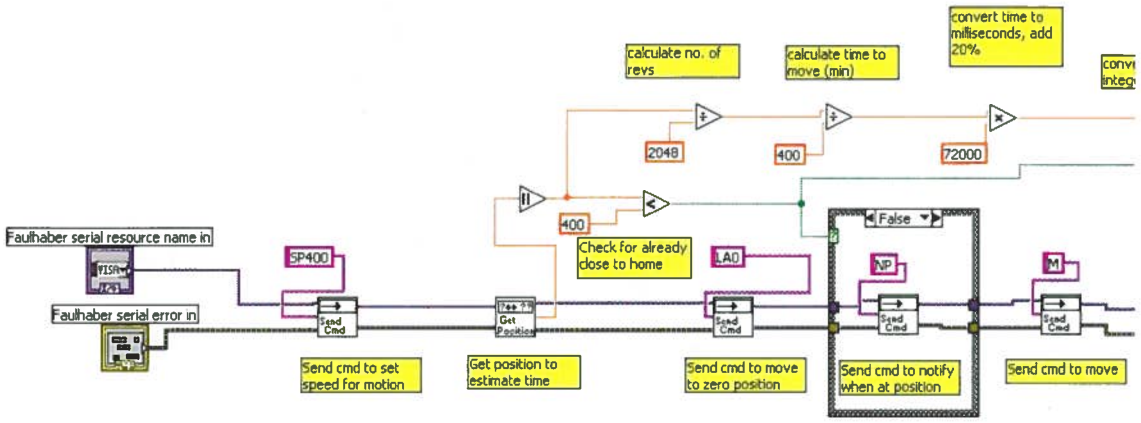

calculate no. of revs
calculate time to move (min)
convert time to milliseconds, add 20%
conv integ
2048
400
72000
400
Check for already close to home
False
Faulhaber serial resource name in
Faulhaber serial error in
SP400
LA0
NP
M
Send cmd to set speed for motion
Get position to estimate time
Send cmd to move to zero position
Send cmd to notify when at position
Send cmd to move

Figure 46 Motor functions – Go to home position – Part 1

After an insertion motion, or after a manual motor movement, it is necessary to issue a command to reposition the motor back to the home location. This motion is done at a medium speed since the motor is assumed to be in a safe condition when this function is called.

For very short moves, i.e. when the system is already very close to the 'zero' position, then the code will not check for the notification character. In this case the 'in position' character would arrive so quickly that the code would miss it. For longer moves back to home, the travel time is estimated, a percentage is added, and the motion is set to notify when completed.

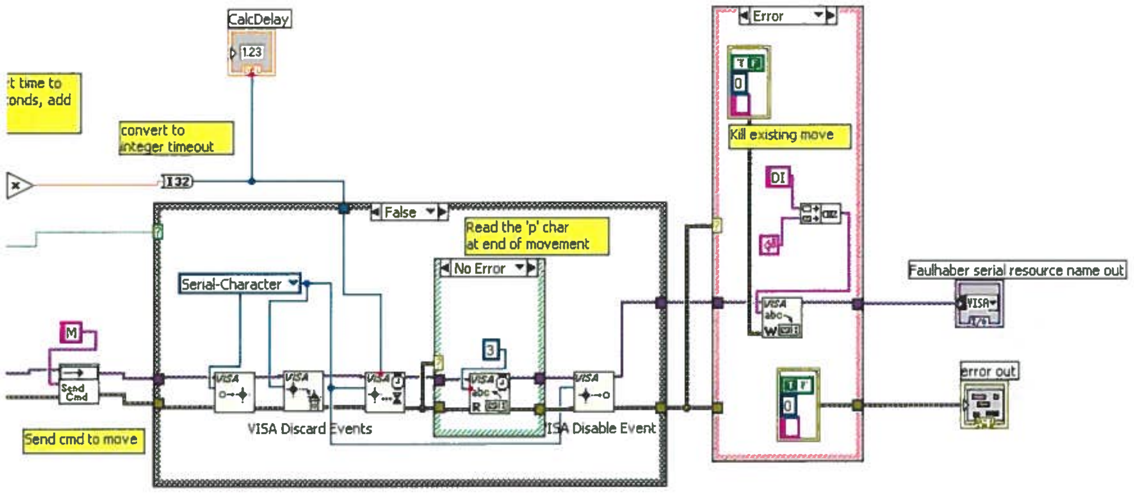


Figure 47 Motor functions – Go to home position – Part 2

For long movements back to home position, the code will wait for the notification character. The estimated travel time is used for a timeout value. If the timeout expires, the code will force the motor to be disabled, and set an error motor status. If the code does not timeout, i.e. because the notification character arrives, or if the movement was short, the code will return and allow the motor controller to complete its final positioning of the motor at the zero location.