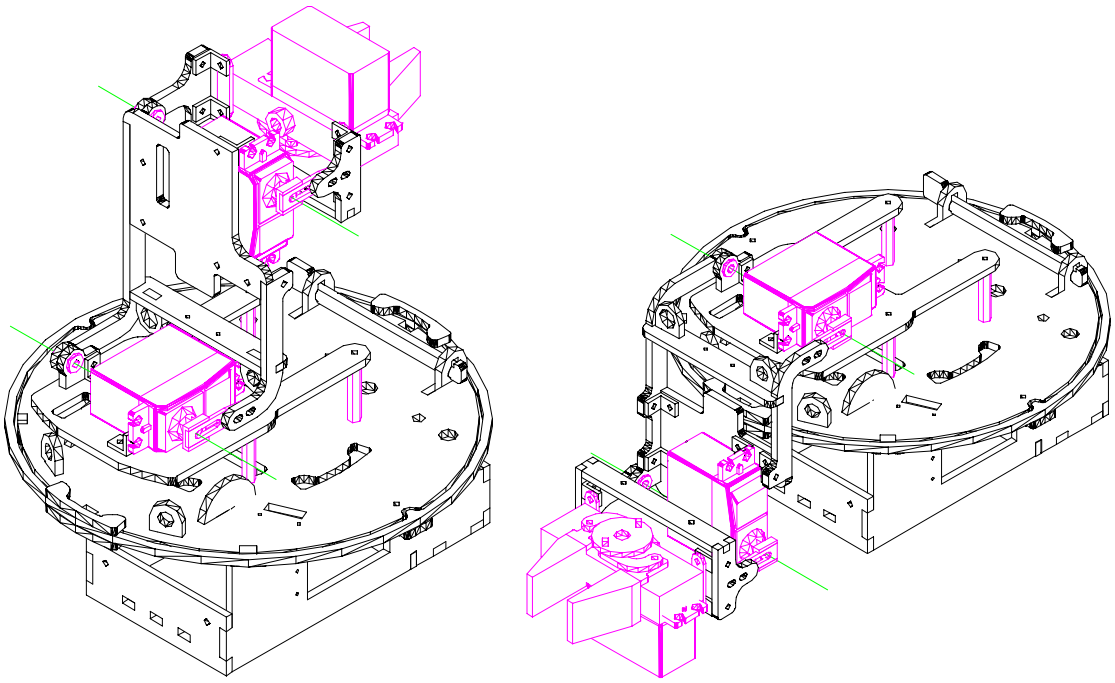
# MEKARM™ PLANAR MANIPULATOR USERS MANUAL

by
Keith L. Doty


Version 01



316 NW 17 Street Suite A Gainesville Florida 32603 *www.mekatronix.com*

# AGREEMENT

This is a legal agreement between you, the end user, and Mekatronix™. If you do not agree to the terms of this Agreement, please promptly return the purchased product for a full refund.

1. **Copyright Notice.** Mekatronix™ hereby grants to any individuals or organizations permission to reproduce and distribute copies of this document, in whole or in part, for any personal or non-commercial educational use only. This copyright notice must accompany any copy that is distributed.

2. **Copy Restrictions.** Other than cases mentioned in **Copyright Notice**, no part of any Mekatronix™ document may be reproduced in any form without written permission of Mekatronix™. For example, Mekatronix™ does not grant the right to make derivative works based on these documents without written consent.

3. **Software License.** Mekatronix™ software is licensed and not sold. Software documentation is licensed to you by Mekatronix™, the licensor and a corporation under the laws of Florida. Mekatronix™ does not assume and shall have no obligation or liability to you under this license agreement. You own the diskettes on which the software is recorded but Mekatronix™ retains title to its own software. The user may not rent, lease, loan, sell, distribute Mekatronix™ software, or create derivative works for rent, lease, loan, sell, or distribution without a contractual agreement with Mekatronix™.

4. **Limited Warranty**. Mekatronix™ strives to make high quality products that function as described. However, Mekatronix™ does not warrant, explicitly or implied, nor assume liability for, any use or applications of its products. In particular, Mekatronix™ products are not qualified to assume critical roles where human or animal life may be involved. For unassembled kits, the user accepts all responsibility for the proper functioning of the kit. Mekatronix™ is not liable for, or anything resulting from, improper assembly of its products, acts of God, abuse, misuses, improper or abnormal usage, faulty installation, improper maintenance, lightning or other incidence of excess voltage, or exposure to the elements. Mekatronix™ is not responsible, or liable for, indirect, special, or consequential damages arising out of, or in connection with, the use or performances of its product or other damages with respect to loss of property, loss of revenues or profit or costs of removal, installation or re-installations. You agree and certify that you accept all liability and responsibility that the products, both hardware and software and any other technical information you obtain has been obtained legally according to the laws of Florida, the United States and your country. Your acceptance of the products purchased from Mekatronix™ will be construed as agreeing to these terms.

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

# MANIFESTO

Mekatronix™ espouses the view that personal autonomous physical agents will usher in a whole new industry, much like the personal computer industry before it, if modeled on the same beginning principles:

- Low cost,
- Wide availability,
- Open architecture,
- An open, enthusiastic, dynamic community of users sharing information.

Our corporate goal is to help create this new, exciting industry!

**WEB SITE:** http://www.mekatronix.com
**Address technical questions to** tech@mekatronix.com
**Address purchases and ordering information to an authorized Mekatronix Distributor** http://www.mekatronix.com/distributors

## Disclaimer

While Mekatronix™ has placed considerable effort into making these instructions accurate, Mekatronix™ does not warrant the results and the user assumes the risks to equipment and person that are involved.

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

# TABLE of CONTENTS

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

# LIST OF FIGURES

# LIST OF TABLES

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

## 1. SCOPE

This manual assumes you have an assembled MekArm™ manipulator and have connected it properly to an MTJPRO11 microcontroller. A MekArm™ mounted on the top plate of a TJ Pro™ robot, as illustrated on the cover page, represents a typical configuration and will be assumed for purposes of exposition in this manual.

## 2. SYSTEM PREREQUISITES

If you already have a TJ Pro™ Robot, you will only need to mount the MekArm™ and install the MekArm™ Distribution Software (Part# mekarmdist01.zip) to get the arm up and running. The MekArm™ can be mounted on either the TJ Pro robot™ or the MindStamp™ platform. Refer to the *MekArm Assembly Manual* for instructions to mount the MekArm™ on a TJ Pro™ robot. If you have either of these systems, you will most likely have items 3, 4, 5, 6, 7, 8 and 9 listed below. For embedded designs, you will need to purchase an MTJPRO11 microcontroller and the other items.

*Note: For WINDOWS 2000 or NT Users, Mekatronix's High-Speed Serial Downloader 11 (Part# HSSDL11) is a must, due to serial port control and buffering performed by those systems!*

In general, to install and operate a MekArm™ you will need

1. An assembled and tested MTJPRO11 microcontroller (either Part# MTJPRO11A-BOT or MTJPRO11A-TOP) that is operational.
   *To be operational the MTJPRO11 must possess properly wired power and download switches, visible LEDS, and a charge jack and charge circuit (Refer to a Mekatronix distributor for parts and prices and the TJ Pro Assembly Manual for circuit details).*

   *Mekatronix's TJ Pro robot and MindStamp microcontroller development system meet all hardware operational requirements.*

2. MekArm™ distribution software package, (Part# mekarmdist01.zip),
   *This software is usually bundled with the arm.*
3. C-compiler (Part# ICC11-w5)

*Note: The MekArm™ cannot be operated under Interactive C with the MTJPRO11 microcontroller because of a conflict with the usage of pin PA4.*

4. Mekatronix MB2325 Communications board (Part# MB2325),

5. Nine Pin RS232 Serial Modem Cable: 6 feet (Part# DB9RS232MC ),
6. 6-Wire Serial Ribbon Cable: 6 ft long with 1x6 female connectors on each end (Part# C2325a),
7. AC adapter for internal robot battery charger or optional external battery chargers. USA standard (110vac to 12v DC, 500ma) (Part# BC12A500)
8. Package of 6 AA, NiCd, 800mah rechargeable batteries (Part# AANCMEKA800)
9. External battery charger. Includes: 6 AA battery holder, resistors, wire, jack (does NOT include the BC12A500 which is required.) (Part# EXTCHGR6a)
10. High-Speed Serial Downloader 11 (Part# HSSDL11) for WINDOWS 2000 and NT users.
11. A personal computer running WINDOWS 95 or better…you can run MekArm™ on old DOS machines, but Mekatronix does not support that mode of operation.

## 3. FOR THOSE WHO DO NOT READ MANUALS

The old Engineering adage, "When all else fails, read the instruction manual" entails more truth than the obvious. Manuals must be complete enough to permit individuals with various levels of competency and knowledge to utilize the information. Those with high levels of competency and experience will find much of the material in any manual redundant to their needs, hence, the penchant not to read them. Others might not find enough information, hence, their frustration. This section is extremely terse and written for the former and those too impatient to read the whole manual. Refer to later sections for needed details.

*Note: WINDOWS 2000 and NT can cause downloading problems. See Section 3.5 .*

***READ SECTION 4 ON SAFETY BEFORE CONTINUING!!!***

### *3.1 Quick Start: Install the MekArm™ Hardware*

1. Charge your 6-AA NiCd batteries overnight.
2. Mount and connect the MekArm™ to your microcontroller system as instructed in the *MekArm™ Assembly Manual*. Refer to Sections 6, 7, and 8.
3. Connect your personal computer with the microcontroller system using the serial cables and MB2325 Communications board. Refer to Section 9.

### *3.2 Quick Start: Install the MekArm™ Software*

1. Install your C-compiler (Part# ICC11-w5) in your preferred working directory in a WINDOWS 95 or better OS. For example, install your compiler in the directory `c:\icctjp`, if you have a TJ Pro™ robot, or `c:\iccms`, if you have a MindStamp. Refer to Sections 10 and 11.
2. Unzip the MekArm™ Distribution Software package. Refer to Sections 12 and 13.
3. Move the *subdirectories* of the distribution package to your preferred working directory. From here on, I will assume your working directory is `c:\icctjp`. Replace that directory name with your actual working directory. Refer to Sections 12 and 13.

4. If you have Mekatronix's Image Craft C (ICC11), setup your IDE (Integrated Development Environment) Compiler Options as follows. Refer to Section 11.
   a) *Preprocessor*: `c:\icctjp\include c:\icctjp\InclArm`
   b) *Linker*: Same as TJ Pro™ except for libraries.
      Additional Libraries: `libtjp8 libarm`
      *Note: At least one space between the names in a) and b)*
5. Transfer archive libraries `libtjp8.a` and `libarm.a` in `LibArm` to `c:\icctjp\Lib`. Refer to Section 11 .
6. Remove only the Mekatronix-specific header files from `c:\icctjp\include`. Updated version of these header files are in `c:\icctjp\InclArm`. The *Preprocessor* setup in Step 4 will insure that the compiler can find the new header files. Refer to Section 11.

### *3.3 Quick Start: MekArm™ Program Development, Downloading, Debugging, and Execution*

After performing the setup in Sections 3.1 and 3.2 , you can develop, download, debug, and execute MekArm™ programs in the same manner as programming any Mekatronix robot or MindStamp™. Refer to Sections 14, 15, and 16.

*CAUTION! CAUTION! CAUTION!*

*Exercise caution with the MekArm™. Even a small moving arm can cause damage to objects and people if misused or mishandled. Especially be careful during program debugging.*

### *3.4 Quick Start: MekArm™ Operation*

To learn about the MekArm™ library functions, data structures and example programs so you can figure out how to use them in your application programs refer to Sections 17, 18, 19, and 20.

### *3.5 Problems with WINDOWS 2000 Professional and NT*

ICC11 downloader does not work with WINDOWS 2000 or NT. The Mekatronix High-Speed Serial Downloader 11 (Part# HSSDL11) must be used. Even the HSSDL11 will almost always report a transmission error at the end of the second phase of transmission, but this error report is invariably false due to buffering and hardware control anomalies with WINDOWS 2000. Only errors in the first phase of downloading by the HSSDL11 must be taken seriously.

## 4. SAFE USE OF THE MEKARM™

Although small, the MekArm™ is powerful and must be treated with respect.

**WARNING! WARNING! WARNING !**

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

*Severe pinching or other injury may result through careless operation of the MekArm™. To avoid possible severe injury, keep people, faces, fingers and objects clear of the moving arm, especially during program debugging and at program startup.*

Startup of MekArm is most critical. MekArm™ programs assume the arm's initial position is the HOME CONFIGURATION or pose (**Figure 1**). If the arm is not in the home position and the program starts, the arm will whip violently to the home position from its current configuration. This motion can cause severe eye injury if your face is too close to the robot or the robot is holding an object and flings it.

## *4.1 MekArm Startup Policy*

Write all MekArm™ programs to assume MekArm™ is in the HOME CONFIGURATION at startup. This systematic procedure is necessary because the MekArm™ operates open-loop, that is, it has no joint sensor feedback telling the robot microcontroller the arms' current joint angles. With adoption of this procedure, you will always know to manually rotate MekArm™ into its HOME CONFIGURATION or HOME POSE, as shown in **Figure 1**, before downloading any programs and executing them.

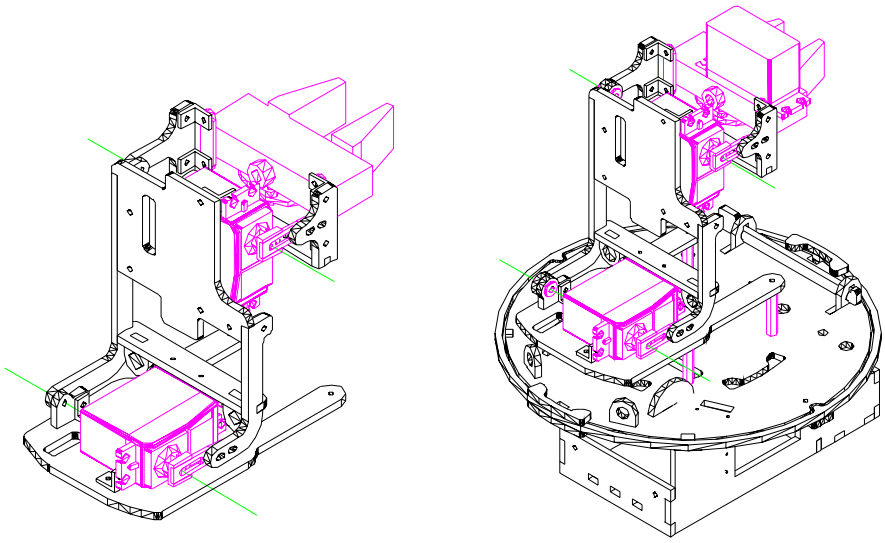


**Figure 1. In the HOME CONFIGURATION of the MekArm™, both the shoulder and the wrist servos are rotated almost fully clockwise, resulting in the fingers pointing to the rear of the robot. The finger gap is not specified in the home position.**

### *4.2 Emergency Stop Procedure: MekArm™ Out of Control*

During the development of MekArm™ programs, errors might cause the arm to move wildly or unsafely. Execute the emergency stop procedure, to wit, turn the robot power switch off. DO NOT press the reset button!

# EMERGENY STOP PROCEDURE

*TURN ROBOT POWER OFF!*

*DO NOT PRESS the RESET BUTTON!!!*

*In an emergency, turn power off to the robot. The power switch is the toggle switch on the right-side of the robot. Keep hands clear of MekArm™ when turning the power off. This action will release the servos and they can be easily rotated by hand back to the HOME CONFIGURATION. Do not press the Reset button before moving MekArm™ to the HOME CONFIGURATION as this will cause the robot to fly wildly back to the HOME CONFIGURATION.*

With the power off and MekArm™ disabled, rotate the arm to its HOME CONFIGURATION. Make appropriate corrections to your program before turning power back on and downloading your program again.

## 5. DESCRIPTION OF THE MEKARM™ MANIPULATOR

The *MekArm*™ manipulator (**Figure 2**) constitutes a two degree-of-freedom (2-DOF) planar robotic arm with an actuated, parallel-finger gripper, all driven by servomotors under control of a robot. The three, independent, servo-driven motions, starting at the base, are called the *shoulder* joint, *wrist* joint and the *fingers/gripper*. The green lines projecting out of the shoulder and wrist servos designate the axis of rotation for the joints.

MekArm™ mounts on the TJ Pro™ robot to provide a 4-DOF manipulator with gripper, a total of five degrees of motion. The wheels of the robot provide two degrees of freedom in the plane of motion of the robot. The plane of motion of the forearm and end-effector is perpendicular to the floor and the gripper fingers move parallel with respect to each other.

The parallel-fingered gripper, or *hand*, provides the TJ Pro™ with the ability to manipulate small objects and represents a major advance in the robot's capabilities.

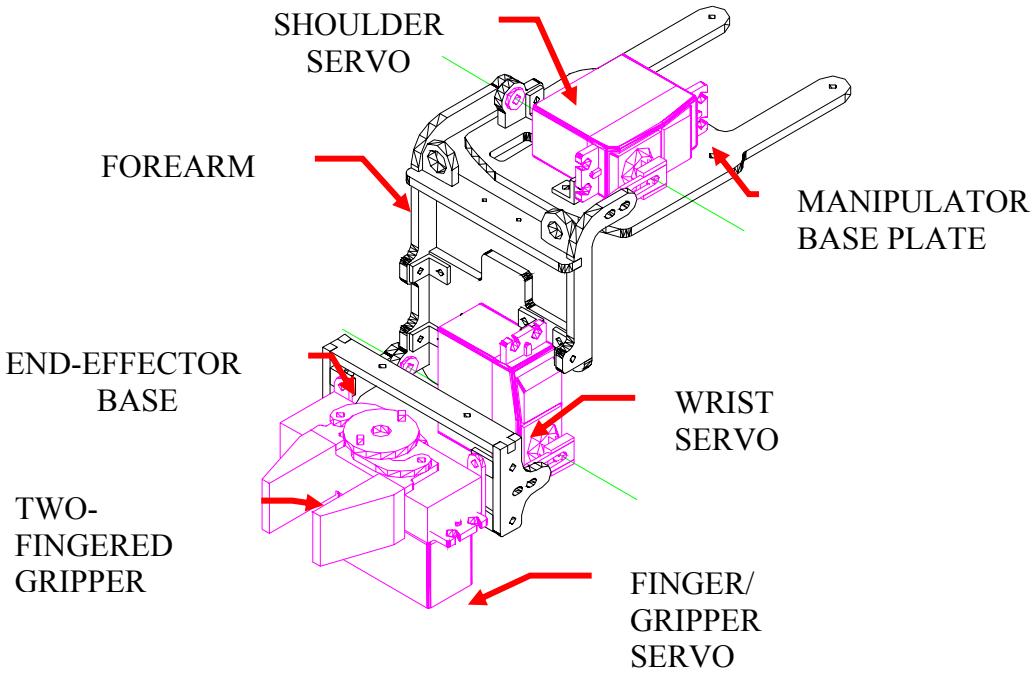



**Figure 2. The MekArm™ consists of three principle subassemblies, 1) the manipulator base, 2) the forearm, and 3) the end-effector plate with gripper. The manipulator possesses two rotating, or revolute joints, the shoulder and wrist servos. The gripper servo opens and closes the two fingers in parallel.**

## *5.1 MekArm™ Coordinate Frames*

In order to describe the motion of the MekArm™ mathematically, coordinate frames must be assigned to the various moving bodies of the arm. The standard Mekatronix frame definitions for the MekArm™ are illustrated in Figure 3 and Figure 4. Frame position and orientation, which together define the frame *configuration* or *pose*, with respect to the previous frame in the sequence $F_r$, $F_s$, $F_w$, $F_f$, and $F_t$, is specified in Table 1.

The five coordinate frames in the HOME CONFIGURATION of the MekArm™(Figure 3) will be specified as the 1) Robot body frame $F_r = \{ x_r, y_r, z_r \}$, 2) Shoulder frame $F_s = \{ x_s, y_s, z_s \}$, 3) Wrist frame $F_w = \{ x_w, y_w, z_w \}$, 4) Gripper frame $F_g = \{ x_g, y_g, z_g \}$, and 5) Fingertip frame $F_t = \{ x_t, y_t, z_t \}$. On the first three frames in this configuration, the x-axes point forward and, on the last two, the x-axes point towards the robots rear. The z-axis (bold green line) of the robot frame is vertical while the z-axes (bold red lines) of the shoulder and wrist frames are along the axes of rotations, indicated by the thin green lines. The z-axes of the gripper (bold red line) and fingertip frames (bold green line) point downward in this configuration.

The frame origins may be located as follows (Figure 4). The green circle in the middle of the robot top plate specifies the origin of the robot coordinate frame $F_r$. The origin of the shoulder frame $F_s$, designated by the lower red circle, is located at the midpoint of the shoulder turning axis bounded by the outer sides of the left and right forearm links. The origin of the wrist frame $F_w$, designated by the middle red dot, is located at the midpoint of the wrist turning axis bounded by the outer sides of the left and right forearm links. The origin of the gripper frame $F_g$, designated by the upper red dot, is located at the intersection of the turning axis of the gripper servo and an extension of the negative x-axis of the wrist frame $F_w$. The origin of the fingertip frame $F_t$, designated by the upper green dot, is located at the intersection of the finger x-axis and the plane perpendicular to the fingers at the fingertips.

Mathematical computation of frame transformations is beyond the scope of this manual. To learn more about MekArm™ mathematics and applications refer to the companion *MekArm™ Education Manual*. A software package that comes with the manual provides source code for *Coordinated* and *Parallel* motion control, particularly important applications. For details and pricing ask a Mekatronix distributor.

Several points of interest concerning the coordinate frames should be noted. First, the z-axes of all the frames attached to the servos fall along the axis of rotation of that servo. Counterclockwise rotation of the servo constitutes the positive sense of rotation and each servo angle ranges from 0 to 180 degrees.

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

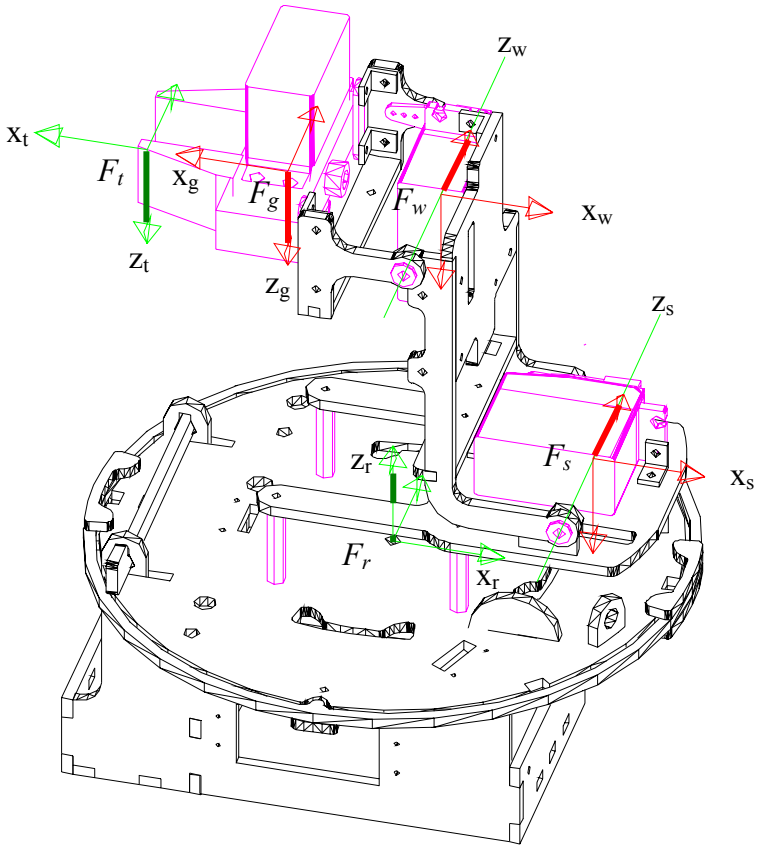



**Figure 3. This sketch illustrates the position of the five coordinate frames associated with the MekArm™ when in the HOME CONFIGURATION.**

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

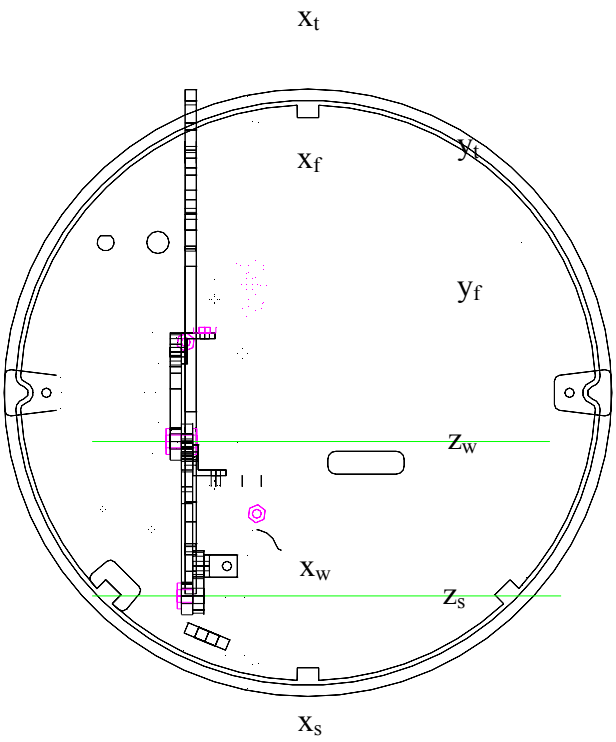



**Figure 4. The following description applies to the locations of the origins of the MekArm™ coordinate frames when the arm is in the HOME CONFIGURATION.**

**Table 1. Parameters relating MekArm™ coordinate frames.**

| | Frame Parameters | | | | | |
|---|---|---|---|---|---|---|
| **Frame Transform** | **Distance (inches)** | **ΔX (inches)** | **ΔY (inches)** | **ΔZ (inches)** | **X-Axis Rotation** | **Z-Axis Rotation** |
| Robot→Shoulder $F_r \rightarrow F_s$ | 2.730 (69.3mm) | 2.275 (57.8mm) | 0 | 1.508 | -90° | 0° |
| Shoulder→Wrist $F_s \rightarrow F_w$ | 3.663 (93.0mm) | -1.730 (-43.9mm) | -3.228 (-82mm) | 0 | 0° | 0° |
| Wrist→Gripper $F_w \rightarrow F_g$ | 1.744 (44.3mm) | -1.744 (-44.3mm) | 0 | 0 | -90° | 180° |
| Gripper→Finger-tip $F_g \rightarrow F_t$ | 1.594 (40.5mm) | 1.594 (40.5mm) | 0 | 0 | 0° | 0° |

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

## 6. MOUNTING MEKARM™ ON THE TJ PRO™ ROBOT

Collect together the parts listed in Table 2 to attach the MekArm™ to the TJ Pro™ robot. Refer to **Figure 5** during assembly.

**Table 2. Hardware for Mounting the MekArm™ onto the TJ Pro™ Robot**

| Qty | Part# | Description |
|---|---|---|
| 4 | SAAS75 | ¾ inch Anodized Aluminum Standoff |
| 4 | SAA25 | ¼ inch Anodized Aluminum Standoff |
| 4 | MSS50-4-40 | ½ inch 4-40 Slotted head, Stainless steal Machine Screw |

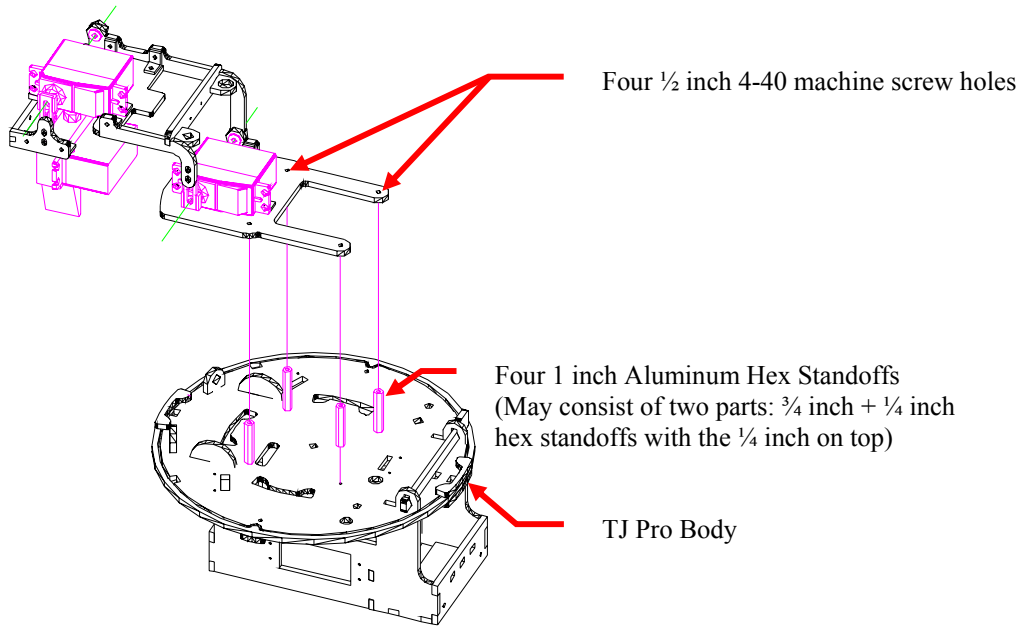



**Figure 5. This figure illustrates how to mount the MekArm on the TJ Pro™ robot.**

1. Screw the ¾ inch (1 inch if available) aluminum hex standoffs onto the threads of the printed circuit board mounting screws projecting through the top of the TJ Pro™. There should be enough thread to support the MekArm. If not, tighten the printed-circuit board mounting screws a bit tighter or replace them with 5/8 inch or ¾ inch 4-40 machine screws (not provided).

2. Insert the four 4-40 machine screws through the four holes provided on the top of the MekArm Manipulator Base.

3. If standoffs are not 1 inch, then screw the ¼ inch aluminum hex standoffs onto the ½ inch screws projecting out on the underneath side of the Manipulator Base.

4. Hold the Manipulator Base horizontally and steady and align the four screw protruding through the ¼ inch standoffs (Manipulator Base) with the holes on the four ¾ inch (1 inch) standoffs already mounted on the TJ Pro™ top plate.

5. Gradually tighten the screws going to opposite corners and just a few turns at a time until tightened fully, all the while keeping the manipulator base level.

## 7. MOUNTING MEKARM™ ON THE MINDSTAMP™

For the MindStamp™ microcontroller development system, the MTJPRO11 microcontroller sits on top of ¼ inch standoffs fastened to an 6.5inch by 8inch plastic sheet mounted on ¾ inch standoffs. MekArm™ can be mounted as shown in (Figure 6). The procedure is virtually identical to the procedure for mounting on the TJ Pro™ robot as described in Section 6.

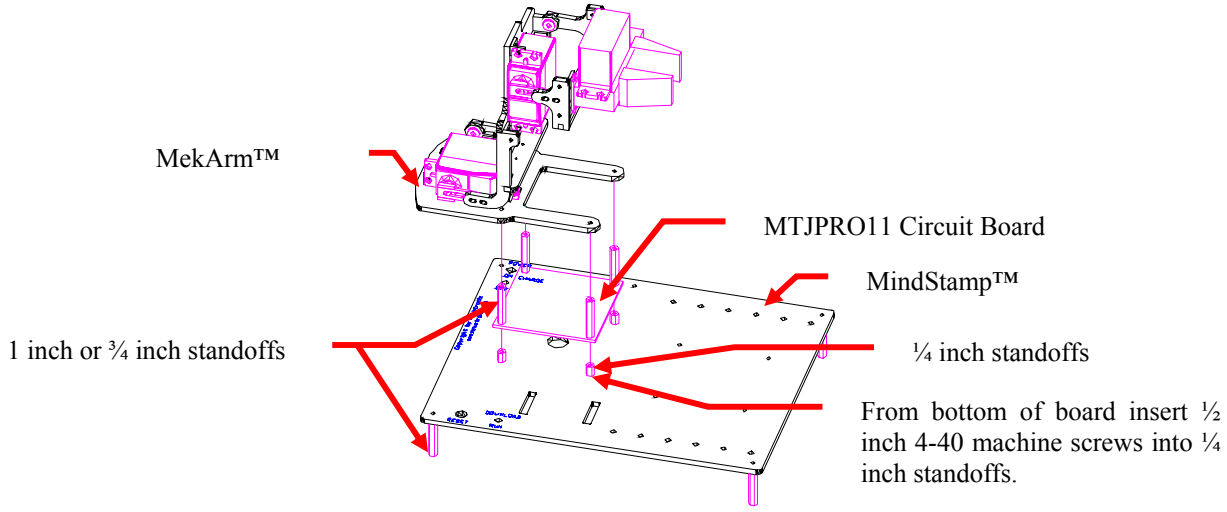



**Figure 6. Place MekArm™ on 1 inch or ¾ inch standoffs attached to the MTJPRO11 microcontroller circuit board. MindStamp™ will have to be elevated and counter balanced or fastened to a box that stands about 3.2 inches above the table top in order for the configuration to be stable and to insure that the arm has room to work.**

## 8. CABLING OF MEKARM™

Thread the servo cables of MekArm™ through the various wire passes as shown in Figure 7. For details on connecting the shoulder servo cable to the 3-pin male header PA4, the wrist servo cable to 3-pin male header PA5 and the gripper servo cable to 3-pin male header PA6 on the MTJPRO11 microcontroller, refer to the *MekArm™ Assembly Manual*.

316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

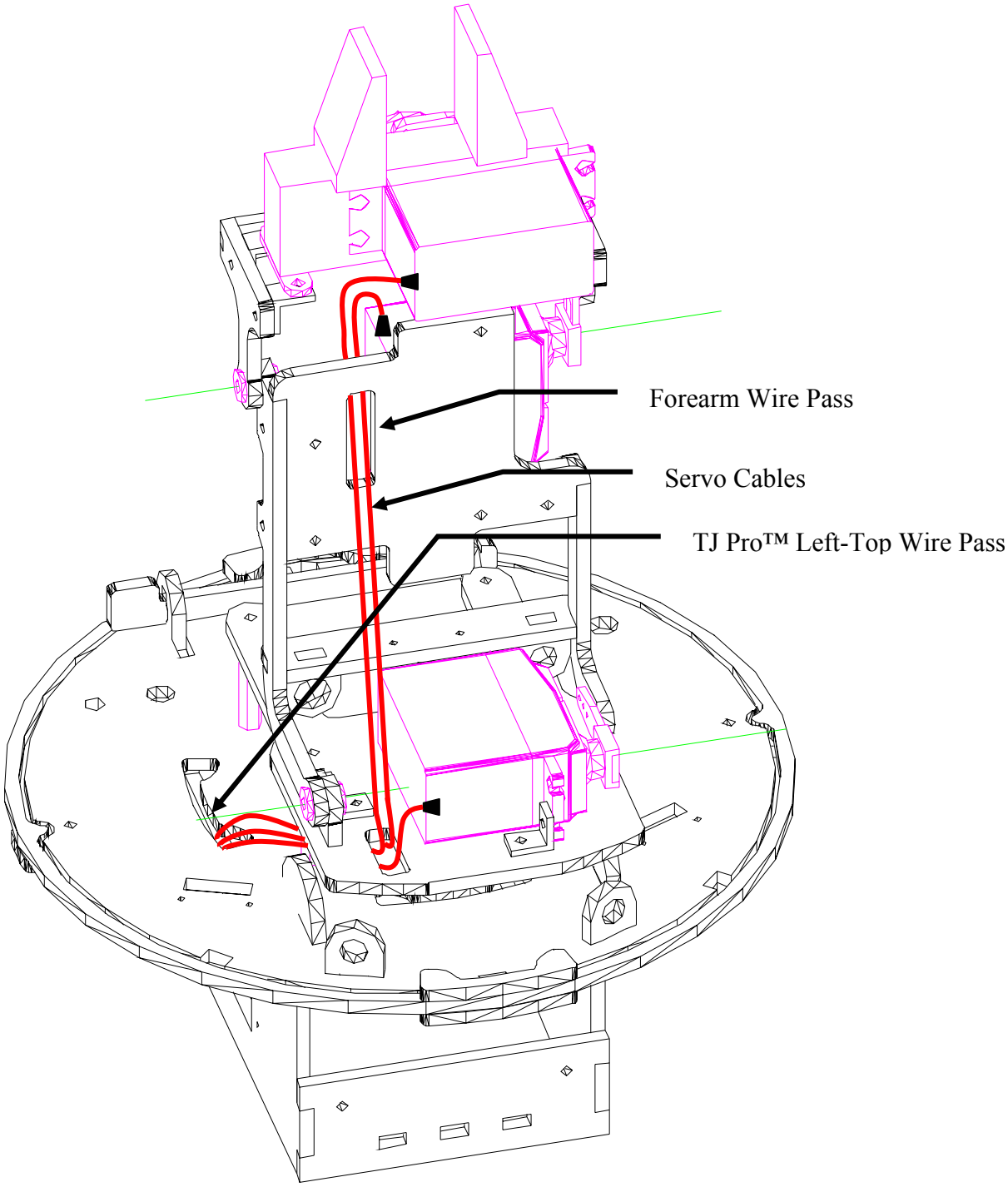



**Figure 7. The red lines indicate how to thread the three servo cables through the MekArm™ wire passes.**

# 9. SETUP SERIAL COMMUNICATION

In order to download programs to MekArm™ for execution and to receive serial communication from the arm, a serial connection on your personal computer COM1 Port must be established.

The MB2325 serial communications board (Figure 8) permits the user to download and upload code and data to the MTJPRO11 microcontroller on a TJ PRO™, MindStamp™ or other user MTJPRO11 based system via a 6-wire serial communications link (Part# C2325a). The 6-wire communications line connects into the MB2325 bidirectional serial communications board at one end (lower-right corner in (Figure 8) and to the MTJPRO11 serial interface at the other (Figure 9). The MB2325 D-connector plugs directly into a COM port of your personal computer 25 pin D-connector or through a serial cable. If your computer serial connector only has a 9 pin D-connector, you will need to acquire a 9 pin to 25 pin plug converter or cable (Part# DB9RS232MC).

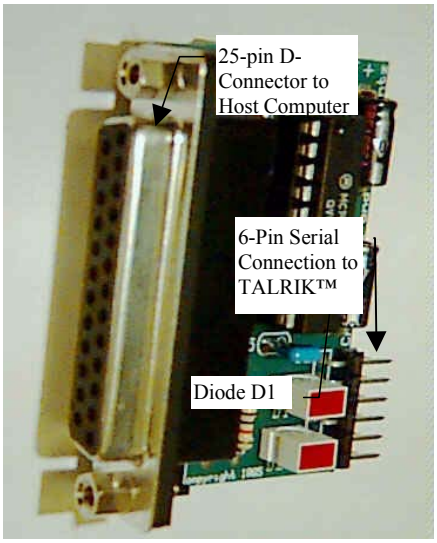



**Figure 8 MB2325 Communications Board**

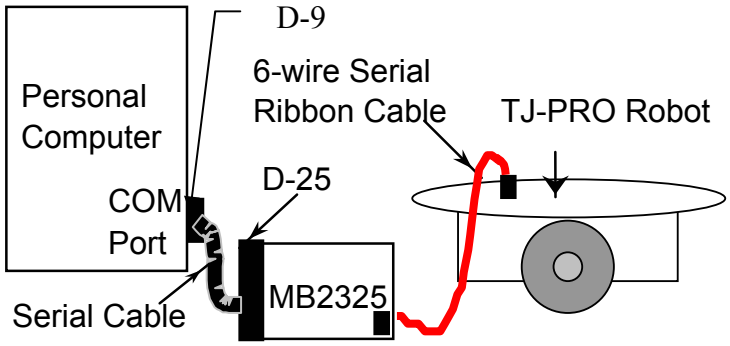



**Figure 9. This diagram illustrates the serial connection between a personal computer, the communications board (MB2325) and a Talrik Junior Professional.**

## 10. SUMMARY OF SOFTWARE INSTALLATION PROCESS

Depending on your purchases, the software installation process may involve installation of a C-compiler (Part# ICC11-w5) in addition to the *MekArm™ Distribution Software* package (Part# mekarmdst01.zip).

**Summary of Installation Process**

1. *a. Download ICC11 from the authorized Mekatronix vendor's web site and install, or*
   *b. Insert ICC11 diskette or CD and install ICC11 and remove diskette when finished.*
2. *a. Download MekArm™ Distribution Software from an authorized Mekatronix vendor's web site and install, or*
   *b. Insert the most recent version of the MekArm™ Distribution Software diskette or CD and install.*
3. *Installation of MekArm™ software essentially means copying files into appropriate directories of* `c:\icctjp`.

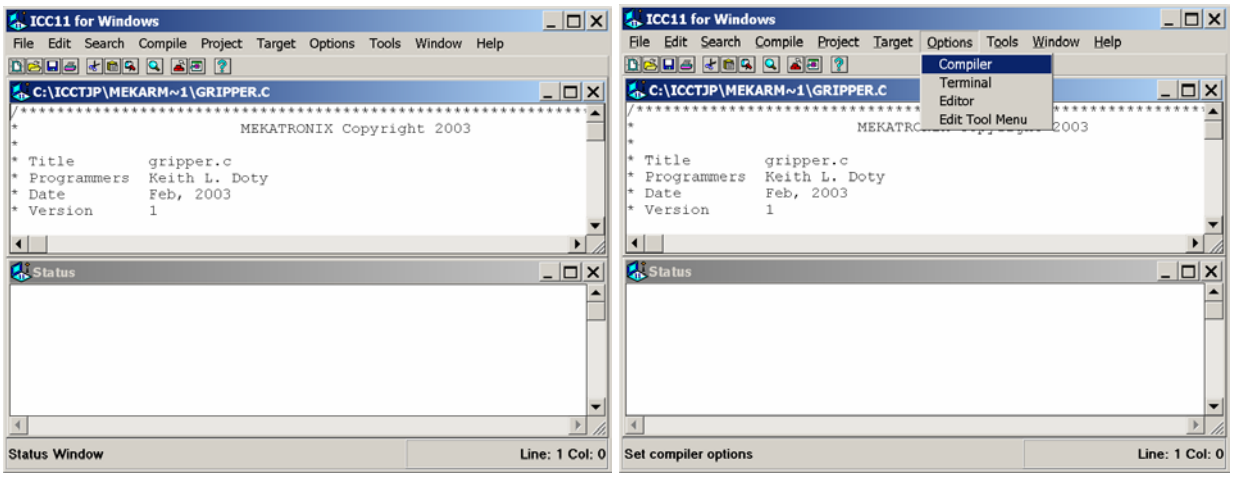
## 11. COMPILER INSTALLATION AND IDE SETUP

If you have not previously installed the ICC11 C-compiler, insert ICC11 diskette and install ICC11, or download the compiler and the ICC11 manual from an authorized Mekatronix distributor (www.mekatronix.com/distributors) web site. You will see an executable file such as `v51win.exe`. Execute it and follow directions. Specify `c:\icctjp` as your root directory. Let the installation update your DOS CONFIG file. Reboot. Put the ICC11 icon shortcut on your desktop.

After ICC11 installation, you should observe the following directory structure

- `c:\icctjp`
  - `Bin`
  - `Examples`
  - `Include`
  - `Lib`
  - `Libsrc`

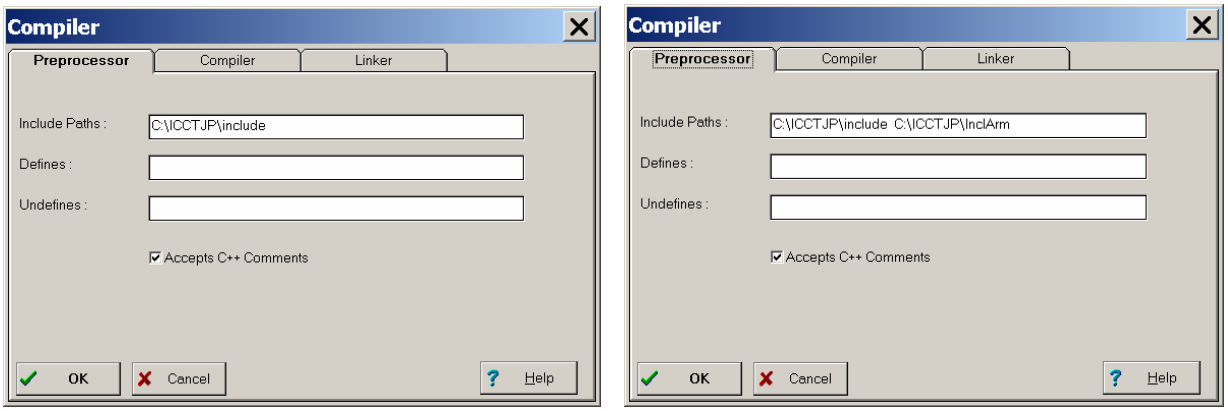
Invoking `c:\icctjp\BIN\ICC11IDE.EXE` opens a screen image of the Image Craft C (ICC11) IDE (Integrated Development Environment). In Figure 10a the source code `gripper.c` has been opened for editing, compiling and debugging. Before you can do any compiling, however, you must establish the compiler options.




a) b)

**Figure 10. a) An instance of IDE opened for use and b) Selecting Options → Compiler to configure the Preprocessor, Compiler and Linker options.**

1. Select Options→Compiler on the toolbar Figure 10b.
2. Select the Preprocessor tag, if necessary.
   In the Preprocessor screen (Figure 11a) add the path `c:\icctjp\InclArm` to the Include Paths: text box (Figure 11b). There must be at least one space between the paths.
3. Select the Compiler tag to open the Compiler screen Figure 12.
4. Leave the standard Compiler settings alone, unless, of course, you wish some of the other features to be implemented, such as Strict ANSI Checking.
5. Select the Linker tag to open the Linker screen Figure 13.
6. Fill in the Linker text boxes as shown in Figure 13.
   Note: the space between `libtjp8 libarm` in the Additional Libraries: text box is essential.




a) b)

**Figure 11. a) Standard Preprocessor settings for the IDE and b) the MekArm™ setting with `C:\ICCTJP\InclArm` added to the Include Paths: . Note, the space between paths names is essential.**

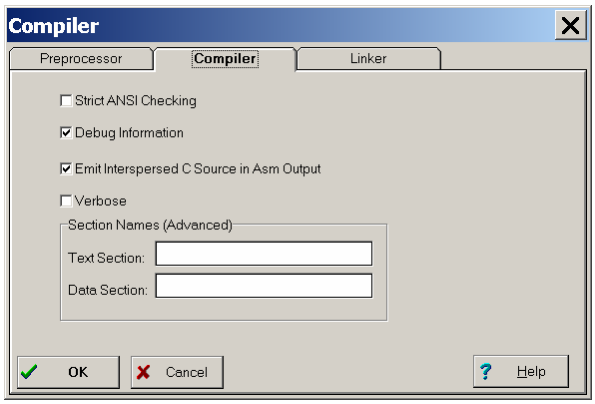



**Figure 12. Standard Compiler settings for the IDE and MekArm™.**

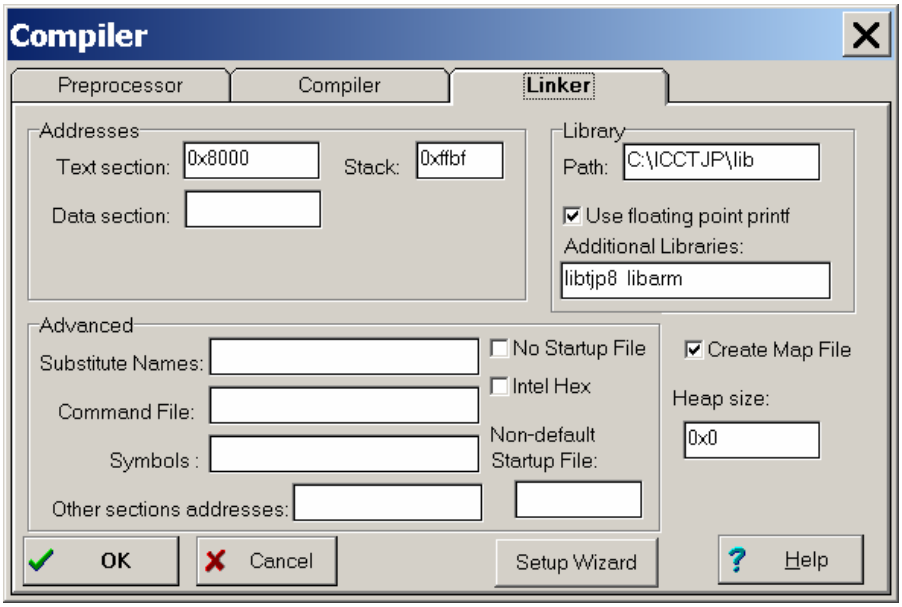



**Figure 13. Standard Linker settings for the MekArm™.**

## 12. MEKARM™ DISTRIBUTION SOFTWARE

The *MekArm™ Distribution Software*[1] supplies the basic libraries and drivers for the servo mechanism on the arm, several interactive user applications illustrate how to use the device and the MekArm™ library functions. The self-documenting source code for the library functions can be purchased separately. Refer to a Mekatronix distributor for pricing.

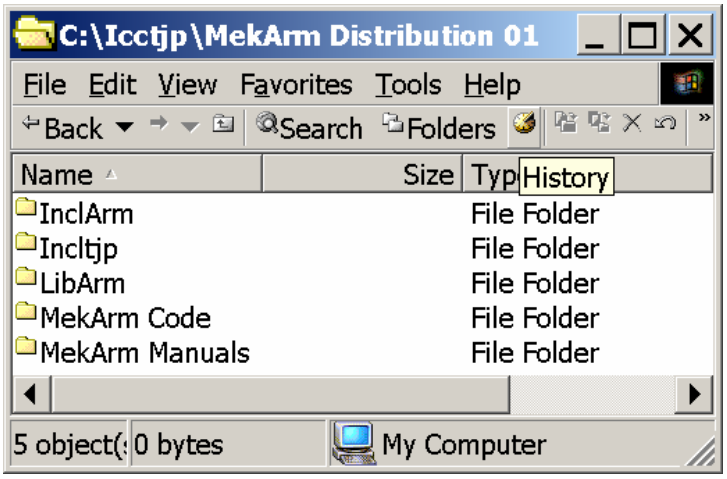
The five folders in the *MekArm™ Distribution Software* (Figure 14) must be unzipped from mekarmdst01.zip into the user's *MekArm™ folder*. If you possess a TJ Pro™ robot, your

[1] Part# mekarmdst--.zip, where -- designates version. For version 01, -- = 01 and part number = mekarmdst01.zip.

*MekArm™ folder* is `c:\icctjp`. For a MindStamp™ create a *MekArm™ folder* `c:\icctjp` from which you can now install and run MekArm and MindStamp™ applications. The TJ Pro™ setup is more general than the MindStamp™ setup. If you choose to run previously written MindStamp™ code in `c:\icctjp`, you must replace `msbase.h` with `tjpbase8`.

Of course, you can define your own *MekArm™ folder* as well.

The software package consists of the header files for the MekArm™ (Figure 15) along with updated header files of the TJ Pro™ robot (Figure 16), library archives for both the MekArm™ and the TJ Pro™ (Figure 17), MekArm™ application code (Figure 18) and the MekArm™ assembly and users manuals (Figure 19).




MekArm™ Distribution Directories

`InclArm`
MekArm™Header Files

`Incltjp`
TJ Pro™ Header Files

`LibArm`
MekArm™ Function Library
TJ Pro™ Function Library

`MekArm Code`
MekArm™ Applications

`MekArm Manuals`
Assembly and Users Manuals

**Figure 14. MekArm™ Distribution Directories**

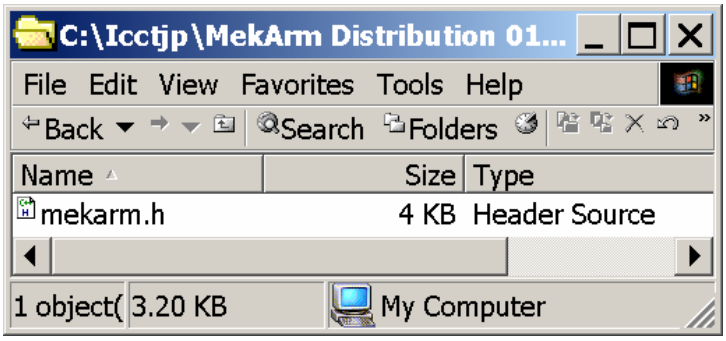



**Figure 15. MekArm Header file defines constants and global data structures used by MekArm™ programs.**

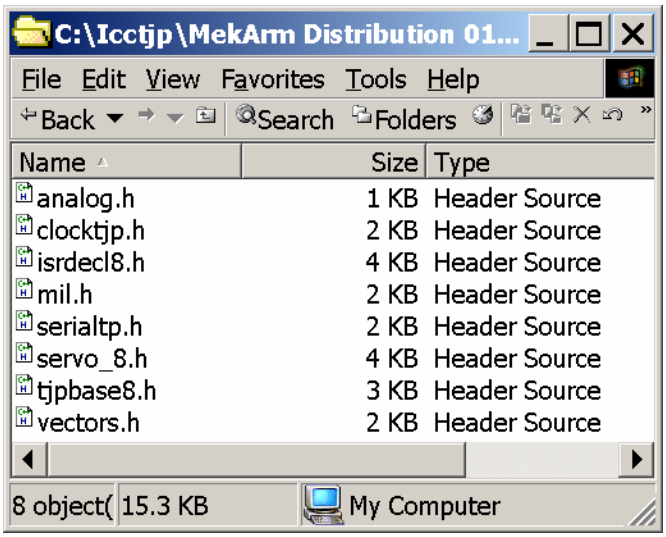



**Figure 16. Updated header files for the TJ Pro™ robot.**

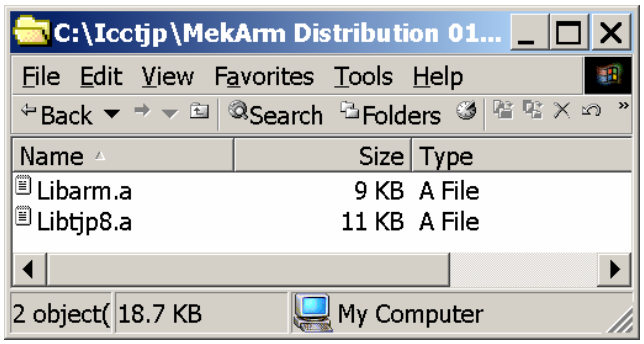



**Figure 17. Library archives for MekArm™ and the TJ Pro™ robot**

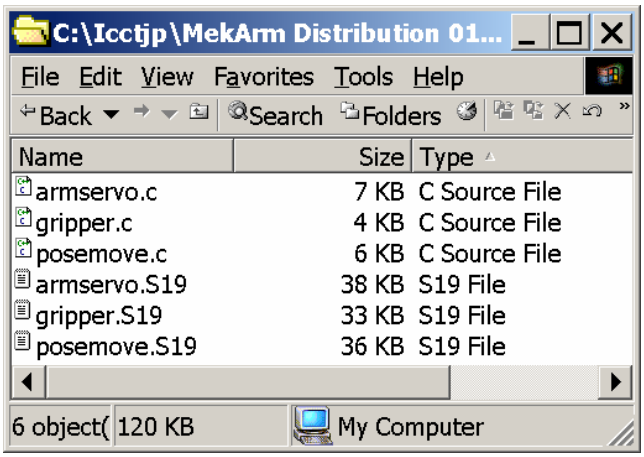



**Figure 18. Application source code <.c> and object code <.s19> for the MekArm™**

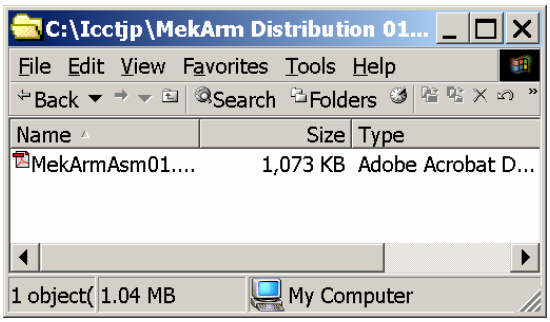



**Figure 19. MekArm™ Assembly and Users Manuals in ADOBE <.pdf> format.**

## 13. INSTALL THE MEKARM™ SOFTWARE

When you license the *MekArm™ Distribution Software*, you can either order delivery of the software on a CD at extra charge, or download the software on the web from the distributor's website for no additional charge. You must make arrangements with the distributor to obtain access codes that allow you to download the software on the web.

The five folders in the *MekArm™ Distribution Software* (Figure 14) must be unzipped from mekarmdst01.zip into the user's *MekArm™ folder* (Step 2, below). If you possess a TJ Pro™ robot, your *MekArm™ folder* is `c:\icctjp`. For a MindStamp™, create a *MekArm™ folder* `c:\icctjp` from which you can reinstall ICC11 and install and run MekArm and MindStamp™ applications. The TJ Pro™ setup is more general than the MindStamp™ setup. If you choose to run previously written MindStamp™ code in `c:\icctjp`, you must replace `msbase.h` with `tjpbase8.h` in your MindStamp™ programs. Of course, you can define your own *MekArm™ folder* as well.

You will need a C-compiler (Part# ICC11-w5) to program MekArm™. If you have not already done so,

1. Install your C-compiler (Section 10) in your *MekArm™ folder* in a WINDOWS 95 or better OS. From here on, I will assume that your *MekArm™ folder* is `c:\icctjp`. Follow the directions of the compiler *Installation Program* and select `c:\icctjp` when asked by the installer in what folder to install the compiler.

With the compiler installed

2. Unzip the *MekArm™ Distribution Software* package using WinZip™ software.
3. Move the *subdirectories* of the distribution package to your *MekArm™ folder* `c:\icctjp`.

4. Transfer archive libraries `libtjp8.a` and `libarm.a` found in `c:\icctjp\LibArm` to `c:\icctjp\Lib`. Permit these two archives to overwrite any existing archive files with the same names.
5. Remove only the Mekatronix-specific header files from `c:\icctjp\include`. Updated version of these header files are in `c:\icctjp\InclArm`. The Preprocessor setup in Section 10 will insure that the compiler can find the new header files.

At this point you are ready to setup your IDE (Integrated Development Environment) Compiler Options as follows.

# 14. EDIT, COMPILE, DOWNLOAD, AND EXECUTE PROGRAMS

After performing the previously specified hardware and software setup, you can develop, debug, download, and execute MekArm™ programs in the same manner as programming any Mekatronix robot or MindStamp™.

*CAUTION! CAUTION! CAUTION!*

*Exercise caution with the MekArm™. Even a small moving arm can cause damage to objects and people if misused or mishandled. Especially be careful during program debugging.*

## *14.1 Configuration for Program Development*

While a serial connection between the MekArm™ and your personal computer is not necessary during editing, it will be necessary for downloading and, for those applications that require serial communication, execution.

*As a standard practice, during program development,*

1. *Connect the MTJPRO11 microcontroller to your personal computer by means of the serial cables and communication board as described in Section 9.*
2. *Keep the charger on the batteries during program development.*
3. *If MekArm™ is mounted on a TJ Pro™ mobile robot, mount the robot with the wheels off the table top.*
4. *Provide enough clearance and a large enough workspace to enable MekArm™ to move collision free.*

## *14.2 Edit and Compile MekArm™ Programs*

You can use your favorite program editor or the IDE editor, but be sure to save the program as a text file with the ".c" suffix. The assumption here is that you will use the IDE editor.

Refer to Figure 20.

1. *Invoke* `c:\icctjp\BIN\ICC11IDE.EXE`
2. *Open a new program file with ".c" suffix, or one that you wish to continue editing.*
3. *Edit the program in the IDE window.*
4. *When you are ready to compile, select* Compile *on the toolbar and then* Compile To Executable *Figure 20a.*
5. *Save your new version. Make sure it has a ".c" suffix, if it does not already.*
6. *The* Status *screen (Figure 20b) will display errors, if any. A successful compile is indicated by the word SUCCEEDED.*
7. *Correct errors and repeat compile until successful.*

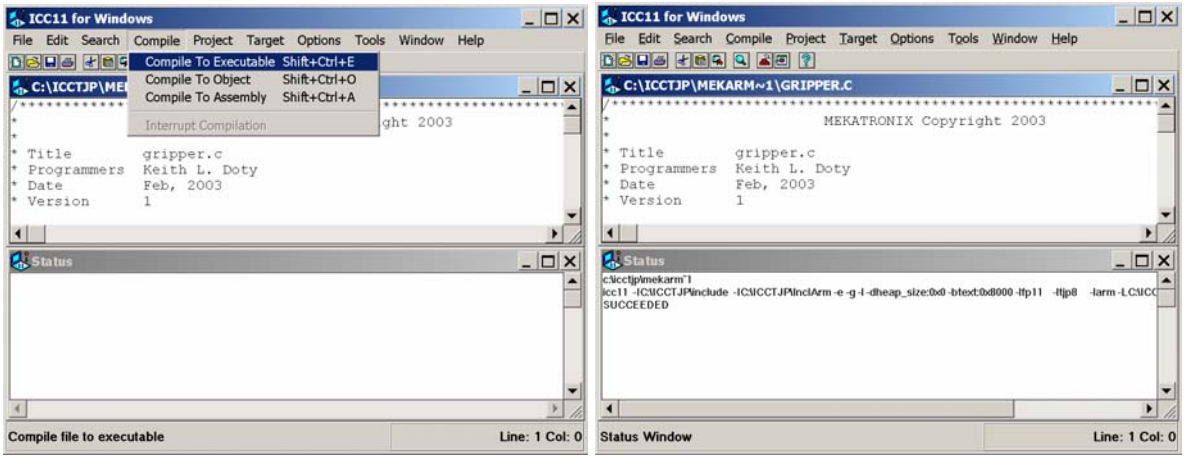
You are now ready to download a successfully compiled program.

## *14.3 Download and Execute MekArm™ Programs*

The Download procedure for Mekatronix's High-Speed Serial Downloader 11 (HSSDL11) is described in Section 15.3 , while the procedure for IDE is found in Section 16.

After a successful download, execute your program as follows,

1. *If the program is interactive, do not disconnect the serial interface and open* Terminal *or some other terminal simulator program before continuing.*
2. *Switch the MTJPRO11 to* RUN *mode.*




**a)** **b)**

**Figure 20. a) Compiling a program opened in the IDE editing window. b) A successful compile of the program in the edit window is shown in the Status window.**

3. *Press* RESET *to start program execution.*
4. *If you use the TJ Pro™* `START` *macro after initialization code in your program, you will have to press the rear bumper for the program to continue. Effectively, the rear bumper is temporarily used as a start switch. This feature is handy when transporting the robot*

*from table to floor. Of course, be careful while doing this! Inadvertent bumping the back bumper will start the robot!*

# 15. DOWNLOAD A MEKARM™ PROGRAM WITH THE HSSDL11

If your computer operates under WINDOWS 2000 or NT, you cannot use the downloader of the IDE and must instead use Mekatronix's high-speed serial downloader HSSDL11.

The low cost High-Speed-Serial–Downloader-11(Part# HSSDL11), provides an ultra fast (115.2Kbaud) serial download rate from your PC to the robot after the initial boot loader phase at 1200baud. This greatly reduces your development time and improves your productivity. Contact a Mekatronix distributor for purchasing.

Mekatronix encourages the use of the HSSDL11 software for downloading code onto the MTJPRO11 microcontroller. The HSSDL11 operates in WINDOWS 95, 98, 2000, XP and NT and can be opened simultaneously with the IDE window. You can conveniently compile and edit in IDE. After downloading using the HSSDL11, you can open up Terminal in IDE to receive data from the robot, if the program produces serial output.

*NOTE:* Terminal *in IDE and HSSDL11 both use the same COM port (default: COM1). The HSSDL11 releases the COM port after a download, even with the application window still open. Hence, HSSDL11 can remain open when* Terminal *in IDE is opened. However, you cannot download from HSSDL11 when* Terminal *or any other device captures the COM port. You will have to close the* Terminal *window or other device holding COM open before HSSDL11 can download.*

## *15.1 Installation*

Just copy the HSSDL11 directory with all its subdirectories and files into you hard drive and establish a shortcut to it from your desktop. For purposes of exposition, I will assume the path to the program is c:\HSSDL11\ DOWNLOADER.EXE.

## *15.2 HSSDL11 Setup*

The HSSDL11 remembers its setup, so you will only have to perform this procedure once or whenever you want to make changes. The name of the currently selected program object code (name.s19) appears above the HSSDL11 menu bar for convenience. Repeated compiles and loads of the same program during development require only pressing the HSSDL11 menu Download button or the d-key on the keyboard. For more HSSDL11 downloading details refer to Sections

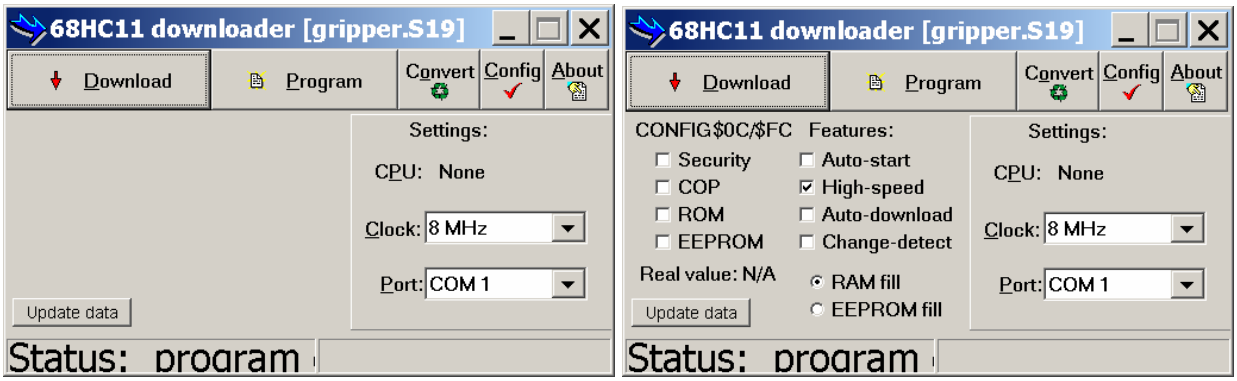
1. Open the HSSDL11. Result Figure 21a.

2. Confirm default Settings: in the drop-down boxes,
   Clock: 8MHz
   Port: COM1

3. Click on Config button, result, Figure 21b, and verify the default settings.

| | |
|---|---|
| CONFIG$0C/$FC | DO NOT CHECK ANY ITEMS HERE!<br>Default: NO item is checked. |
| Features: | Check only High-speed option.<br>Some older computers will not allow you to communicate at 115.2Kbaud and you may have to "unclick this option". The baud rate then defaults to 9600 baud. |
| RAM fill | Select |

4. Click on Config button again to leave configuration mode. Result Figure 21a.




a) b)

**Figure 21. a) HSSDL11 screen. b) Toggle the Config button to see how the HSSDL11 is configured or to change that configuration or to make the configuration selections invisible, as in a).**

## *15.3 HSSDL11 Download Procedure*

Be sure the power switch is on, the power light is on, the physical serial connection is installed, the HSSDL11 has been setup and its window open.

1. If necessary, click on Program button to browse and find the `.s19` file of interest. Select Open to open the desired .s19 file. For example, `c:\icctjp\MekArm Code\gripper.s19`.

2. Flip the *Download/Run* switch to *Download*.

3. Press the red *Reset* button.

Make sure diode D2 on the MB2325 board lights when reset is held down and goes off when reset is released. (Note, for WINDOWS 2000, NT, XP and some laptops, the lights may not work in this manner.)

4. Select Download in the HSSDL menu.
   First, the 252 bytes of the boot loader file loads at 1200 baud. Next, your program loads at 115.2Kbaud into the robot (Figure 22).

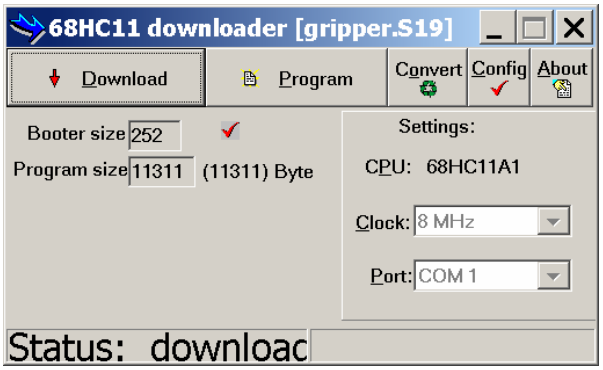



**Figure 22. a) HSSDL11 screen after a download. The screen indicates the size, in bytes, of the Boot loader and the Program downloaded.**

You are now ready to execute the code on the robot.

## *15.4 HSSDL11 Operation Tips*

We have found that under normal operations with several programs and windows open, the HSSDL11 has no problems.

*Precaution:*

*A PC executing background programs may loose synchronization with the robot serial port at these high speeds, so you may need to close such programs if you get repeated boot failures.*

If the HSSDL11 displays an error message that the COM port is not available, it means that some other application is using COM1 and must be closed before the HSSDL11 can operate.

Sometimes you will forget to press the *Reset* button prior to downloading or forget to place the microcontroller in download mode. In these situations, the HSSDL11 will signal a boot error! Check your switch settings, press *Reset* and verify the D2 light on the MB2325 board flashes on when the reset button is held down. Try again.

## *15.5 Problems with WINDOWS 2000, NT and XP*

The software and hardware protocols for the COM ports on WINDOWS 2000, NT and XP Operating Systems apparently cause difficulties for simple, direct, unbuffered serial communications with the MTJPRO11 microcontroller. Often, the HSSDL11 will report a download error in the second phase of the download process when your program's .s19 file is actually downloaded. This error report is nearly invariably false and can be ignored. Occasionally, the buffered character count or port timing, or whatever is causing the problem, works out and the HSSDL11 reports no errors! In either case, you can almost always count on a good program download. Errors in the boot loader phase, however, should never be ignored and the process repeated whenever the HSSDL11 reports a boot loader error.

If the HSSDL11 does not seem to work at all on XP, then the next section illustrates how to accommodate XP.

## *15.6 Operating HSSDL11 With XP*

Install the HSSDL11 as detailed previously. Your objective is now to execute the principal programs in WINDOWS 95 compatible mode.

1. Edit the `autoexec.bat` file to include the `C:\ICCTJP\BIN` directory in the path.
2. In Windows Explorer, select file `C:\ICCTJP\BIN\ICC11IDE.EXE` and right click on it. Refer to screen in Figure 23. Select Properties and Compatibility tab. Select Compatibility Mode and select Windows 95 in the drop-down box.
3. Establish WINDOWS 95 Compatibility Mode for `C:\ICCTJP\BIN\ISH.EXE` and `C:\HSSDL11\ DOWNLOADER.EXE` in the same manner as you did for `C:\ICCTJP\BIN\ICC11IDE.EXE`.

With these changes you should be able to run the HSSDL11 on XP, but you still may get a false error reporting by the HSSDL11 at the end of the second phase of the download…as before, an error you can usually ignore.

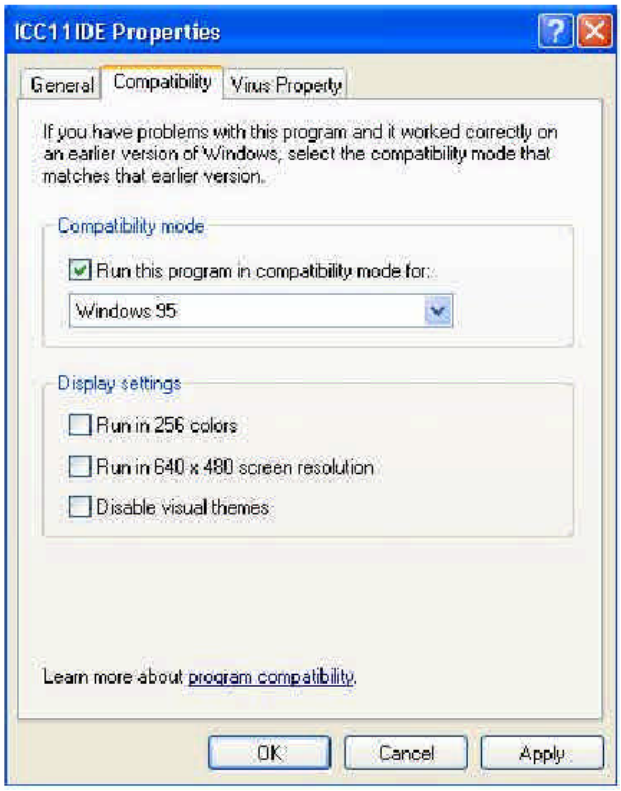


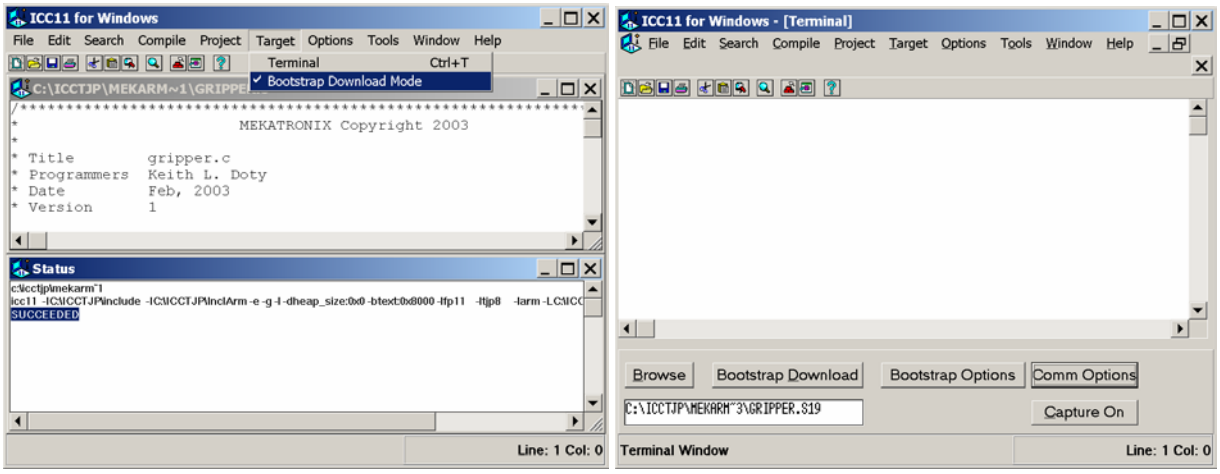

**Figure 23. Running ICC11 IDE in WINDOWS 95 Compatibility Mode.**

## 16. IDE DOWNLOADER

If you are NOT using WINDOWS 2000 or NT, the IDE Downloader option should work. This section tells you how to set up the IDE Downloader option.

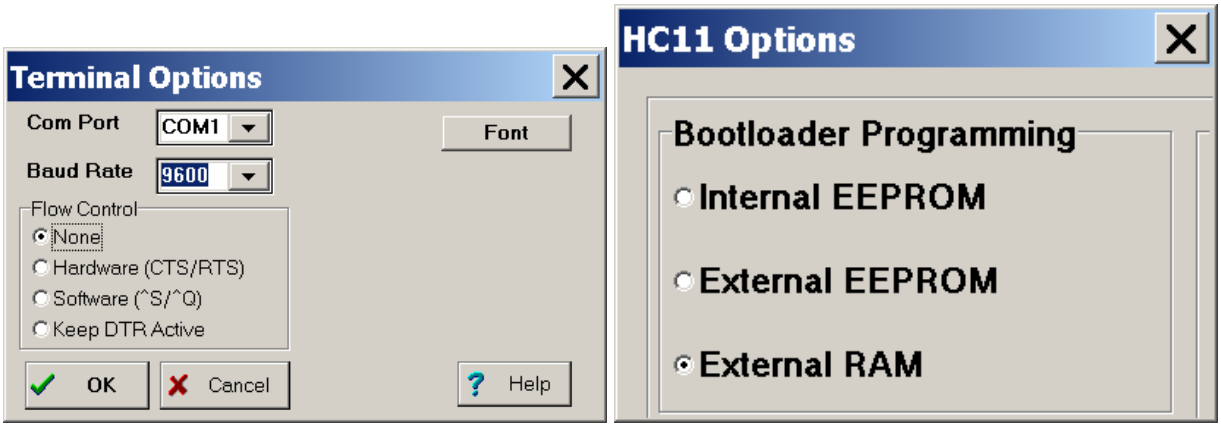
1. Select Target and toggle Bootstrap Download Mode to check it, if it is not already checked.
2. Select Target again and select Terminal with Bootstrap Download Mode checked (Figure 24a).
3. In Terminal (Figure 24b) select Comm Options, if this is your first-time ever use of Terminal, or, if you wish to change a communications parameter (Figure 25a). Close when finished.
4. In Terminal, fill out Bootstrap Options if this is your first-time ever use, or, if you wish to change MTJPRO11 memory selection (Figure 25b).Close when finished.
5. In Terminal, enter file path and name of the `.s19` file to be downloaded into the text box in the lower-left. Or, Browse to get file path and file (Figure 24b) to be downloaded. The example file show is `C:\ICCTJP\MEKARM~3\GRIPPER.S19`.
6. Select Boostrap Download, follow the instructions and watch the lights on the MB2325 flash.

After a successful download, switch the MTJPRO11 to *RUN* mode and press *RESET* to start program execution. If the program is interactive, you must leave the serial connection and open Terminal or some other terminal simulator program before executing the program.




a) b)

**Figure 24. a) Enable Bootstrap Download Mode or select Terminal. b) Terminal open with Bootstrap Download Mode option checked. Text box in lower-left registers path to `.s19` object file. You can change the path with the Browse button.**




a) b)

**Figure 25. a) Enter Comm Options from Terminal screen b) Bootstrap Options , select External RAM.**

## 17. APPLICATIONS OF MEKARM™

Mounted on a mobile robot, MekArm™ provides a vital function missing from previous Mekatronix robots, the ability to manipulate objects. This capability propels the TJ Pro™ robot into a higher plane of performance and functionality.

A companion *MekArm™ Education Manual* discusses manipulator characteristics and provides source code for *Coordinated* and *Parallel* motion control. For details and pricing ask a Mekatronix distributor.

While designed specifically for the TJ Pro robot, in the near future, you will be able to attach MekArm™ to other Mekatronix robots, the Talrik II and Robobug. You can also use MekArm™ in stand-alone mode wherein the manipulator mounts on a platform, for example, the MindStamp™, or one of your own design that houses an MTJPRO11 microcontroller.

# 18. MEKARM™ LIBRARY FUNCTIONS

The header file `mekarm.h` conveniently divides, for purposes of discussion, into two parts. One part (Figure 26) illustrates the defined constants and predefined MekArm™ configurations, or poses, and the other part (Figure 27) consists of the library functions in libarm.a and global data structures available for external use.

The two dimensional array `ArmPose[N_POSES][N_JOINTS]` (Figure 27) allocates space for `N_POSES=20` predefined poses, of which the first nine are predefined by Mekatronix and should not be changed. A programmer can change the remaining eleven configurations to any pose desired.

The named configurations consist of two numbers, a shoulder servo setpoint and a wrist servo setpoint. When those two servos move to the specified positions, the arm will be in the named configuration. Refer to Section 20 to see what the arm looks like in the Mekatronix named configurations.

The macros

```
#define SHOULDER_JOINT_NUM    4
#define WRIST_JOINT_NUM       5
#define FINGER_JOINT_NUM      6
```

define the servo numbers of the MekArm™ servos that are used in the servo command

```
servotjp(servo_number, servo_setpoint).
```

This function definition is found in header file `c:\icctjp\Incltjp\servo_8.h`.

## *18.1 The armMove() Function*

The default “pose” for unused slots should be `REST`, which is not really a pose, but, rather, idles the shoulder and wrist servos. A `REST` pose command given to the arm in some configurations

will result in the arm moving under the action of gravity to an unknown configuration. This is an undesirable situation, since the arm move command

```
armMove(int shoulder_end_pt, int wrist_end_pt)
```

remembers the previous configuration of the arm and will assume it to be there, even though gravity has changed the configuration. Consequently, the next `armMove(s,w)` command will cause the arm to move, possibly violently, from the gravity determined configuration to the assumed configuration at the beginning of the new move command. For example, the programmed command sequence

```
…
armMove(EXTEND);
armMove(REST);
armMove(HOME);
…
```

will, first, extend the arm (Figure 32) from its current position, which must be `HOME` if this is the very first `armMove(s,w)` command in the program.

***WARNING! WARNING! WARNING!***
*ALWAYS PLACE MEKARM™ IN THE HOME CONFIGURATION*
*BEFORE STARTING ANY MEKARM™ PROGRAM*

The next command turns the servos off and the arm will drop to the floor under the action of gravity. Since no joint feedback sensors exist, the arm will assume a configuration unknown to the software. The next command `armMove(HOME)` assumes the arm configuration is `EXTEND` and will cause the arm to move violently to the `EXTEND` position from the gravity determined position and, thence, to the `HOME` position in a normal, coordinated way.

The important characteristics of the library function `armMove(s,w)` are

1. At startup, the initial arm configuration must be `HOME`.
2. `armMove(s,w)` remembers the previous arm configuration.
3. `armMove(s,w)` moves the shoulder servo to setpoint `s` and the wrist servo to setpoint `w`.
4. `armMove(s,w)` moves the joints in a *coordinated* manner, namely, all the joints start and stop at the same time during the motion from a known initial configuration to the specified final configuration, regardless of the amount of motion of each joint.

Header file `c:\icctjp\Incltjp\servo_8.h` specifies the shoulder and wrist servo setpoint limits by means of the global array

```
unsigned int servo_limits[MAX_SERVO][2]=
 {
      {PW0L,PW0U},
```

```
        {PW1L,PW1U},
        {PW2L,PW2U},
        {PW3L,PW3U},
        {PW4L,PW4U},            // Shoulder
        {PW5L,PW5U},            // Wrist
        {PW6L,PW6U},            // Finger
        {PW7L,PW7U}
};
```

The values of the constants in the array are defined earlier in the file and can be changed if necessary. The first constant sets the lowest limit and the second constant the upper limit of a setpoint. For example, for the shoulder servo, the setpoint must satisfy,

```
PW4L ≤ shoulder_setpoint ≤ PW4U
```

The global variable

```
unsigned char srv_lim_default= SRV_LIMIT_DEFAULT;
```

defined in that same header file specifies whether the servo driver will enforce the limits or not. The default value `SRV_LIMIT_DEFAULT = 1` enables enforcement. Assigning `FALSE` or `0` to `srv_lim_default` in a program disables enforcement. Typically, limit checks should only be disabled when you calibrate your servos to determine their ranges.

*WARNING! WARNING! WARNING!*
*You can burn out a servo if you drive it past its limits for more than a minute or so.*

## *18.2 The joint_move() Function*

The function `armMove()` uses `joint_move()` but, for most applications, the former is sufficient and, therefore, separate use of the later is not normally needed. The function is described here for completeness and for those programmers who wish to develop their own motion control functions.

This function permits you to move a joint from an assumed valid, current setpoint to a new setpoint in steps of `delta_move` :

```
joint_move(unsigned char joint_N, int joint_new_setpt, int
           joint_current_setpt, int delta_move );
```

As you see, the function requires four parameters, the joint number, its new setpoint, its previous setpoint and the step size. The step size determines how the arm moves incrementally and must be negative if

```
joint_new_setpt < joint_current_setpt.
```

## *18.3 The Function fingerGapSet()*

The function `void fingerGapSet(unsigned char gap)` allows you to command the fingers to open or close to a specified width, measured in hundredths of inches, with an accuracy of about ±0.02 inches. To compensate for the non-linearity of the gripper servo, which also exhibits hysteresis, two arrays were generated from measurements of the finger gap.

The array

`dec_gap_hundredths_inch[]` (decreasing gap size with increasing index number)

was generated by taking measurements at gripper servo settings starting with `MIN_FINGER_SET_PT=3000` and ending with `5300`, taken in `FINGER_STEP_SIZE=100` step increments, for a total of `N_GAPS = 24` measurements. The array

`inc_gap_hundredths_inch[]` (increasing gap size with decreasing index number)

was generated by measurements of the finger gap with gripper servo settings starting at `5300` and ending at `MIN_FINGER_SET_PT=3000`, in `FINGER_STEP_SIZE=100` step decrements, for a total of `N_GAPS = 24` measurements.

The measured finger gap values, in hundredths of an inch and listed below, also appear in the `mekarm.h` header file and can be changed by the programmer if necessary. The actual measurements were made between the fingers and not the finger pads. The 0.125 inches of each pad was subtracted to get the numbers below. Since the soft pads compress easily, and the hard plastic sides of the fingers do not compress (at least not with the force exerted by a caliper!), this technique made for more reliable and accurate measurements of the finger gap for each setpoint of the gripper servo.

```
const unsigned char dec_gap_hundredths_inch[N_GAPS] =
   {
     129, 128, 126, 122, 118, 113, 108, 101, 95, 87, 80, 71,
      63, 56, 49, 34, 34, 30, 24, 18, 13, 10, 7, 2
   };


const unsigned char inc_gap_hundredths_inch[N_GAPS] =
   {
     129, 128, 126, 122, 118, 113, 108, 101, 95, 87, 80, 72,
      63, 56, 47, 29, 29, 24, 18, 13, 10, 5, 2, 2
   };
```

The function `fingerGapSet(gap)` use the above gap arrays to determine the gripper servo setpoint that will generate the gap, using linear interpolation between measured points. Consistent accuracy of ±0.02 has been obtained. However, please note that backlash in the gripper drive mechanism can increase this error by another several hundredths of an inch.

```
/*********************************************************************
*                        MEKATRONIX Copyright 2003
*
* Title         mekarm.h
* Programmers   Keith L. Doty
* Date          Feb, 2003
* Version       1

* Description: Constants and prototypes for MekArm.

*********************************************************************/
#ifndef MEKARM_H
#define MEKARM_H

//Constants
#define N_POSES              20
#define N_JOINTS              2
#define SHOULDER_JOINT_NUM    4
#define WRIST_JOINT_NUM       5
#define FINGER_JOINT_NUM      6
#define JOINT_STEP_SIZE       10
#define FINGER_STEP_SIZE      100
#define MIN_FINGER_SET_PT     3000
#define N_GAPS                24
#define STEP_TIME             20
#define TRUE                  1
#define FALSE                 0


/* Named MekArm Configurations */
#define REST         0000,0000
//Policy: move MekArm to HOME and then REST when not active
#define HOME         4600,4500
#define SALUTE       4600,3050
#define TRANSPORT    4600,1650
#define EXTEND       3200,3050
#define PRESENT      3200,4500
#define HIGH_PICK    3200,1680
#define LOW_PICK     2400,2350
#define FORK_LIFT    1700,4500
```

**Figure 26. The macro defines contained in the `mekarm.h` file consists of named numerical constants used by the library functions in `libarm.a` and the named MekArm™ configurations.**

```
//Prototypes

//Move arm from current setpoints to those specified in the arguments
   void armMove(int shoulder_end_pt, int wrist_end_pt);

//Move a joint from current setpoint to the new setpoint in steps of <delta_move>
void joint_move(unsigned char joint_N, int joint_new_setpt, int joint_current_setpt,
                int delta_move );

//Open the fingers to a width of <gap>
   void fingerGapSet(unsigned char gap);

//Globals

unsigned int ArmPose[N_POSES][N_JOINTS]= {
         { REST },     { HOME },      { SALUTE },    { TRANSPORT },
         { EXTEND },   { PRESENT },   { HIGH_PICK }, { LOW_PICK },
         { FORK_LIFT },{ REST },{ REST },{ REST },{ REST },
         { REST },{ REST },{ REST },{ REST },{ REST },{ REST },{ REST }};
/*
 The finger gap does not change linearly with the servo setpoint and it exhibits
 hysteresis: the new gap value depends on whether the gap is increasing or
 decreasing from the current position.The following two variables store those gap
 values for a particular gripper. If your gripper varies significantly from these
 numbers, use the program <fingerCntrl.c> to create your own table. The difference
 in servo setpoint between each entry must be 100 steps. The range of steps equals
 3000 to 5300 for a total of 23 steps: 3000 + 23*100 = 5300.
*/

// Use dec_gap_hundredths_inch[N_GAPS]when new gap is less than current gap
   const unsigned char dec_gap_hundredths_inch[N_GAPS] =
      {
        129, 128, 126, 122, 118, 113, 108, 101, 95, 87, 80, 71,
         63, 56, 49, 34, 34, 30, 24, 18, 13, 10, 7, 2
      };

// Use inc_gap_hundredths_inch[N_GAPS]when new gap is greater than current gap
   const unsigned char inc_gap_hundredths_inch[N_GAPS] =
      {
        129, 128, 126, 122, 118, 113, 108, 101, 95, 87, 80, 72,
         63, 56, 47, 29, 29, 24, 18, 13, 10, 5, 2, 2
      };
/*********************************************************************/


#endif
```

**Figure 27. The remainder of the `mekarm.h` file consists of prototype functions and data structures used by the library functions in `libarm.a`.**

# 19. OPERATION OF THE MEKARM™

Once you have assembled, mounted, and connected the MekArm™ to your robot, and read Section 4 on safety, you may desire to run the sample application programs (Figure 18) that come with the *MekArm™ Distribution Software*.

The three application programs

```
armservo.c
gripper.c
posemove.c
```

provide interactive control of MekArm™ from a terminal simulation program on your personal computer. The program `armservo.c` allows you to interactively change the shoulder, wrist, and gripper servos and observe the resulting MekArm™ motion. The program `gripper.c` allows you to interactively control the finger gap in terms of inches or in terms of the gripper servo setpoint. In the former case, the setpoint is calculated and printed on the screen. In the latter case, the gap in inches is calculated and printed on the screen. To learn more about these two programs just download their .s19 files and execute. If you want further information, open them in IDE and read the comments and the code. The program `posemove.c` will be discussed in more detail.

The `posemove.c` interactive program permits interactive control of MekArm™ in a simple direct way. The user simply selects one of the named configurations and the arm moves to that location. The program also allows you to interactively control the gripper finger gap. The next few paragraphs detail the use of this program.

1. Download `posemove.c` into the MTJPRO11 memory.
2. Execute a terminal simulation program on your personal computer. For example, `Hyper Terminal` in `autodetect` emulation mode or `Terminal` on the IDE.
3. Put the microcontroller into *RUN* mode and press *RESET*.

You will see the following message on your screen.

```
CAUTION! CAUTION! CAUTION! CAUTION!

FOR YOUR SAFETY AND PROTECTION
KEEP HANDS AND FACE CLEAR OF MOVING ROBOT.
MEKARM CAN THROW OBJECTS, KEEP YOUR DISTANCE.

To signify you have read this message and are
ready to proceed, press Y or y for yes.
```

After reading the above message, press the y-key on your keyboard. Another message will appear on your screen.

```
WARNING! WARNING! WARNING! WARNING!

FOR YOUR SAFETY AND PROTECTION AND
TO PROTECT YOU AN THE ROBOT FROM DAMAGE,
MANUALLY PUT THE ROBOT IN HOME POSITION
BEFORE CONTINUING WITH THIS PROGRAM.

IS THE ROBOT IN HOME CONFIGURATION?

After you put the robot in HOME CONFIGURATION
Press Y or y for yes to proceed.
All other keys are no.
```

To avoid risk of injury to yourself and the MekArm, do not press the y-key until you have manually moved MekArm™ into the `HOME` configuration (see Figure 29).

At this point your screen will display the text shown in Figure 28. By pressing a single key, you can move the arm from its current configuration to one of the eight named configurations (refer to Section 20 to see how each configuration appears). The keys ‘p’, ‘c’, and ‘z’ allow you to control the gripper for any arm configuration. The gripper gap is initially set at 1 inch (100 hundredths, `Current Finger Gap (Hundredths)`) and the gripper step size at 5 hundredths of an inch (0.05 inch, `Current Finger Step Size (Hundredths)`). You can change the gripper step size, measured in hundredths of an inch, by pressing the s-key and entering a number from 1 to 100.

For each p-key press the gripper opens up another 10 hundredths of an inch until it reaches the largest gap which is a multiple of 10 from the current gap. For each c-key press the gripper closes up another 10 hundredths of an inch until it reaches the smallest gap which is a multiple of 10 from the current gap.

The `Current Configuration` is shown to be 1, i.e., `HOME`. You can now enter a digit from 1 to 8 to move the arm to another configuration. If you enter one of the keys ‘p’, ‘c’, and ‘z’, then the program switches to Gripper Control at the bottom of the screen.

Whatever the position of MekArm™, a command to `REST` (Enter 0) idles the servos, regardless of the configuration of the arm. A `REST` command will allow the arm to collapse to its lowest gravitational potential energy state. Therefore, do not execute this command except in the “safe `REST` configurations”, especially if you do not want the arm to move under the influence of gravity after the command is executed.

Safe Rest Configurations: HOME, SALUTE, TRANSPORT, FORK_LIFT

The SALUTE configuration is, technically unstable in REST mode, but, because of gear friction, the arm usually does not move. If the gripper were to carry a sufficiently heavy load, the TRANSPORT configuration would also become unstable with a REST command. The load limitations of the arm, however, make it unlikely an operational load will create enough gravitational force to overcome the shoulder gear friction.

Enjoy your MekArm™ (☺ )!

```
MEKARM COORDINATED MOTION FOR DEFINED CONFIGURATIONS

        Programmer      Keith L. Doty
        Date            Feb, 2003,
        Version         1

 MEKARM CONFIGURATIONS
        0 REST
        1 HOME
        2 SALUTE
        3 TRANSPORT
        4 EXTEND
        5 PRESENT
        6 HIGH_PICK
        7 LOW_PICK
        8 FORK_LIFT

 GRIPPER CONTROL
   Hold key down for continuous action.
        p. Press 'p' to oPen Gripper
        c. Press 'c' to Close Gripper
        z. Press 'z' to Turn Off Gripper Servo
        s. Press 's' to enter new step size

     Current CONFIGURATION:                     1
     Current Finger Gap (Hundredths):           100
     Current Finger Step Size (Hundredths):     5

     Select a control(0,1,2,3,4,5,6,7,8,p,c,z,s):
```

**Figure 28. This illustrates the display on a terminal simulator generated by execution of the program `posemove.c`.**

# 20. MEKARM™ NAMED CONFIGURATIONS

Mekatronix defines 8 named configurations for the MekArm™ and a `REST` mode in any arbitrary configuration achievable by the arm. The named configurations, given below, appear in the header file `mekarm.h` and are illustrated in Figure 29 through Figure 36 .

```
/* Named MekArm Configurations */
  #define REST          0000,0000
  #define HOME          4600,4500
  #define SALUTE        4600,3050
  #define TRANSPORT     4600,1650
  #define EXTEND        3200,3050
  #define PRESENT       3200,4500
  #define HIGH_PICK     3200,1680
  #define LOW_PICK      2400,2350
  #define FORK_LIFT     1700,4500
```

The two numbers beside each name in the header file provide the (shoulder_servo, wrist_servo) settings for that configuration. For example, the configuration `LOW_PICK` is produced when (shoulder_servo, wrist_servo) = (`2400,2350`). Execution of

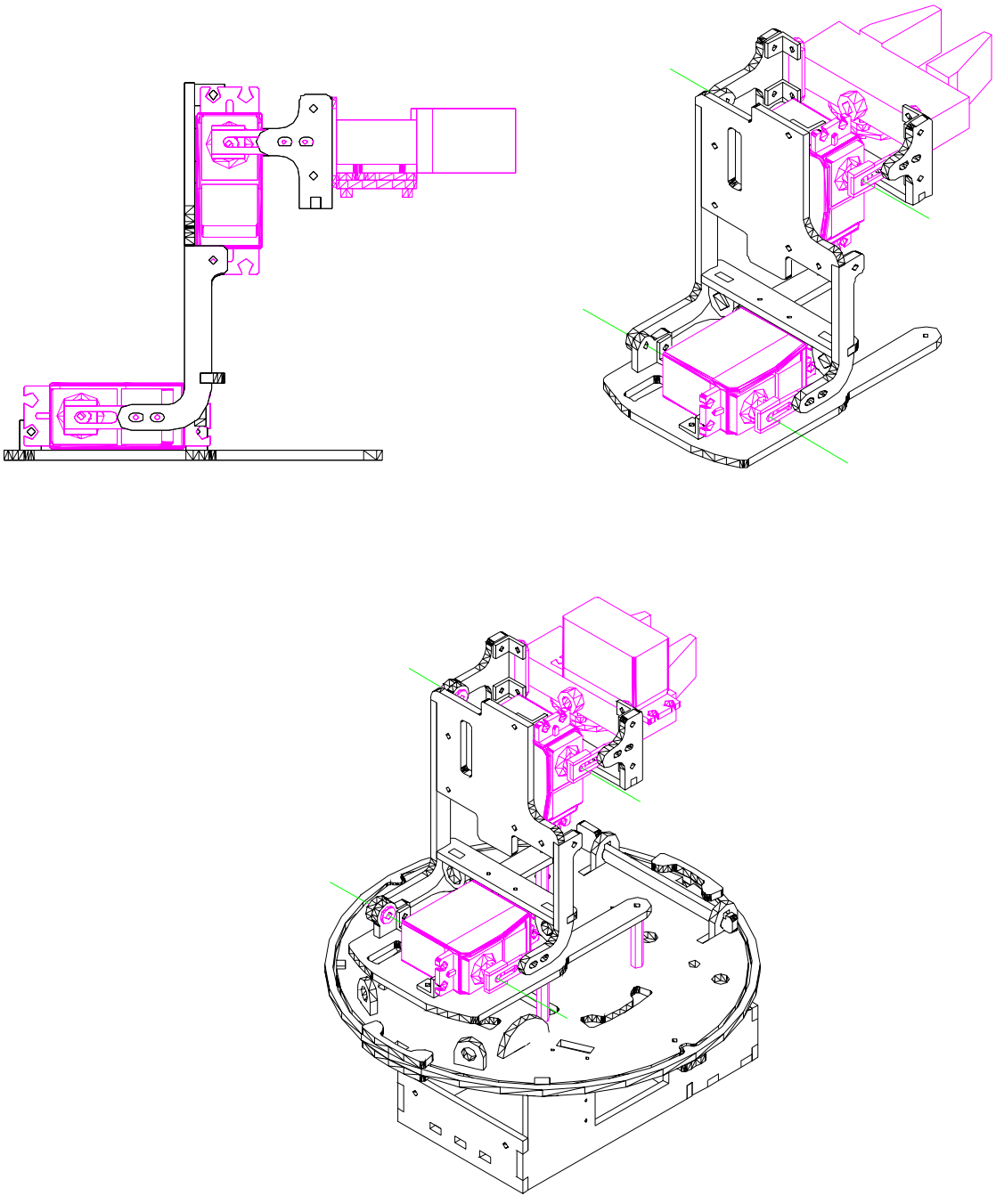
```
armMove(LOW_PICK);
```

for example, is equivalent to the execution of the statement

```
armMove(2400,2350);
```

For most MekArm™ programs, you will probably be able to stay with these defined configurations and just move the arm from one to another to implement your application requirements.

**Figure 29 HOME CONFIGURATION or POSE**



316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

**Figure 30 SALUTE CONFIGURATION or POSE**

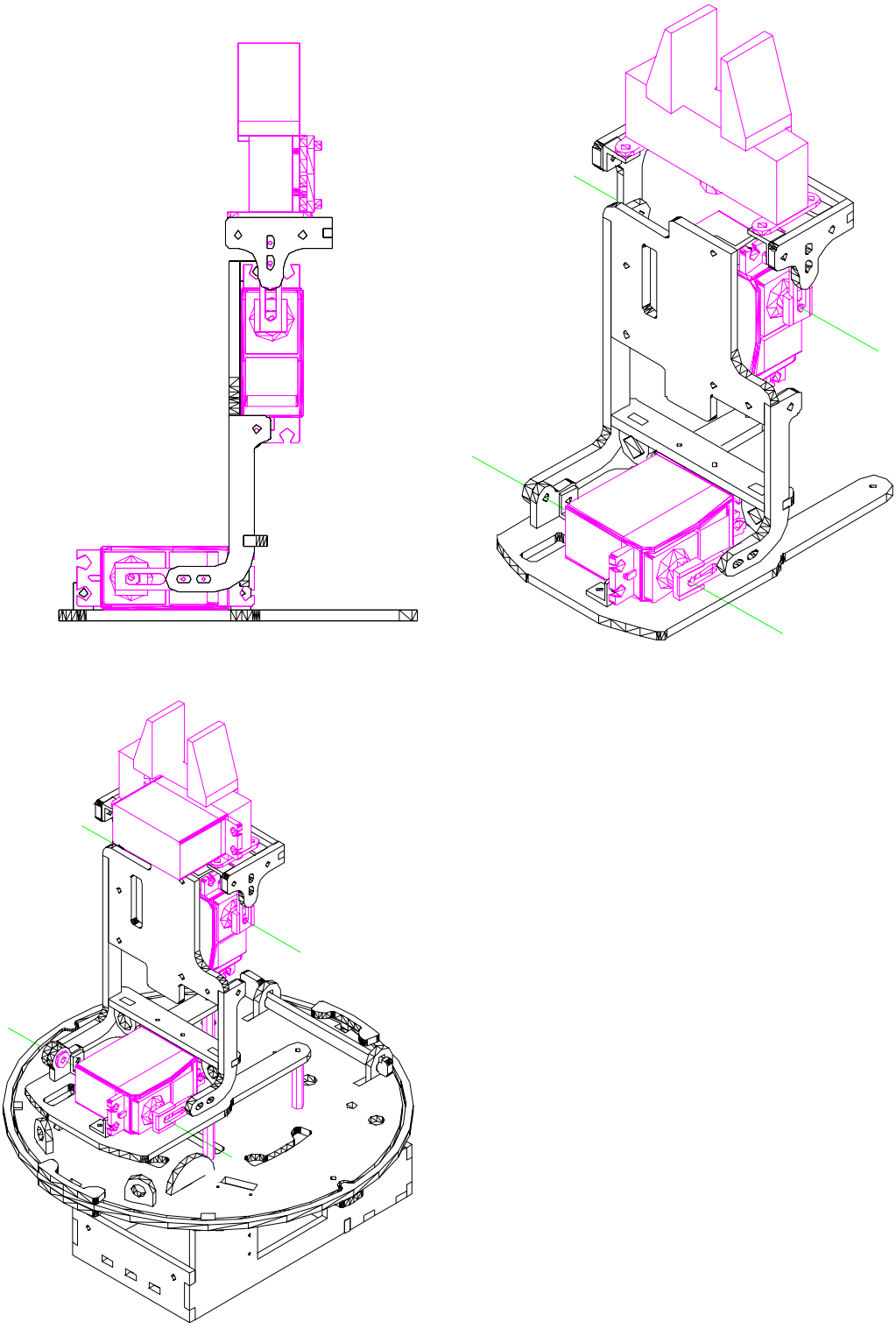


**Figure 31 TRANSPORT CONFIGURATION or POSE**

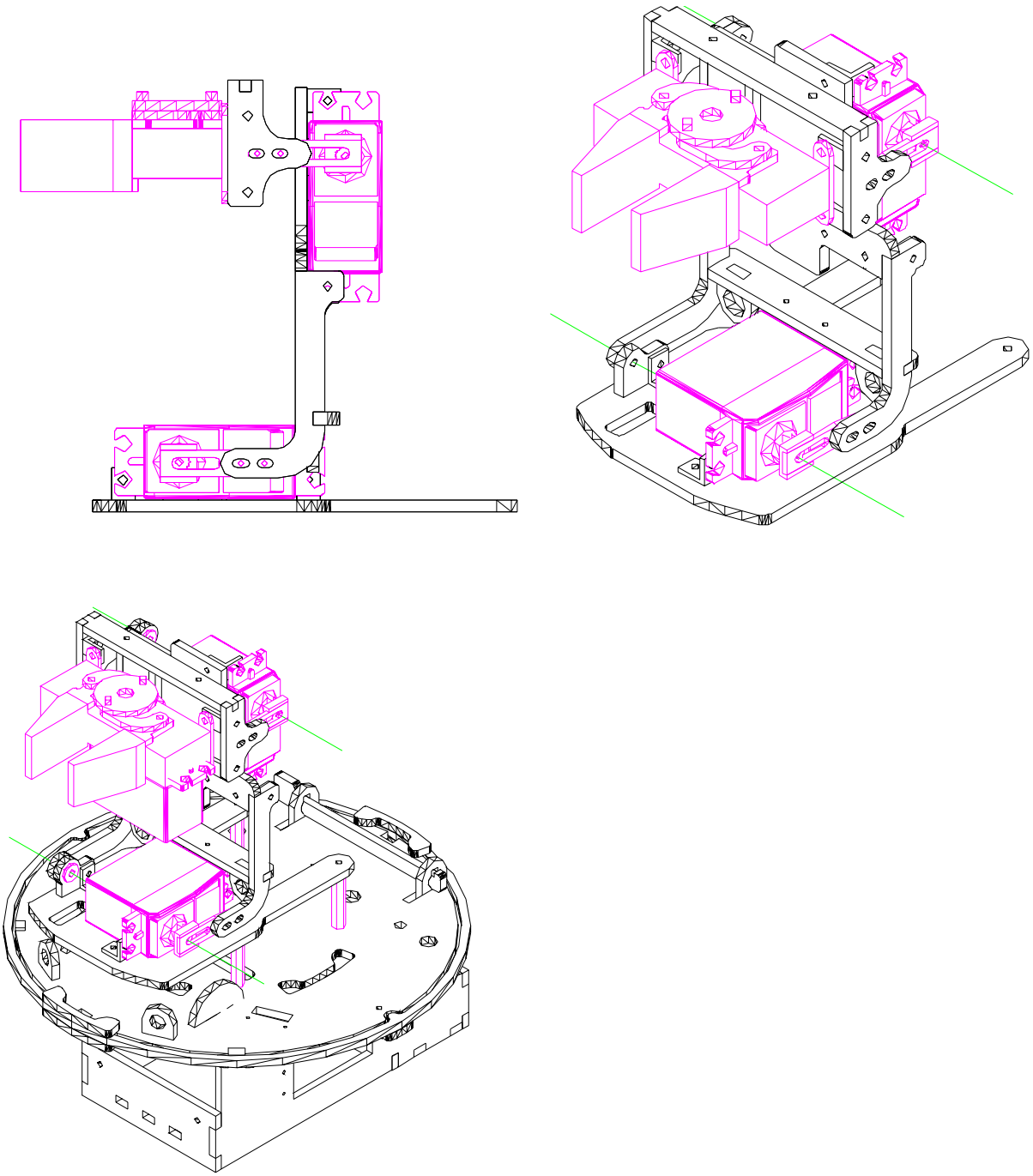
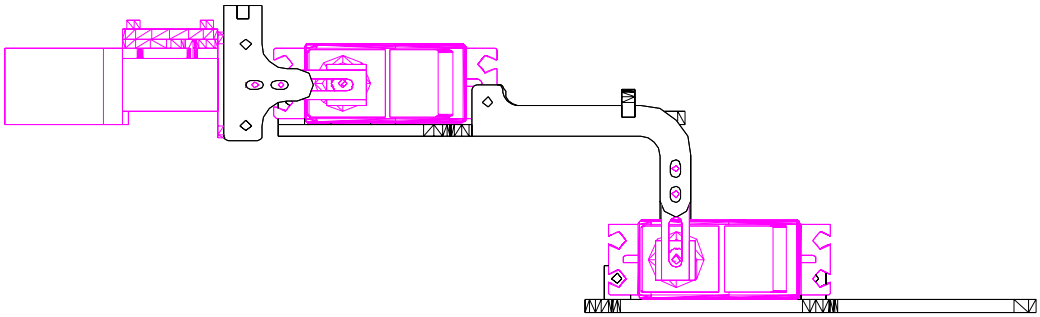
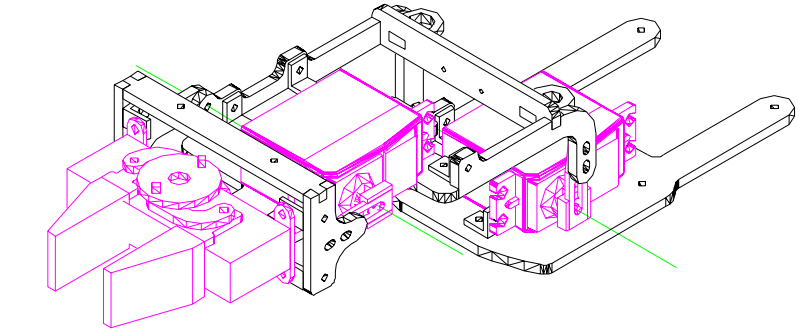


**Figure 32 EXTEND CONFIGURATION or POSE**





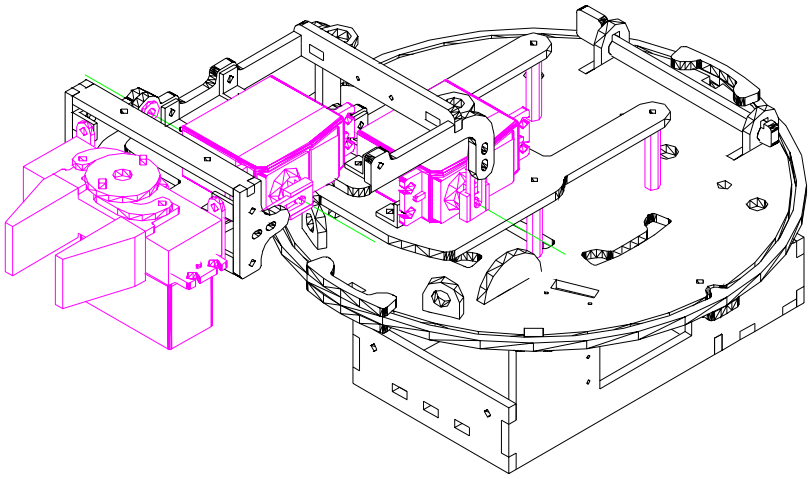


**Figure 33 PRESENT CONFIGURATION or POSE**

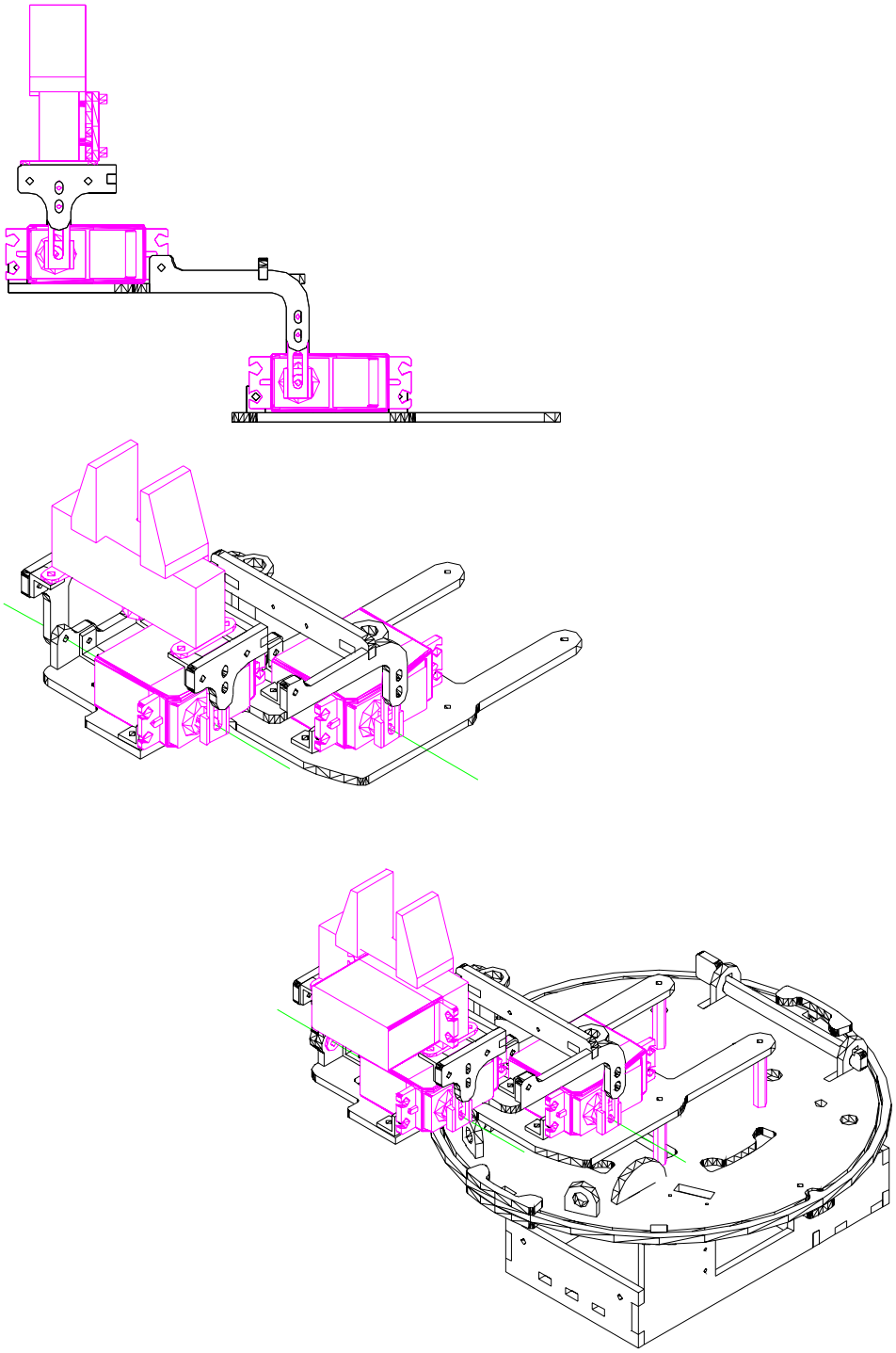


**Figure 34 HIGH PICK CONFIGURATION or POSE**

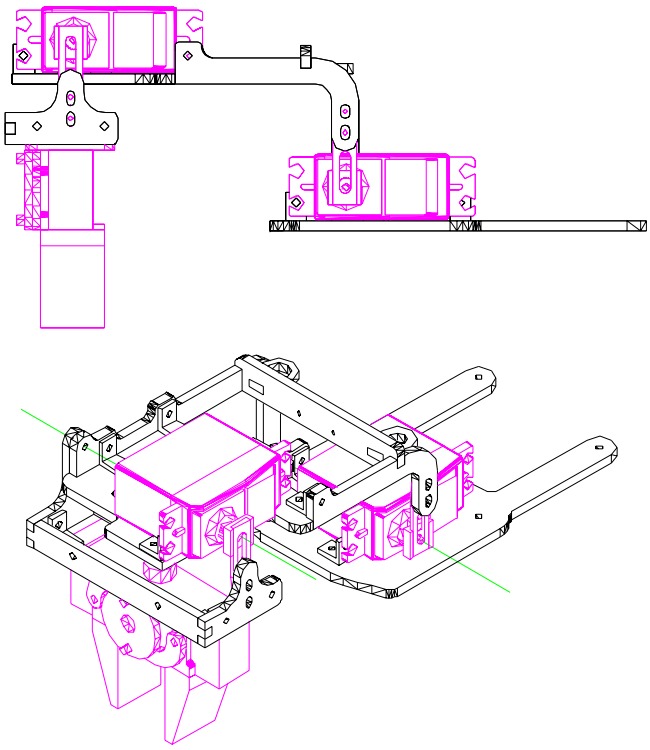
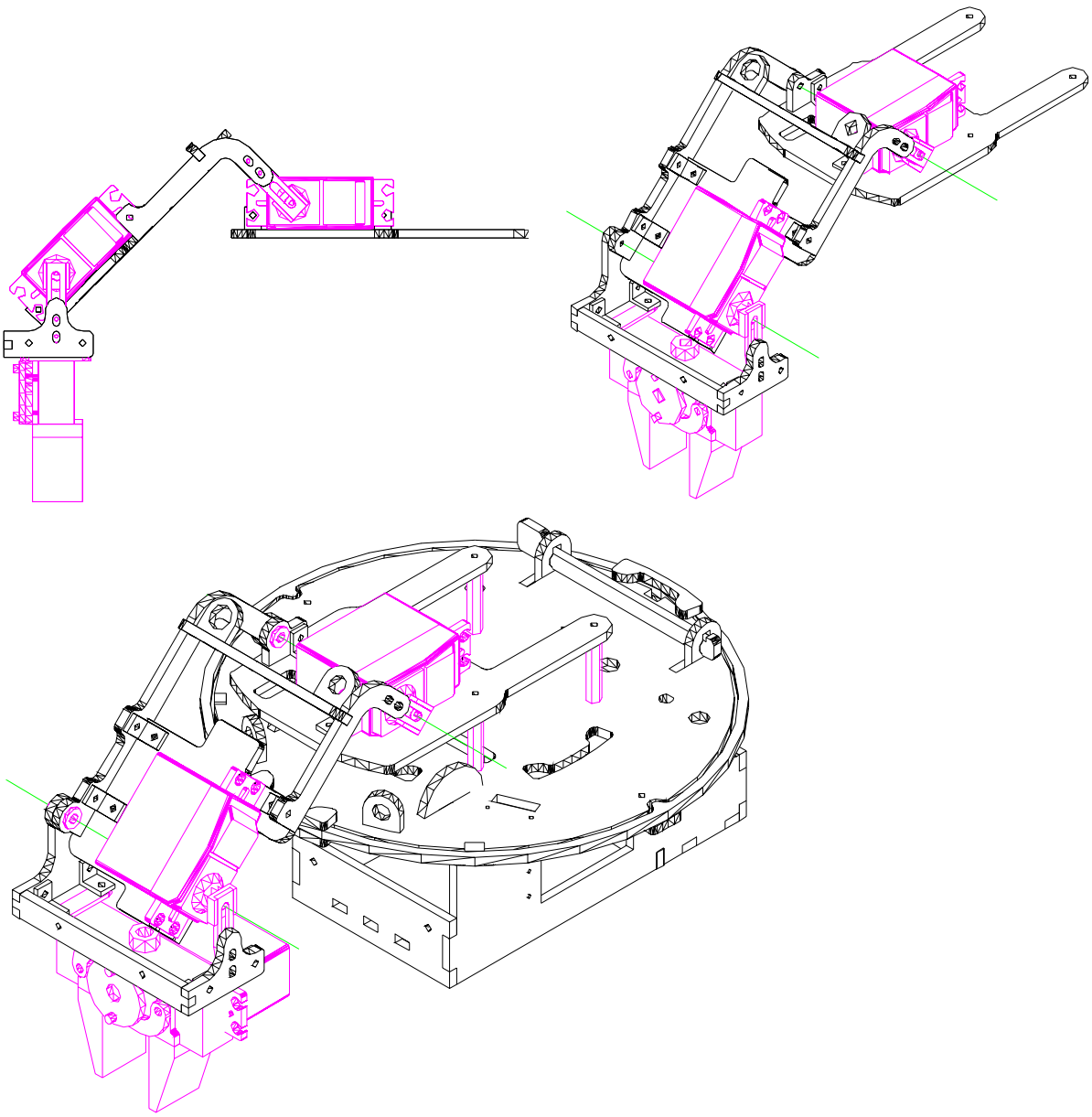


**Figure 35 LOW PICK CONFIGURATION or POSE**



316 NW 17 Street Suite A Gainesville Florida 32603 www.mekatronix.com

**Figure 36 FORK LIFT CONFIGURATION or POSE**