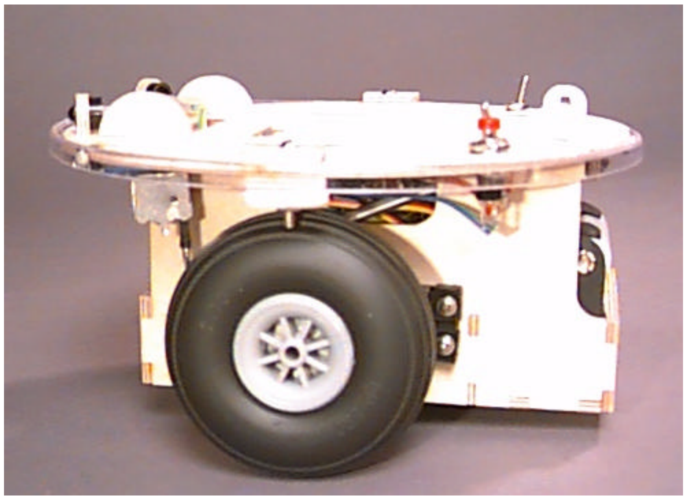
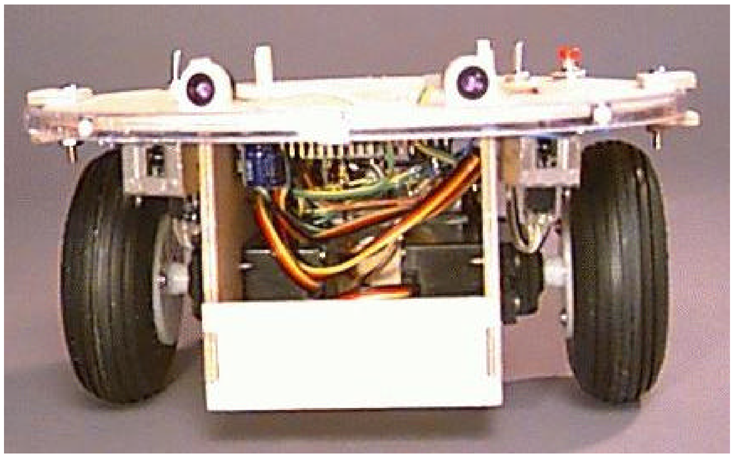
TALRIK JUNIOR USERS MANUAL

by

Keith L. Doty





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, you 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.

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.

TABLE of CONTENTS

LIST of FIGURES

LIST of TABLES

1. SAFETY AND HANDLING OF TALRIK JUNIOR™ (TJ™)

1.1 TJ™'s Static Sensitive Parts

Some of TJ™'s components are static sensitive. These are located on the printed circuit boards. Do not touch these boards without being properly grounded. Static discharge can destroy these parts.

1.2 Caution for Biological Organisms

Most individuals find TJ™, exciting, enjoyable, amusing, and fascinating. TJ™'s small size minimizes its[1] threat to organic life, although pets and small children might find TJ™ either an interesting playmate or an alien to avoid. A child's hands, feet, and mouth probing TJ™ may lead to minor injury to both child and robot, so, to eliminate adverse reactions with pets, children, or nervous adults, be sure TJ™ operates under supervision by someone who knows how to turn it (him!) off. Remember, TJ™ is an autonomous machine and, as such, becomes "out of control" once set in motion. An add-on IR remote control package can be purchase separately to allow you to override TJ™'s autonomous mode.

1.3 Holding, Carrying, and Transporting TJ™

TJ™'s small size and portability make it ideal for safe demonstrations and presentations of machine intelligence algorithms. Porting TJ™ from desk-top platform to floor and back, the most frequent carrying operation, requires handling TJ™ carefully. The recommended carrying technique is to firmly hold TJ™ from the rear with the thumb on top of the key holding the plate down and the battery case floor and tail skid resting on the remaining four fingers. Users at their own risk may devise other carrying techniques.

To transport TJ™ for long distances, place it securely padded into a suitably sized box.[2] Do not expose TJ™ to extreme cold or heat in an automobile during winter and summer for more than a few hours. Although TJ™ has survived a hot automobile for 12 hours in *40° C*, such treatment cannot help but reduce its lifetime.

TJ™ has flown in airline luggage compartments across the Atlantic, ridden in automobiles, passed around in a class of electrical and computer engineering students, and used by middle school students for schools science projects, all without ill effects! The following caution, however, is our official advisement.

Caution: *TJ™ should not be exposed to moisture, or continuous high humidity (>90%), or high temperatures, 40° C (100°F), or low temperatures, 5°C (40°F).*

[1] We want to say "his", since TJ is definitely a male name for us. Some future robots will certainly bear female names. We will resist "his" for now, but the anthropomorphic tendency to use personal pronouns in referring to TJ is hard to resist and we cannot guarantee that personal pronouns usage will not appear!
[2] An accessory TJ™ carrying case will be offered soon.

1.4 Storage

Place TJ™ in a labeled (don't forget where "he" is), enclosed, sealed, padded box in an office or home environment when storing for extended periods of time.

2. TJ™ OVERVIEW

This section provides the user general orientation to TJ™ .

2.1 What Can TJ™ DO?

TJ™ provides an entry-level platform for the development and testing of machine intelligence and autonomous behavior algorithms. The creative and entertainment value of developing your own intelligent, physically embodied autonomous agent provides tremendous motivation and pleasure. As with most high-tech tools (toys!) the implications have broad scope. By attaching different mechanisms to such an autonomous agent, the user can develop autonomous vacuum cleaners, autonomous lawnmowers, autonomous construction vehicles, autonomous butlers or maids, intelligent kinesthetic art, tile laying robots, security robots, maintenance robots, space station robots, planetoid explorers, undersea mining, rescue vehicles, war robots, and so on. As the community of autonomous robot developers increases, the applications coming forth will be legion, many of which will be totally surprising.

The modest beginning represented by TJ™ and other MEKATRONIX robots to follow will lay the groundwork and develop the technical infrastructure for a new mechanical species to serve mankind. The arrival of this new species will tremendously affect social, political, religious, philosophical, scientific, engineering, and mathematical interests. To our children's children, these days will seem primitive indeed.

2.2 Operating Environment for TJ™

TJ™ is designed to operate in an office or home environment with smooth tiled floors. Although TJ™ can handle short pile rugs, he spends more energy overcoming rug friction than tile friction and his operational lifetime on a battery charge reduces considerably. Shag rugs and extremely rough surfaces are inimical to TJ™ and should be avoided.

In general, operating environments comfortable to lightly clothed humans will probably be suitable for TJ™. However, we reiterate the previous caution on environmental constraints.

Caution: *TJ™ should not be exposed to moisture, or continuous high humidity (90%), or high temperatures, 40° C (100°F), or low temperatures, 5°C (40°F).*

2.3 Physical Orientation

TJ™'s wheel axis determines the robot's left-to-right axis. The diameter perpendicular to the wheel axis determines the front-to-back axis. The "TALRIK JUNIOR™" label appears on the front of the robot. Figure 1 presents an elevation view of TJ™.

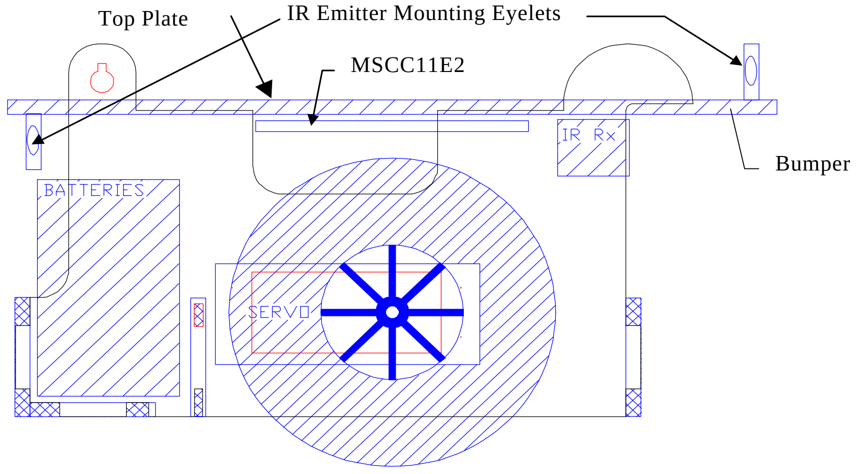


Figure 1 Schematic side view of TALRIK, JR.™

2.4 Sensor Layout

Figure 2 schematizes the standard layout of TJ™'s sensor suite. Table 1 defines the meaning of the labels and a typical application of the sensor. The user is not limited to these applications and can devise and implement other schemes, both in layout and in function. The TJ-ARGOS™ extension expands TJ™'s sensory capability by 9 photoresistors, 2 side-looking IR detectors and emitters, and 1 rear-looking digital IR detector for IR communications or remote control.

Table 1 TJ™'s Minimum Sensor Suite

Label	**Name**	**Function**
IRDLF	Infrared Detector, Left Front	Proximity Sensor
IRDRF	Infrared Detector, Right Front	Proximity Sensor
FBSWL	Front Bumper Switch, Left Front	Front contact Sense
FBSWC	Front Bumper Switch, Center Front	Front contact Sense
FBSWR	Front Bumper Switch, Right Front	Front contact Sense
RBSW	Rear Bumper Switch	Rear contact Sense

2.5 Switches

TJ™ control switches on the top plate near the back provide easy access. The layout of the switches can be determined from Figure 2.

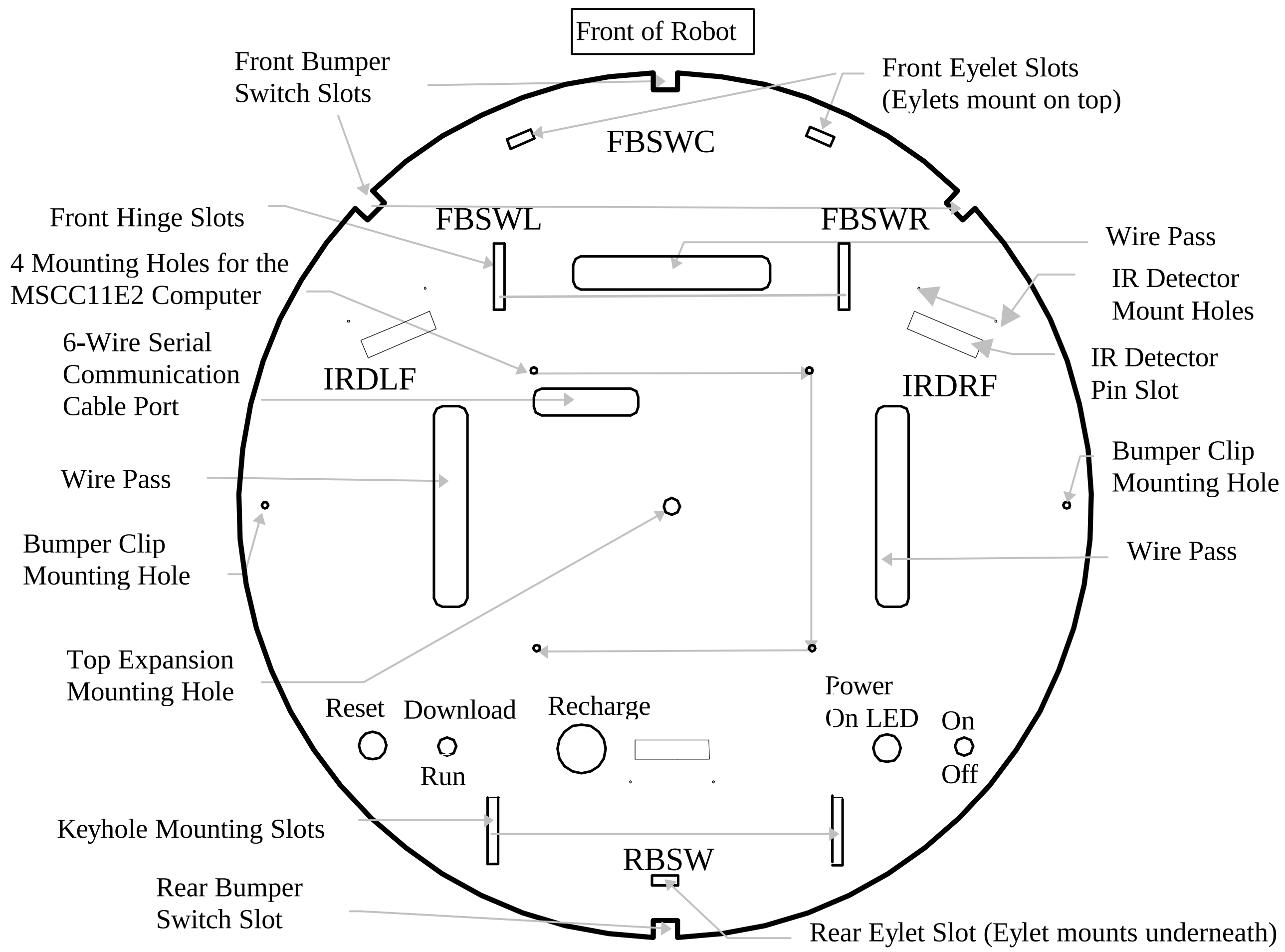

Front of Robot
Front Bumper Switch Slots
Front Eyelet Slots (Eylets mount on top)
FBSWC
Front Hinge Slots
FBSWL
FBSWR
Wire Pass
4 Mounting Holes for the MSCC11E2 Computer
IR Detector Mount Holes
6-Wire Serial Communication Cable Port
IRDLF
IRDRF
IR Detector Pin Slot
Bumper Clip Mounting Hole
Wire Pass
Bumper Clip Mounting Hole
Wire Pass
Top Expansion Mounting Hole
Reset
Download
Run
Recharge
Power On LED
On
Off
Keyhole Mounting Slots
RBSW
Rear Bumper Switch Slot
Rear Eylet Slot (Eylet mounts underneath)

Figure 2 Sensor layout on top of the TJ™ plate. The lead symbol R=Rear, F= Front, IRD=IR detector. The symbol BSW = Bumper Switch, L= Left, C= Center, R= Right. For example, FBSWC = Front Bumper Switch Center, and IRDFL = IR Detector Front Left. The IR emitters and the IR Detectors are typically mounted on opposite sides of the top plate.

The red *RESET* button resets the MC68HC11E2 microcomputer. The *ON/OFF* switch provides power. The visible red led adjacent to the *ON/OFF* switch lights when the switch is moved to the *ON* position.

In *DOWN-LOAD* mode the *DOWN-LOAD/RUN* mode switch permits the user to download programs using PCBUG11. In *RUN* mode, a press of the reset button will invoke the start-up program the user has loaded into TJ™.

2.6 Batteries

TJ™ requires a minimum of six AA Nickel-Cadmium rechargeable batteries. ***Only use nickel-cadmium AA batteries. Alkalines, for example. will produce too much voltage and may damage the electronics.***

The battery pack is located in front of the rear caster on the underneath side of the robot. TJ™ can run autonomously on a smooth tile floor for about 1.5 to 2 hours on a fully charged battery pack. This time depends critically upon how frequently the motors change speed and the running time of the motors. NiCad batteries sustain useful power until just before they become fully discharged. So, if you see TJ™ really slowing down, it is time to recharge!

2.7 Recharger

The charging technique is known as a trickle charge. This means that a small current is constantly applied to the batteries. Because this is a low level current the robot can be left charging indefinitely. While the robot is not moving around plug it into the charger. This will keep the batteries charged. Be sure to use only a Mekatronix approved AC adapter.

TJ™'s charger must be at 12 volts and be able to supply 500 milliamps. Lower voltages will not charge the batteries and higher voltages my damage the electronics. The charger plugs into the receptacle mounted on the left bridge support.

Recommendation: During program development and high usage times, keep TJ™ connected to the charger whenever possible. This will insure fresh batteries during experimentation.

Caution*: If the output DC voltage of the AC adapter exceeds 12 volts, the 1/4 watt charging resistors will act like a fuse and burn out and char a spot on the underside of the top plate. The smoke and smell are unpleasant, so be careful.*

2.8 Installing TJ™'s batteries (IMPORTANT: NiCads only)

1. Turn *ON/OFF* switch to *OFF*.
2. Insert six NiCad AA batteries into the pack, oriented with proper voltage polarity as labeled on the pack. Failure to do so may destroy all the electronics.
3. Turn and slide the plate key out of the rear supports and rotate TJ™'s plate forward, exposing the rear battery compartment.
4. Carefully place the battery pack into the battery compartment. Practically any orientation of the pack will work, except with the battery snap terminals facing down!
5. Snap the 9v snap connector to the battery pack and the female connector on the other end into the male power header on the MSCC11 microcontroller mounted on the underneath side of the top plate.
6. Rotate the top plate down onto the sides, being careful not to pinch any wires between the top plate and the sides.
7. With the top plate down, slide the key into the key slot, turn 90 degrees to lock the key into place.

2.9 Replacing TJ™'s batteries (IMPORTANT: NiCads only)

1. Turn *ON/OFF* switch to *OFF*.

2. Turn and slide the plate key out of the rear supports and rotate TJ™'s plate forward, exposing the rear battery compartment.
3. Unplug the battery connector from the male power header attached to the underside of the plate.
4. Carefully remove the battery pack from the battery compartment.
5. Remove the old batteries and insert six NiCad AA batteries into the pack, oriented with proper voltage polarity as labeled on the pack. Failure to do so may destroy all the electronics.
6. Snap the 9v snap connector to the battery pack (if not already snapped on) and the female connector on the other end into the male power header on the MSCC11 microcontroller mounted on the underneath side of the top plate.
7. Rotate the top plate down onto the sides, being careful not to pinch any wires between the top plate and the sides.
8. With the top plate down, slide the key into the key slot, turn 90 degrees to lock the key into place.

TJ™ will now have power when the *ON/OFF* switch is turned to *ON*.

2.10 TJ™ Bumper

TJ™'s bumper girdles the plate. Objects higher or lower than TJ™ platform will not be seen. Such obstacles do exist in home and office, and can stall the robot when it strikes them. A stalled robot is not a pleasant thing to watch and hear! The wheels struggle to turn and the motors might eventually overheat and burn out if the surface has high friction. Our recommendation is either to 1) TJ™ proof such obstacles or 2) Hang around and get him out of trouble or 3) Write a program that detects motion, perhaps using the underneath photoresistors of the ARGOS extension as vision flow detector. We recommend 3). The idea behind 3) is that if TJ™ does not detect any light pattern change on the floor, he assumes he is stuck and backs up, turns around, and leaves the area.

Behind the bumper, four push button switches surround TJ™'s waist. TJ™ is configured only to distinguishes between front and back collisions. Experts can make hardware and software changes to enable TJ™ to distinguish any subset of button closures activated by the bumper.

Caution: *DO NOT LIFT or CARRY TJ™ by the bumper.*

2.11 Serial Communication

The MB2325 serial communications board shown in

Figure 3 permits the user to download and upload code and data to TJ™ via a 6-wire serial communications link. The 6-wire communications line connects into the MB2325 bidirectional serial communications board at one end (lower-right corner in (Figure 3) and to TJ™'s serial interface at the other (Figure 4). 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 25 to 9 pin plug converter.

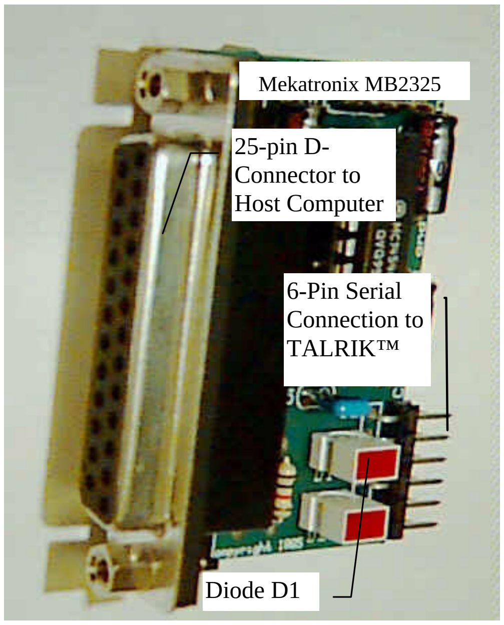

Mekatronix MB2325
25-pin D-Connector to Host Computer
6-Pin Serial Connection to TALRIK™
Diode D1

Figure 3 MB2325 Communications Board

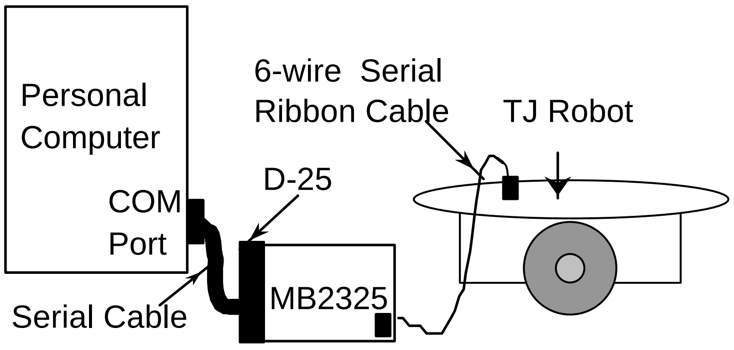


Figure 4. This diagram illustrates the serial connection between a personal computer, the communications board (MB2325) and a Talrik Junior.

2.12 Where is TJ™ During Program Development?

You will probably spend considerable time developing and testing programs to run on TJ™. During the initial phases of this process the user will undoubtedly make many changes and run many variations of the programs. This procedure can be expedited by mounting TJ™ on a test platform that elevates the drive wheels off the floor or desktop so that the wheels turn freely without moving the robot. To maintain fresh batteries during these long development sessions, keep TJ™ connected to the charger. Also, keep the serial communications cable to TJ™

connected to minimize plugging and unplugging it. When the user is ready to test TJ™ on the floor, disconnect both the charger and the serial communications cable from TJ™.

2.13 Halting a Moving TJ™

Chasing down a moving TJ™ takes more skill than transporting it. A recommended procedure is to chase it from behind and reach down to turn the *ON/OFF* switch *OFF*. You can also program the robot to stop on certain conditions that can be artificially generated. For example, you can program the robot so that a touch of the rear bumper while the robot is going forward will stop it. With the IR remote control unit accessory unit, TJ™ can be stopped by the IR remote.

3. TJ™ SET-UP

You may find that the TJ™ Assembly Manual provides useful nomenclature and diagrams depicting the various components and structures and you may need to refer to it from time to time.

3.1 Getting TJ™ Ready

This section tells you how to unpack and prepare TJ™ for operation. If you bought a TJ™ kit you will have already read the TJ™ Assembly Manual and put it together and you can skip the next section.

3.2 Unpacking an Assembled TJ™

1. Carefully unwrap TJ™ from the bubble wrap being careful not to catch wires or drop the robot.
2. Visually inspect the robot for any obvious damage.
3. Make sure that no cables have become disconnected from the robot during shipment. If cables have disconnected, it is usually obvious how to reconnect them as they were.
4. In the case of the IR detectors, the black side of the cable should line up with the edge of the case. Check to insure the IR connectors are fully plugged into the IR emitters ("eyes" on top of the robot plate). The IR emitters are diodes, so if the plugs are not oriented properly, the IR emitters will not emit. Refer to the TJ™ Assembly Manual if in doubt about the connections.
5. Check the bumper to see if it is pushing up against any switches. If it is, just gently shift the bumper away from the switch. You may have to loosen and retighten the bumper clamps at each side to adjust the bumper. Latter program testing will reveal any specific problems.
6. Remove the Software disks and any other documentation.

7. You may have also purchased the following (recommended)
    - A wall plug adapter for recharging the robot ,
    - An MB2325 serial communications board and cable,
    - 6 Nickel Cadmium rechargeable batteries.

3.3 Computer Requirements

Program development for the TJ™ can be performed on a low cost DOS system. Nothing more is required. Users with more expensive, sophisticated computer systems, of course, can run DOS under the *Windows95* environment.

3.3.1 Entry Level System (DOS)

The following computer system will meet all the basic hardware support for developing TJ™ programs and applications.

1. 286-PC or later processor running under *DOS* with several Megs of available hard disk space and 1 Meg of RAM.
2. An available serial port. We will assume that COM1 is used. See the next section about COM port problems to avoid.
3. 1.44 Meg floppy

*3.3.2 A more expensive system (*Windows95*)*

The following high-end hardware offers a more sophisticated, but non-essential, working environment.

1. A high speed (100MHz or better) computer running *Windows 95*
2. Some modems work and some do not with PCBUG11. You may need to disconnect your modem (see the section titled *COM port problems to avoid*, page 15).
3. A mouse on COM2
4. A printer
5. Access to the *Internet*, especially the *World Wide Web*,
6. *Windows* terminal (*terminal.exe*)

3.4 COM port problems to avoid

Motorola's freeware *Pcbug11* uses the serial port in a manner that does not permit sharing the serial interrupt. Therefore, the port that communicates with the robot should not share the same interrupt with any other system resource. For example, COM1 and COM3 share the same interrupt in DOS, as do COM2 and COM4. So, if the robot is connected to COM1, nothing else should be connected to COM1 or COM3 or use their interrupts. We have seen some extreme instances when the internal modem of a computer had to be removed to get *IC* or *Pcbug11* running, because the modem used the same interrupt as the robot COM port and disrupted the robot serial communication.

3.5 Serial Communication between Host Computer and TJ™

This section assumes TJ™ has fully charged batteries and the processor functions. Without either of these prerequisites, the user cannot begin.

The MB2325 serial communications board generates the proper voltage conversions that permit the user to download and upload code and data to TJ™ via a 6-wire serial communications link. Whenever testing programs on TJ™, the user will connect the host computer with TJ™ via this interface. Rather than duplicate this interface circuit on each robot, Mekatronix™ chose to make a single external interface for the host computer. This approach has a twofold advantage, 1) removal of this circuitry from the robot platform saves battery energy and reduces printed circuit board cost, 2) only one MB2325 board is required per computer instead of one per robot. In multiple robot systems, or swarm systems, this approach yields an increasing cost advantage.

3.6 Plugging MB2325 into the host computer

The MB2325 serial communications board (

Figure 3, page 13) has a 25-pin, female, type D-connector. On most computers COM1 is a 25-pin male D-connector. If your computer has a 9-pin connector you will need a 9- to 25-pin converter (available at most computer and electronics stores). When the MB2325 is plugged into the COM1 port of the host computer D1 will light. This light lets you know that the serial port on the host computer is working. If D1 does not light, something is wrong with the serial port of the host computer and the robot will not be able to communicate with it.

3.7 Connecting the Host computer to TJ™

After the MB2325 is connected to the computer, take the 6-pin rainbow colored cable and insert either end onto the only 6-pin male header (J2) on the MB2325. On the back of the robot is a 6-pin rainbow cable with a male connector. Connect this male header to the female connector on the other end of the cable coming from the computer. This end is also a keyed connector. Now, place the *DOWN-LOAD / RUN* switch to the *DOWN-LOAD* position. Turn the *ON/OFF* switch to *ON*. At this point the red LED on the robot's top board should light. Press and momentarily hold *RESET* (the red button) on the robot. With *RESET* pressed, D2 on the MB2325 should light. If D2 does not come on, reverse the 6-wire connector at one end only and try again. D2 should now light with *RESET* pressed. When *RESET* is released, D2 on the MB2325 turns off.

3.7.1 Verify proper operation

The lights on the MB2325 let you know that the basic connections are working. If the POWER-ON light does not light when the power switch is turned to *ON*, the robot may not be charged. If the batteries are charged, check battery and power connections to the circuit boards. Particularly, check underneath the robot to verify the battery cable has not come undone from the power connector. If the lights on the MB2325 board do not function as specified, check that no other programs or devices are using the serial port.

3.8 Software Language Support

TJ™ programs can be developed with Image Craft C (ICC11), the compiled version of C for the MC68HC11 or with Motorola MC68HC11 Assembly Language. Image Craft also has a real-time executive system REXIS. Mekatronix™ sells both ICC11 and REXIS. Motorola MC68HC11 Assembly Language is freeware and is supplied at no charge.

ICC11 augmented with the appropriate Mekatronix library files is sold by authorized Mekatronix distributors (www.mekatronix.com/distributors) and is the preferred programming language for extensive program development on TJ™ and is the language of choice for other Mekatronix™ robots.

4. SERIAL COMMUNICATIN WITH YOUR TJ™ ROBOT

Serial communication between your PC and the robot may be established several ways, through

1. Serial Cables, a
2. 900 MHZ radio communication link, or
3. IR communication link.

Through the serial communication link between your robot and personal computer you can
1. Download executable ".s19" object files from your PC to the robot,
2. Download data and commands from the PC to the robot, and
3. Upload data from the robot to the PC.

4.1 Serial Cable Setup

The radio and IR serial links are options. Refer to their respective manuals. The standard option, serial communication between robot and PC through cables, is described here.

1. On Windows95 open an MSDOS window. If you are using the ICC11 DOS compiler, then execute `seticctj.bat`. Assuming a DOS prompt, set the current directory by typing

```
cd <path to ICCTJ>
```

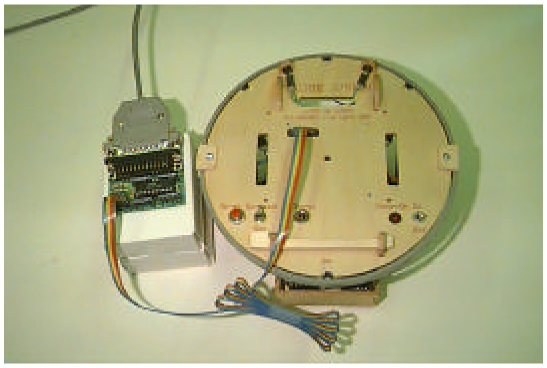


Figure 5 This Photograph shows the gray serial cable from a PC COM port mating with the D-25 connector on the communications board (com-board = MB2325 = the exposed circuit board sitting on the white boxes). The multicolored 6-wire serial cable attaches to the male header on the com-board and into the serial slot on the TJ™ plate. Note the same color orientation of both ends of the 6-wire cable for the configuration pictured.

2. Connect one end of a C2325, a 6-wire serial communications cable, to the Mekatronix MB2325 serial communications board. There is only one 6-pin male header on the MB2325, just below the two LEDs. Connect the other end of this cable to the serial port on the MSCC11 through the oblong slot on the TJ™ plate underneath the lettering MEKATRONIX

(Figure 5). The header seen in this serial port slot is the J54 male header on the MSCC11. Corresponding pins of the 6-pin headers on the MB2325 and the TJ™ plate, match from left-to-right as oriented in the diagram. Thus, the leftmost pin of one end connects by wire to the leftmost pin of at the other end.

1. The MB2325's 25-pin female D-connector can connect directly to your Personal Computer COM port or via a serial cable. Make such a connection. This setup establishes a link from TJ™ to your PC. Check to make sure diode D1 lights when you connect the MB2325 to your PC via the serial cable or directly into its serial port. The advantage of using a serial cable between the MB2325 and COM port now becomes obvious, you can easily see the LED status lights on the MB2325. Refer to Figure 2 for switch and connector locations.

   Important: Do not connect the serial cable to a COM port that is already being used.

2. Flip the DOWNLOAD/RUN switch to DOWNLOAD.

3. Turn on TJ™ power with the ON/OFF switch flipped to ON. Make sure the red LED Power-On light shines. If not check batteries and battery connections before going further.

4. Press the red push-button RESET switch. LED D2 lights when you hold RESET down. D2 is the LED closest to the edge of the Mb2325 com-board. If it does not light, reverse the C2325 6-pin connector to the MB2325 and try again.

5. With the physical serial communication link in place, download a program using either the IDE Terminal program, or PCBUG11. Refer to Sections and for download procedures.

Alert!

PCBUG11 is archaic freeware and will not execute on many new computers. Mekatronix does not support PCBUG11. If you choose to use PCBUG11 and cannot get it to work we highly recommend purchase of ICC11 IDE from Mekatronix.

You now have a successful serial communication link between your robot and personal computer.

5. IF YOU DO NOT HAVE A C COMPILER

If you did not purchase ICC11 from an authorized Mekatronix distributor (www.mekatronix.com/distributors) and you do not have your own C compiler for the MC68HC11 family of computers, you can program in Assembly language. You can assemble your programs using the Motorola freeware Assembler located in the directory `ASSEMBLER` on the TJ™ software distribution disk. The Motorola Assembler will generate S19 files from your assembly language programs. S19 files are ASCII files that the Motorola boot loader can read and convert to MC68HC11 machine code. The download process for S19 files is described in Section 4 of this manual.

Assembly language programming is beyond the scope of this manual. A good, readable beginning textbook for learning MC68HC11 assembly language programming is:
Microcomputer Engineering, Gene H. Miller, Prentice Hall, 1993, ISBN 0-13-584475-4

6. INSTALLATION OF ICC11 AND TJ™ SOFTWARE

If you purchased the ICC11 software this section provides guidance in its installation and integration with the TJ™ software. Be sure to refer to your ICC11 manual for installation particulars for the ICC11 system. The IDE environment makes the WINDOWS version a sweet tool for programming your Mekatronix robots and makes life much easier than the DOS version. Mekatronix highly recommends the WINDOWS version. Experience has shown it is well worth the additional cost.

Note: In typed commands to your computer, angle brackets indicate parameters for which you must substitute the appropriate information. Do not actually type the angle brackets. For example, `<enter>` *means "Press the Enter key on the keyboard ;* `<filename>` *means "Type the filename alphanumeric key sequence on the keyboard and the path to it, if necessary."*

Summary of Installation Process

1. Insert ICC11 diskette and install ICC11, or download from an authorized Mekatronix distributor (www.mekatronix.com/distributors) web site.
2. Remove ICC11 diskette, insert the most recent version of the TJ Distribution Software diskette and install, or download from an authorized Mekatronix web site and install. Installation essentially means copying files into appropriate directories in `c:\icctj`.

7. INSTALLATION OF ICC11 FOR WINDOWS

Insert ICC11 diskette and install ICC11, or download 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:\icctj` as your root directory. Let the installation update your DOS Config file. Reboot. Put the ICC11 icon shortcut on your desktop.

After installation you should observe the following file directory structure

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

7.1 IDE Compiler and Linker Setup for TJ

Once you setup the IDE environment, you will not have to change it from invocation to invocation, since it remembers its most recent state.

1. Double click on ICC11 icon to enter the Integrated Development Environment (IDE).

2. Under OPTIONS in the menu bar select COMPILE.
3. In COMPILER select the Linker tab.
4. In the Linker setup window enter the following:

a. Text section:	0xf800
b. Stack:	0x00ff
c. Data Section:	0x0000
d. Additional Libraries:	libtj

(You may wish to use the Setup Wizard to save this configuration under the name TJ.)

If you specified `c:\icctj` as your root during installation, then

e. Library Path: `c:\icctj\Lib`

will be the default.

5. Select Preprocessor under COMPILER:

If you specified `c:\icctj` as your root during installation, then

Include Paths: `c:\icctj\include`

will be the default.

6. Close COMPILER window.

Continue immediately to the next section.

7.2 IDE Setup for Downloading into a Robot

1. Select TARGET on main menu bar.
2. Select Bootstrap Download Mode.
3. Select TARGET on main menu again.

   Verify that Bootstrap Download Mode has a check mark by it and then
4. Select Terminal.

   A window opens up. Expand it to fill your screen. At the bottom you will see several selection buttons. You will address those next.
5. Select Bootstrap Options. The HC11 window opens.
6. In Bootloader Programming select "Internal EEPROM".
7. Set HW echo mode to "normal".

8. Set baud rate to default (1200)
9. Close HC11 window.

You are now ready to edit, compile and download programs to your robot from the IDE.

7.3 Integrating TJ™ Software with ICC11 for Windows

Integrating TJ™ software amounts to copying files into `icctj` directory at the appropriate places. Refer to Section 11.3 and Figure 7.

8. COMPILE AND EDIT ON IDE

1. Select File in the menu bar and click on Open.
   Browse to select the source code file of interest, for example, c:\icctj\tjcode\avoidtj.c, and open it.

A window opens which allows you to edit the program. In the example case, the program needs no editing, but normally you would edit your program at this time. Refer to the ICC manual for details on editor commands.

2. Use Window on the menu bar to tile the Status and Editor windows (ALT-T).

3. Select Compile in menu bar and then select Compile to Executable.

   The compiler compiles the program in the currently active edit window. If no edit window is open, no compilation is possible.

4. Watch the Status window. A successful compile ends with the word SUCCEEDED, otherwise note errors and correct your program in the edit window and repeat Steps 3 and 4.

9. DOWNLOADING USING IDE

You must carry out the setup in Sections 7.1 and 7.2 before downloading any files into the robot. Your setup will be in the state that you left it in at the end of your previous IDE session.

Connect your PC, serial cable, MB2325 communications board and the six wire Mekatronix serial cable C2325 to the robot as described in Section 4.

Open up IDE by double clicking on your ICC11 icon.

1. Select *Target* in the menu bar.
2. If *Bootstrap Download Mode* is checked, select *Terminal* and skip Step 3. Otherwise select *Bootstrap Download Mode* and do Step 3.

3. Select *Target* again and click *Terminal*.

You now have the terminal window open.

4. If necessary, select *Browse* to find the .s19 file of interest. Select the desired .s19 file.

5. Flip the robot's Download/Run switch to Download.

6. Press the robot's red Reset button.

   Make sure diode D2 on the MB2325 board lights when reset is held down and goes off when reset is released.

7. Select *Bootstrap Download* in the IDE *Terminal* window.

8. Select OK on the IDE message.

9. The .s19 file loads into the robot.

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

10. EXECUTION OF ROBOT CODE ON IDE

The IDE terminal simulator (*Terminal*) accepts robot serial output directly! This allows you to monitor robot IO while the robot remains connected to the PC, a very convenient feature of IDE.

Execute Program Procedure

1. Download .s19 file into robot from *Terminal* window (refer to Section 9).
2. Flip robot Download/Run switch to Run.
3. Press robot red Reset button.

CAREFUL PROGRAM WILL START IMMEDIATELY!

Your program will start to run. If your program transmits serial output, it will appear on the terminal simulator of IDE that you currently have open. If your robot produces IO to the *Terminal* screen, you may have to press reset several times or hold it down for a second to get clean communication.

11. INSTALL ICC11 FOR DOS

Insert ICC11 diskette and install ICC11, or download 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:\icctj` as your root directory. Let the installation update your DOS Config file. Reboot. There is no IDE for DOS.

After installation you should observe the following file directory structure

- `C:\icctj`

- `Bin`
- `Examples`
- `Include`
- `Lib`
- `Libsrc`

11.1 Setup ICC11 DOS with seticctj.bat

Always execute `seticctj.bat` in C:\icctj after opening a DOS window. Do not execute it more than once in an opened DOS window:

```
C:\ICCTJ>icctj<enter>
```

The DOS batch file `seticctj.bat` is listed below:

```
PATH %PATH%;C:\ICCTJ\bin
set ICC11_INCLUDE=C:\ICCTJ\include
set ICC11_LIB=C:\ICCTJ\lib
set ICC11_LINKER_OPTS=-btext:0xf800 –bdata:0x0000 –ltj -dinit_sp:0x00FF -dheap_size:0
```

Refer to the ICC11 Manual for explanations about the linker options, the last line.

11.2 Generate TJ Library Object Files

1. This step should not be necessary and should only be done if you truly understand what's taking place.

   If you wish to modify `libtj.a`, after execution of `seticctj.bat`, change the directory to `Libsrc` and execute `libtjmake.` If you have the TJ library source code (separate purchase of TJ Education software package) the batch file will actually recompile before constructing the library file, otherwise it will only collect the object files already there into the library,

   ```
   C:\ICCTJ>cd Libsrc
   C:\ICCTJ\Libsrc>libtjmake <enter>
   ```

   This will make the TJ™ library archive `libtj.a` that holds object versions of the standard TJ™ support C files. This library `Libtj.a` must be stored in the ICCTJ `Lib` directory. The paths assume that you installed the system in `c:\icctj`. If you placed it differently you will have to change the paths in the `PATH` command.

   Note: Unless you change the TJ library files, you so not need to execute `libtjmake`. Ask your authorized Mekatronix distributor (www.mekatronix.com/distributors) *about obtaining the TJ source code library files.*

   A listing of `libtjmake` appears in Figure 6.

   NOTE: the source code for
   ```
   icc11 -c vectors.c, serial.c, motjdr.c, analog.c, irtj.c
   ```
   does not come with the TJ distribution software. It is available in the TJ Education package.

After installing ICC11 and the TJ Distribution Software, the user can write C programs, compile them and download their S19 file output to TJ™ to test, debug, and run. A convenient method for initial testing of programs on TJ™ is to place TJ™, with charge plug engaged, next to the host computer on an elevated platform with the drive wheels suspended in the air so they do not

```
echo off
echo This script file makes a library file for the TJ Routines
echo Title         Make TJ Libraries
echo Programmer    Keith L. Doty
echo Date          Feb. 7, 1998

icc11 -c vectors.c
icc11 -c serial.c
icc11 -c motjdr.c
icc11 -c analog.c
icc11 -c irtj.c
del ..\lib\libtj.a
ilib -a libtj.a  serial.o
ilib -a libtj.a  motjdr.o
ilib -a libtj.a  analog.o
ilib -a libtj.a  vectors.o
ilib -a libtj.a  irtj.o
ilib -t libtj.a
copy libtj.a ..\lib\*.*
```

Figure 6 The `libtjmake` which generates `libtj.a` library object file for the TJ™ robot.

contact any physical surface. After you develop confidence in your program's operation, you will typically make a closely monitored trial run on the floor. Remember that TJ™ is autonomous and the behavior you expect may not arise!

11.3 Integrate TJ™ Software with ICC11 DOS or WINDOWS Versions

Insert the TJ Distribution Software Diskette into your floppy drive or open up the directory into which you downloaded it from the web. In DOS or Windows95 execute

`installtj [<path to icctj file>]<enter>`

If no parameter is entered for `<path to icctj file>` the default is `c:\icctj`. The

```
@echo off

if "%1"=="" goto noargument

copy seticctj.bat %1
copy TJReadme.txt %1
copy TJProg.txt %1
copy IDEsetup.txt %1

copy bintj\*.* %1\bin\*.*
copy includetj\*.* %1\include\*.*
copy libtj\*.* %1\lib\*.*

if not exist %1\libsrctj md %1\libsrctj
copy libsrctj\*.* %1\libsrctj\*.*

if not exist %1\tjcode md %1\tjcode
copy tjcode\*.* %1\tjcode\*.*

md %1\pcbug11
copy pcbug11\*.* %1\pcbug11\*.*

md %1\assembler
copy assembler\*.* %1\assembler\*.*
```

```
: noargument

copy seticctj.bat c:\icctj
copy TJReadme.txt c:\icctj
copy TJProg.txt c:\icctj
copy IDEsetup.txt c:\icctj

copy bintj\*.* c:\icctj\bin\*.*
copy includetj\*.* c:\icctj\include\*.*
copy libtj\*.* c:\icctj\lib\*.*

if not exist c:\icctj\libsrctj
   md c:\icctj\libsrctj
copy libsrctj\*.* c:\icctj\libsrctj\*.*

if not exist c:\icctj\tjcode md c:\icctj\tjcode
copy tjcode\*.* c:\icctj\tjcode\*.*

md c:\icctj\pcbug11
copy pcbug11\*.* c:\icctj\pcbug11\*.*

md c:\icctj\assembler
copy assembler\*.* c:\icctj\assembler\*.*

: end
```

Figure 7 Listing of the batch file installtj.bat that integrates TJ software into the appropriate ICCTJ directories

Further, you will notice three new text files, `TJReadme.txt`, `TJProg.txt` and `IDEsetup.txt` along with the batch file that sets up ICC11 DOS for the TJ, namely, set`icctj.bat`.

12. COMPILE A TJ™ PROGRAM IN DOS

Operational Procedure During Program Development

Keep TJ™ jacked up so his wheels suspend above the desktop. While TJ™ is on the desk, keep the charge jack plug in for continual recharging to enable hours of uninterrupted coding pleasure!

6. On Windows95 open an MSDOS window or boot up in a DOS only machine. Assuming a DOS prompt, set the current directory by typing

   `cd <path to icctj>\icctj`

   Fill the path parameter `<path to icctj>\` with the path to `icctj`. If you placed the ICCTJ compiler files in the root, then the default path is `C:\ICCTJ.` The `tjcode` directory is where we suggest you put your TJ™ programs. Future Mekatronix enhancements will assume code is placed in `tjcode`.

   Execute `seticctj.bat`.

7. The computer is now in the ICCTJ directory and the DOS prompt is `C:\ICCTJ>`.
   Type: *`comptj <filename> <enter>`*
   to compile a C source file in the `tjcode` directory. The path to `<filename>` defaults to the `tjcode` directory, if no path is given for the filename. Do not put the *`.c`* extension on `<filename>.` The path to `<filename>` can also be explicitly specified or established previously by `PATH.` The batch file `comptj` (compile TJ™) is listed in Figure 8.

```
echo off
echo * This script assumes
echo *   1) ICC11 is root,
echo *   2) You are executing this command from the root, and
echo *   3)The code to be compiled is in TJCODE, a subdirectory of ICC11.
echo *
echo * Do not put the .c suffix on the file to be compiled.
pause

cd tjcode
icc11 –e –v –l -m %1.c
cd ..\
```

Figure 8 The `comptj.bat` file for compiling TJ™ code. Note that the .c extension must be omitted from the source code file name when using this command.

12.1 Example DOS Compilation of a TALRIK JUNIOR C Program

Assume the current directory is `icctj` and that `seticctj.bat` was executed after the DOS window was opened and the directory changed to `icctj`.

Type: *`comptj avoidtj <enter>`*

to compile the sample program `avoidtj.c` found in the `tjcode` directory.

13. PCBUG11 DOWNLOAD OF A PROGRAM INTO TJ™

The following method for downloading is not supported by Mekatronix and is provided only as a convenience for our customers. Because of the extreme age of the PCBUG11 freeware, the downloading technique described here may or may not work. Many subtle errors and lockups arise between PCBUG11 and a WINDOWS environment. For this reason, Mekatronix makes no claim about the usefulness of this procedure. If you have troubles with this method, Mekatronix strongly urges you to purchase the ICC11 WINDOWS version from a Mekatronix distributor.

If you boot your system in a strictly DOS environment, you will have a higher probability of success with the download procedure described below. With these cautions in mind, at your own risk you can try the procedure while operating in an MSDOS WINDOW in WIN95 environment. On some PCs you will be successful. Just note, however, that PCBUG11 appears to permanently lockup the COM port, even after you close it, and you might have to reboot your entire system to get the COM port back for other programs!

Download Procedure

1. Open a DOS WINDOW or boot up in DOS and change directory to `c:\icctj`

2. Each time you create a new DOS WINDOW in WIN95 or just after boot you must execute `seticctj.bat` only once.

3. Establish the physical serial communications link (Refer to Section 4).

4. Flip the DOWNLOAD/RUN switch to DOWNLOAD.

5. Turn on TJ™ power with the ON/OFF switch flipped to ON. Make sure the red LED Power-On light shines. If not check batteries and battery connections before going further.

6. Press the red push-button RESET switch. LED D2 lights when you hold RESET down. D2 is the LED closest to the edge of the Mb2325 com-board. If it does not light, reverse the C2325 6-pin connector to the MB2325 and try again.

7. Use the batch file `loadtj` to download a Motorola S19 code file. This batch file invokes PCBUG11. See the format for `loadtj` below:

```
loadtj<sp><filename><sp><COM_PORT_NUMBER><enter>
```

In the batch command format, `<sp>` means space. The current directory must be ICCTJ. For example,

Type: *`loadtj avoidtj 1<enter>`*

If the TJ™ serial cable is connected to COM1, then execution of `loadtj.bat` will download the Motorola S19 file `avoidtj.s19` from the `tjcode` directory via COM1 to the TJ™'s EEROM. The serial communications will be via COM1 on your PC. Therefore, the cable from the MB2325 board should be connected to COM1 on your PC. The default is COM1 if you do not specify the `COM_PORT_NUMBER`. If you use COMn, replace *1* by *n* in the above command.

14. COMPILE AND LOAD COMMAND UNDER DOS

The batch file `cltj.bat` has the same command format as `loadtj.bat` and simply combines `comptj.bat` and `loadtj.bat` in a single convenient command. DOS must be in the ICCTJ directory for recognition of this command.

```
cltj<sp><filename><sp><COM_PORT_NUMBER><enter>
```

15. EXECUTE A LOADED TJ™ PROGRAM IN DOS

This procedure assumes you have already downloaded an ".s19" file into TJ™ without error.

Do not do this procedure on a Table, unless TJ™ 's wheels are suspended and not touching any surface. Otherwise, TJ™ will start up and roll off!

1. Turn TJ™ power switch to *ON*, if it is not there already.

2. Flip *DOWNLOAD/RUN* Switch to *RUN*

3. Press RESET.

4. Watch TJ™ GO!

To stop TJ™, run him down and flip the *ON/OFF* switch to *OFF*. Turn him on again to set him in motion once again. If you wish TJ™ to not execute his program when you turn power on, just flip the *DOWNLOAD/RUN* switch to *DOWNLOAD* before turning on power.

16. TALRIK JUNIOR™ SOFTWARE REFERENCE

This section is designed as a quick reference to your TALRIK JUNIOR™ (TJ™) robot distribution software that integrates with ICC11. This section assumes that you have a working knowledge of ANSI C, and will guide you through the drivers and subroutines provided by Mekatronix, which allow you to write TJ™ software ranging from the simplest of programs to advanced algorithms.

This document contains the information you need to know to program a standard TJ™ robot. Other TJ™ modules, such as the TJ-PRO™ and the TJ-ARGOS™, utilize extra libraries and modules. In order to program these robots, refer to the appropriate documents.

16.1 Overview of the TJ™ Software Library

The TJ™ software library is made up of a number of C source files, include files, and macro and symbol definitions. Almost any TJ™ program you write will require the following include files and C files.

Include

- `hc11.h`	Symbol definitions specific to the HC11 (I/O ports, registers, etc.).
- `mil.h`	Macro definitions for simplified register manipulation.
- `analog.h`	Header file for the analog function `analog.c`.
- `motjdr.h`	Header file for the wheel motor control function `motjdr.c`.
- `irtj.h`	Header file for TJ™'s infrared emitter control and IR detector sense `irtj.c`.
- `vectors.h`	Header file for interrupt vector specification `vectors.c`.

C Files

- `analog.c`	Routines for initializing and reading the analog ports.
- `motjdr.c`	Drivers and routines for controlling TJ™'s servo motors.
- `irtj.c`	Drivers and routines for utilizing TJ™'s infrared emitters and detectors.
- `vectors.c`	Memory mapping of interrupt vectors.

16.2 Macro / Symbol Definitions

The files `hc11.h` and `mil.h` contain symbol and macro definitions that facilitate accessing and manipulating the HC11's memory-mapped registers. All register names are consistent with the names given in Motorola's *M68HC11 Reference Manual* (M68HC11RM/AD). For instance, to write to PORTB, you do not need to specify its memory address, instead you use the reference symbol PORTB. These two files assume that your chip's register base is at 0x1000. If you are not sure what this means, don't worry about it; if you've changed your register base for some reason, you'll need to modify the constant _IO_BASE in `hc11.h`.

Three macros that perform MC68HC11 register operations in ICC11 are defined in `mil.h`:

CLEAR_BIT(REG_NAME, MASK)
Will set to zero all the bits in REG_NAME which correspond to ones in MASK. Mathematically equivalent to: (REG_NAME *AND* (*NOT*(MASK))).

SET_BIT(REG_NAME, MASK)
Will set to one all the bits in REG_NAME which correspond to ones in MAKS. Equivalent to: (REG_NAME *OR* MASK)

CLEAR_FLAG(REG_NAME, MASK)
Will write a one to the bits in REG_NAME which correspond to ones in MASK. Generally, this is only used when clearing interrupt flags.

16.3 Analog Port Routines

To make use of TJ™'s analog ports, you must include the file `analog.h` in your program. The following routines are available for controlling the analog ports. The C code for these routines are in `analog.c`:

`init_analog()`
Turns on the analog to digital converter. This function must be called once in your program before using any of the analog ports.

`analog(int port)`
Returns the value at the analog port corresponding to *port*.
Example: To get the value at analog port 3 into the variable "a", write: a = analog(3);

16.4 Motor Routines

To use the wheel motors, include `motjdr.h` in your program. The corresponding C program is `motjdr.c`.

`init_motortj(void)`

This function initializes the motor parameters and must be called before the motors can be controlled by the **motortj** function described next.

`motortj(int motor_name, int percent_of_full_speed)`

Will set one of the wheel motors to a desired speed. To select which motor you want to move, two symbols have been defined for motor_name: LEFT_MOTOR and RIGHT_MOTOR. The speed is given as a percentage of the motor's full speed.

Example 1: To move the right motor at half speed (50%) forward:

```
motortj(RIGHT_MOTOR, 50);
```

Example 2: To move the left motor at full back speed:

```
motortj(LEFT_MOTOR, -100);
```

NOTE: The servo motors are non-linear, so the actual motor speed is different from the percentage specified.

`head(int angle)`

Will turn the head servo (TJ ARGOS™ package only) to the angle specified. This angle ranges between –90 and 90 degrees. An angle of 90 will point the head to the right, -90 will point to the left, and zero points forward.

16.5 Bumper Input

The front bumper switches input to PORTC pin-5, PC5, and the back bumper switch inputs to PORTC pin-0, PC0. Both of these port pins must be configured as inputs in your program. The C language command DDRC = 0x00 will make all PORTC pins inputs. With no switch closure, the bumper port inputs are logic zero. With a switch closure the bumper port inputs are logic one. To determine bumps your program must continually check these port pins.

Caution: Switches bounce, so your program could miss a closure or sample the switch port pins to frequently and record multiple closures when only one bump occurred.

16.6 Infrared Control Routines

The include file `irtj.h` contains the following infrared (IR) routines:

`init_ir()`

Initializes the infrared emitters and detectors. Must be called once in your program before trying to read the infrared detectors.

`ir_mode(int mode)`
Tells the infrared drivers whether TJ™ should fire its IR emitters or not. Calling this function with mode = 0 will cause the robot to fire its IR emitters and look for their reflection. If mode = 1, TJ™ will not fire its own emitters, but rather will look for infrared light coming from some other source (other robots, a beacon, etc.).

`ir_value(int num)`
Returns the last reading from one of the infrared detectors. Like the servo routine, two symbols have been defined to ease the use of *ir_value():* RIGHT_EYE and LEFT_EYE.

Example: To get the level of infrared light seen by the left detector and store this value in variable "lirdet": `lirdet = ir_value(LEFT_EYE);`

`discharge()`
Calling this routine will cause the infrared detectors to return zeroes and the emitters not to fire until the detectors have fully discharged. Once they are discharged, normal IR operation is resumed.

16.7 Interrupt Drivers

Three files play an important role in interrupt driver development, namely, two header files *`vectors.h`* and *`isrdec.h`* and *`vectors.c`*, corresponding to the first header file. The preprocessor statements

```
#include <isrdec.h>
#include <vectors.h>
```

must appear in all TJ™ programs that use interrupts. The code `vectors.c` defines all the interrupt service routines, whether used or not.

Code file *`isrdec.h`* conditionally compiles to null functions the unused interrupt service routines declared external in *`vectors.c`*. The file *`vectors.h`* should not be modified. If you plan to write an interrupt driver for an interrupt which is not already in use, you will need to open *`vectors.c`* and look for the name of the interrupt which you plan to use. All MC68HC11 interrupting sources have been prenamed in *`vectors.c`*. To remain compatible with Mekatronix code you must use those prenames for your interrupt service routines. This prename must be placed at the head of *`isrdec.h`* as a #**`define`** <prename> preprocessor command. Without the #**`define`** <prename>, the interrupt service routine associated with <prename> will be compiled as a nul function by *`isrdec.h`*.

For example, suppose you wish to write an interrupt service routine *(isr)* using the real-time interrupt facility of the MC68HC11 microprocessor. The prename of your *isr*, or *interrupt_handler*, found in *`vectors.c`*, is **`RTI_isr()`**. You must use this name to be compatible with Mekatronix include and library files. Further, you must modify the code file *`isrdec.h`* by placing

```
#define RTI_isr()
```

with the other **`#define`** in the file. A comment explaining the function of the interrupt in your application is also advisable.

The actual writing of the *isr* code for the real-time interrupt will require a pragma definition,

```
#pragma interrupt_handler RTI_isr
```

as explained in the ICC11 manual.

17. PROGRAMMING BEHAVIOURS

When you first get your TJ™ we recommend that you write simple programs to become more familiar with the robot and its features while, at the same time, building your confidence in program development. The next section suggests coding experiments to help you do just that.

17.1 TJ™ Experiments

The numbered items below suggest a sequence of ever more complex programs and experiments you can perform to familiarize yourself with TJ™'s capabilities. These experiments will open up to you the rich variety of behaviors you can program. After each successful experiment, save your program and do not change it. Use copies of it to begin other programs, but do not write over and destroy your only copy of a successful program. You will never know when you might want to use it again, either as is, or as a basis for another program.

17.1.1 Robot Experiments

1. **Motivation**
   TJ™ motion control depends upon being able to start the motors.
   Objective
   Learn to code a simple program to turn on one of TJ™'s motors.
   Specification
   Turn on a motor 100 percent in the forward direction. Modify this program to turn on a motor 100 percent in the reverse direction.
   Questions
   Did the motor turn in the direction you thought it would when you coded it?

2. **Motivation**
   Robots often use dead reckoning to navigate and make maps in non-critical situations. The ability to go straight enhances the accuracy of dead reckoning.
   Objective
   Move the robot forward and determine if it goes straight.
   Specification

Turn on both motors in the forward direction at the same percentage.

Questions

Does the robot go straight? Which way does it prefer to turn, to the left or to the right? Which motor, left or right, appears to be turning faster for the 100% forward command?

3. **Motivation**

DC motors may not have symmetrical characteristics. Their performance may differ depending on the direction they turn.

Objective

Move the robot in reverse and determine if it goes straight.

Specification

Turn on both motors in the reverse direction at the same percentage.

Questions

Does the robot go straight backwards? Which way does it prefer to turn, to the left or to the right? Which motor, left or right, appears to be turning faster for the 100% reverse command? Are your results consistent with Experiment 2?

4. **Motivation**

Quantification of the deviation from straight-line motion may lead to compensation techniques to improve that motion.

Objective

Measure the robot’s deviation from straight-line motion when it is supposed to be going straight.

Specification

Turn on both motors at 100% forward for 10 seconds and stop the robot.

Procedure

This experiment can be performed easily on a tile floor. Just line up the left wheel on a tile line and determine how far the wheel has deviated from the axis of that tile line after the robot stops. Extend the wheel axis at the start and stop positions with a string. Assuming the robot is turning in a big circle, the intersection of the two strings determines the radius of the circle(*Why?*).

Questions

After 10 seconds, how many inches has the robot moved forward along its initial direction? How many inches has the robot veered from the line of its initial direction?

5. **Motivation**
   A robot often spins as a preliminary maneuver to escaping a difficult situation.
   Objective
   Learn to spin TJ™ in various ways for a fixed length of time.
   Specification
   Make the robot spin clockwise about the left or right wheel. Modify your program to make TJ™ spin about its center axis.
   Procedure
   At the beginning of the experiment, place erasable marks on a tile floor at the outside edge of each wheel, just below the center of the axle. Draw a line between the two marks. The center of this line is the center axis of the robot. At the end of the experiment repeat the marking procedure. Measure the distance and the compass heading between the center of the two lines you have drawn.
   Questions
   For each of the two types of spins, did the robot stay in one place or did it drift? Measure the drift as described in the procedure. Can you explain what you saw?

The experiments listed above will help you understand some of TJ™'s motion characteristics. You can devise others to test TJ™'s sensor capabilities.

17.2 Scope

Programming behaviors is what autonomous mobile robots is all about, or, at least a substantial part of what it is all about! Without being technical, a behavior is whatever the robot does. The emphasis is on action! From the engineering viewpoint, you want to program behaviors that produce useful results. Make a robot vacuum cleaner, or a robot valet. With an artistic eye, you want to program behaviors that esthetically please or excite. Why not make TJ™ dance? A ballet on wheels! From the scientific perspective you can inquire about the scope of machine intelligence and test your theories on a real robot! Out of intellectual curiosity and the creation urge, you might want to develop physically embodied animats, artificial animals. Develop your own ecology with predator robots that drain the prey robots' batteries and prey robots that hide and avoid predator robots and seek battery recharging stations as food sources. Or, you can tailor a robot to enter many of the robot contests around the world. Many of these contests require manipulation and sensors not supplied with TJ™. But, with one or more MSCC11 single chip computers controlling the additional sensors and servo driven manipulation devices, a TJ™ can often be expanded to meet contest requirements.

17.3 Some Possible Behaviors

TJ™ programs provide the basic hardware interrupt and device driver routines for the robot. These allow the user to access the sensor readings and drive the motors and servos. With these routines, the user can program an unlimited number of behaviors. A representative set, but, by no means, an exhaustive set, of primitive set of behaviors, from which more complex ones can be developed, are listed in Table 2.

Table 2 Primitive Behaviors

TALRIK JUNIOR™	TALRIK JUNIOR™	TJ-ARGOS™	TJ-ARGOS™
Collision Avoidance	Collision Detection	Line following	Light following
IR Light avoidance	Pushing	Collision detection	Shy behavior
Aggressive behavior	Exploring behavior	Wall following	Beacon tracking

17.4 Advice on Developing Behaviors

The following advice is based on several years experience teaching engineering students to program autonomous robot behaviors.

17.4.1 Vulcan Mind Meld

To effectively program a behavior for TJ™, or, quite possibly, any autonomous robot, and to gain insight into the problems encountered by your robot, you should play Vulcan to the robot and imagine performing a Vulcan mind meld with it. All you Trekkie fans know what this means. But, to be specific, try to perceive the universe as the robot does with its limited capabilities. As you imagine yourself one with the robot, play out different sensations and responses. Help yourself by actually recording robot sense data and examine typical responses, or responses to special environmental conditions of interest in the behavior you are developing. The mind meld will help prevent the common error of asking the robot to respond to environmental conditions it cannot detect with its sensors! While this statement is so totally obvious, it is also a difficult self-discipline to psychologically enforce. Why? Humans typically interact with each other or intelligent animals, expecting and perceiving sophisticated behavior and sensory performance. These expectations seem to subconsciously creep into our agenda when working with autonomous machines, often with disappointing results! Autonomous robots have no where near the sensory and behavioral capabilities of an insect, let alone higher animals.

17.4.2 Virtual Mind Meld©[3]

To assist in perceiving the universe as the robot does, you can write programs to generate computer graphic displays that depict the robot's perception in any sense that makes communication with the robot easier. Robot Rorschach tests, color maps…a virtual robot environment. This process we coin as a *Virtual Mind Meld.* The robot portrays its reality in the computer graphics medium to create a virtual reality to bridge the species communication barrier.

17.4.3 Relative calibration of sensors of the same type

Manufacturing tolerances, circuit tolerances, and mounting variations make it possible for two instances of the same type of sensor to respond differently to the same stimulus. Behaviors, therefore, should not be programmed to depend upon two sensors of the same type producing identical responses to the same stimulus. Instead, program sensors of the same type to relatively calibrate themselves in some fixed environment. For example, place a cardboard box in front of, and parallel to, the wheel axis of a TJ™. Measure the response of the two front IR detectors. Note the differences in the readings. If there are none, that's great! In general, however, they will differ somewhat.

[3] Virtual Mind Meld©Mekatronix™

Program the robot’s behaviors to respond to relative sense stimuli, not absolute sense measurements. This will make the robot behave more organically and robustly to uncertain, dynamic environments.

17.4.4 Adjusting to Ambient Conditions

A programmed behavior will often be *brittle*, not flexible or adaptive, if that behavior depends upon specific magnitudes of robot sensor readings. Brittle behaviors fail when the environment changes from the environment in which the behavior was developed. The smaller the change that causes the failure, the more brittle that behaviors is. For example, the IR detectors on TJ™ will detect white objects at larger distances than dark objects. Suppose a collision avoidance algorithm sets a threshold value of the IR as an indication of an impending collision. If this threshold is determined experimentally with light colored obstacles, then dark colored obstacles will not be detected and the robot will bump into them. On the other hand, if the threshold is set for dark colored obstacles, the robot will end up spinning in circles in a light colored environment because it detects threats everywhere. The solution is not to pick an average color threshold, but rather, program the robot to adjust its threshold downward if it has not detected a collision for some specified time, or, to adjust the threshold upward if it is colliding too frequently. The difficulty, of course, is determining exactly what the “specified time” between collision should be or what “colliding too frequently” means! The easy, but difficult to implement, answer is to let the robot learn these parameters based upon some performance criteria.

Robot behaviors and sensors, therefore, should adjust to ambient conditions. Biological organisms perform this function fantastically well. The human eye adjusts to bright sunlight or a darkened cathedral with dimly lit candles. This procedure is easier to state then execute, but serves as a general principle.

17.4.5 Create simple behaviors

The beginning robot practitioner usually formulates behaviors too complicated to implement directly. With experience, the virtue of simple, direct behaviors becomes apparent. Complex behaviors should be broken down into sequences of simple, primitive behaviors. If this can be done, the chances of successful implementation are high. If not, there is little value in trying to implement such behaviors directly.

17.4.6 Build on simple behaviors

As the user accumulates a repertoire of primitive behaviors, complex behaviors open up. Perhaps the easiest way to generate complex behaviors is simply to sequence a collection of primitive behaviors. For example, wall following might be decomposed as follows: 1) detect a “large” object, 2) approach the object until “near”, 3) turn until the robot front-to-rear axis aligns “parallel” with the “surface” of the obstacle, 4) move “parallel” to the obstacle surface. At each instant of time a particular behavior in the sequence is invoked based on the current state of the robot and its sensory inputs. Of course, the programmer will have to establish to the robot’s perception the meaning of such terms as “large”, “near”, “surface”, and “parallel”. Remember to Vulcan Mind Meld!

17.5 Integrating Behaviors

More complex behaviors may require the combination of primitive behaviors in a way not well understood. Artificial neural network activation, opinion guided reaction, non-linear dynamics, and fuzzy logic all offer techniques for integrating behaviors. Each technique offers specific advantages and specific difficulties. Discussion of such issues is beyond the scope of this manual. The reader’s attention is brought to this matter to encourage investigation into these possibilities.

18. TALRIK JUNIOR™ TROUBLESHOOTING GUIDE

To check out your robot, download the program file `avoidtj.s19` in the `tjcode` directory in the distribution software and run it.

***Note**. If the robot moves, avoids obstacles, and responds to a bumper switch closure when running avoidtj.s19, all hardware systems work and your problem must be software. If TJ™ does not avoid obstacles and does not respond to bump contact, use the following trouble guide.*

Is the Problem Hardware or is the Problem Software?

Engineers point their finger at the software and the programmers fault the hardware! The reality, unfortunately, is that determining the causes of an error or malfunction can be frustratingly difficult. Problems can arise from either or both sources. When you encounter an intractable debugging problem, the fault almost invariably stems from a false assumption about the working state of your robot and your program. Systematic testing of both hardware and software, on an incremental basis, can greatly reduce errors and help you to isolate causes of errors when they do occur…and they will occur, that is a promise!

***Note.** In the troubleshooting guide below, you should always keep in mind that problems may have multiple causes and only some highly probable ones are mentioned.*

Check Battery Voltage

Many hardware and apparent software problems with your TJ™ robot result from uncharged or low batteries. Under low battery conditions, loaded programs often become corrupt or the robot processor resets every time the motors pull large amounts of current from the batteries, for example, when the motors make substantial changes in speed or direction, as in starting and stopping.

Symptoms of Low Battery Voltage

As the batteries become low the red power LED blinks when the motors pull large currents. This warning should be heeded and the batteries charged or exchanged for fresh ones. A more radical symptom occurs when the batteries drain too low to support current demands of starting and stopping the motors. The processor rapidly resets and restarts your program each time the motors demand current. In such cases, TJ™ stutter-rolls forward and does little else.

IMPORTANT! When troubleshooting first check to make sure that your batteries are fully charged and the red power LED lights when the ON/OFF switch is flipped to ON.

The numbered items below indicate the most common problems encountered their possible causes and associated fixes.

1. **My program does not compile**. *Refer to Section 6 and redo the instructions appropriate for your case. Be sure you execute seticctj.bat* each time *you open a new DOS Window or start up in DOS. If that does not work, you might want to repeat the whole installation procedure in Section 6.*

2. **My program compiles with errors**. *Here you must use your program debugging skills. Program debugging is beyond the scope of this manual.*

3. **My program compiles, but does not download**. *Refer to Section 4.*
   3.1. *Be sure the Download/Run switch is in the Download position.*
   3.2. *Verify that the robot is connected to the right COM port.*
   3.3. *Check and verify all the physical connections between your PC, the Communications Board (MB2325) and the TJ™. Diode D1 should be on when the Com-board is connected with your PC. LED D2 should light when the reset button on TJ™ is held down.*

4. **My program downloads, but nothing happens when I press the reset button.**
   4.1. *Check to make sure you have switched the DOWNLOAD/RUN switch to RUN and that the power switch is ON.*
   4.2. *Press the reset button several times. Sometimes the switch does not make proper connection.*
   4.3. *Your program may actually be running, but program logic errors prevent any observable robot behavior. This is a tough one. Download and run a working program such as* `avoidtj.s19` *to verify all TJ™'s hardware functions. If you are successful, then the problem must be in your program.*
   4.4. *If you cannot successfully run* `avoidtj.s19 but you have before`*, then check battery connections. More remote, unplug the battery pack and check the reset button with an multi-meter for switch closure (short circuit) when the button is pressed. Replace the switch if it does not operate properly.*
   4.5. *If the switch works, check continuity of the wiring to the reset button and repair if necessary.*

5. **My program downloads, but when I press the reset button the robot just jerks and stutters.**

5.1. *Check for weak batteries. Weak batteries can cause the processor to reset every time the motors demand current. Weak batteries can also corrupt your program. Replace batteries with fresh ones or recharge batteries.*
5.2. *Your program may be syntactically correct but possesses logic errors, causing the robot to behave radically. If you believe your code to be correct, download and run* `avoidtj.s19` *to verify, or not, all TJ™'s hardware. If the hardware checks out, the problem is in your program.*

6. **Power LED Does Not Light when the *ON/OFF* switch is flipped to *ON*.**
   6.1. *Batteries are not charged. Recharge or replace with fresh ones.*
   6.2. *Battery cable from the battery to the top plate is loose or disconnected. Reseat connector.*
   6.3. *Battery cable from the battery to the top plate is reversed. Turn connector 180 degrees and reseat securely.*
   6.4. *Battery snap connector is loose. Resnap.*
   6.5. *Batteries have vibrated loose and are not making good contact in the plastic holder. Reseat batteries in the holder. Wiggle them to see if the red LED lights.*
   6.6. *One or more of the wires to the battery connector have broken away from the connector pin. Resolder the wire(s) to the connector and reseat connector.*
   6.7. *One or more wires in the power circuit involving the LED have broken loose. Resolder.*

   If none of the above work, but you can download programs and run the robot, then the red LED power-on indicator must be replaced.

7. **My program downloads, but when I press the reset button the robot backs up and turns repeatedly.**
   7.1. *The bumper has probably jammed a bumper switch closed. Adjust the bumper so that all switches clear the bumper.*
   7.2. *If the bumper clears the switches and the behavior persists, check your code for program logic errors. If you believe your code to be correct, download and run* `avoidtj.s19` *to verify, or not, all TJ™'s hardware. If the hardware checks out, the problem is in your program logic.*

8. **My program runs, but the robot always bumps into things.** *Possible sources of trouble: programming error, IR emitter and detector connections, IR LED wired in reverse biased.*
   8.1. *Verify that all IR connections are secure and properly oriented and no wires have broken.*
   8.2. *If you have a monitor driven by a black-and-white CCD camera without an IR filter, you can verify that you have IR by pointing the emitters at the camera. The IR will appear as visible light in the image. You can easily see if flipping the IR emitter connector forward biases the diode because it will begin to shine.*

9. **Motors turn the wrong direction.**

9.1. *Possibly a wrong motor specification in your program.*
9.2. *Verify the left motor plug is mated with the left motor header. If not reverse motor plugs.*

Comments on the IR System

IR Detector Connectors: Black (Gnd) wire fits the pin next to the edge of the can
We have intentionally connected the IR detectors incorrectly and have not observed any damage to the detectors. Of course, they do not function when connected incorrectly and we recommend avoid doing so.

IR Emitters Shine
How can you tell if the IR emitters emit IR? Run the program `avoidtj` and see if TJ™ avoids obstacles placed on either side of the robot.

19. TALRIK JUNIOR™ TECHNICAL SPECIFICATIONS

The following paragraphs provide a brief description of TALRIK JUNIOR™'s technical characteristics.

19.1 Mechanical Structure

1. All of TJ™'s body parts are either plastic or beautiful, strong, durable, 1/8 inch thick, 5-ply, birch model airplane plywood.
2. TJ™ fits into a right circular cylinder 7inches in diameter by 3.25 inches high. (Volume approximately 125 cubic inches or 0.072 cubic feet)

19.2 Power Requirements

1. Six AA rechargeable Nickel-Cadmium batteries (ENERGIZER™ or EVEREADY™), 600 ma-hr, 5.4- 7.2 volts (Sold separately).

WARNING!
USE ONLY NiCd BATTERIES FOR TJ™. DO NOT USE ALKALINE OR OTHER BATTERY TYPES WHICH WILL DESTROY THE ROBOT ELECTRONICS.

2. Recharger, 12 volts D.C. rated at 200ma (Sold separately).

19.3 Actuation

Gearhead DC motor drive for each wheel.

1. 5.4- 7.2 Volts
2. 100 -120 ma under load, 80 ma no-load
3. 1.25 revolutions/sec at 7.2 volts (full battery charge). Speed decreases proportionally to the voltage as it drops.

19.4 Robot Controller

1. MC68HC11E2 (256 Bytes of RAM, 2K of EEROM)

2. 5 Volt regulator
3. Low voltage inhibit reset circuit
4. 3-Pin Male Headers (Ground-Power-Signal Rails) for connecting sensors and motors. User expandable.

19.5 Basic TJ Sensor Suite

1. Two Forward Looking IR Emitters, wavelength equals 940nm.
2. One Backward Looking IR Emitter, wavelength equals 940nm.
3. Two Forward Looking IR Detectors for 40KHz modulated 940nm IR.
4. Three Front bumper Momentary Tactile Switches
5. One Back Bumper Momentary Tactile Switch
6. User expandable.

19.6 Robot Control Switches

1. Reset push button
2. Toggle switch: Download Program and Run Program
3. Toggle switch: Off-On.

19.7 TJ System Support Software

TJ™ programs can be written in MC68HC11 Assembly Language, C, or BASIC.

1. Sensor and motor routines provided in assembly language.
2. PCBUG11 freeware for downloading Motorola S19 files.
3. Freeware version of Basic for programming TJ™.
4. Freeware MC68HC11 Assembly Language.

Separate purchase of a commercial C compiler is also available. Contact an authorized Mekatronix distributor (www.mekatronix.com/distributors) for more information.

19.8 TJ Applications Software

Mekatronix™ provides a `avoidtj`, a collision avoidance program that allows TJ™ to explore his environment and avoid bumping into things, most of the time! If TJ™ does bump into something, his bumpers tell him and he moves away.

You can develop your own applications, limited only by your imagination and 2KB of memory!

1. Make TJ™ do figure eights, or any other shape, while at the same time avoiding people and furniture.
2. Program TJ™ to be an artist who draws on cardboard with a pen attached to his body (pen holder not included) (Be sure TJ™ stays on the cardboard!).
3. Design an obstacle course for TJ™ to learn.
4. Scare TJ™ by blasting him with your TV remote!
5. Write a program so TJ™ will be attracted to your TV remote!
6. Control TJ™'s behavior with your TV remote…an IR controlled vehicle!

7. Get two or three TJ™'s and program them to follow each other in single file.
8. Get three TJ™'s and teach them to flock like goslings as they move around together.

19.9 Serial Communication Hardware

TALRIK, Jr's serial communications require logic signals (5 volts and Ground). Downloading your software applications to TALRIK, Jr. from a PC requires voltage conversion of the RS232C levels to the TALRIK, Jr. logic levels. If you do not have this capability already, the additional purchase of an MB2325 communications board and a 6-wire RS-232C communications cable will solve the problem. Only one MB2325 board and cable is necessary to enable you to sequentially load and download any number of Mekatronix™ robots, since the MB2325 board can remain attached to the PC and not the robot.

To develop your applications requires communications between a Personal Computer and TJ™. The additional purchase of an MB2325 communications board and a 6-wire RS-232C communications cable will provide the hardware for that capability. Only one MB2325 board and cable is necessary to enable you to sequentially load and download any number of Mekatronix™ robots, since the MB2325 board can remain attached to the PC and not the robot.