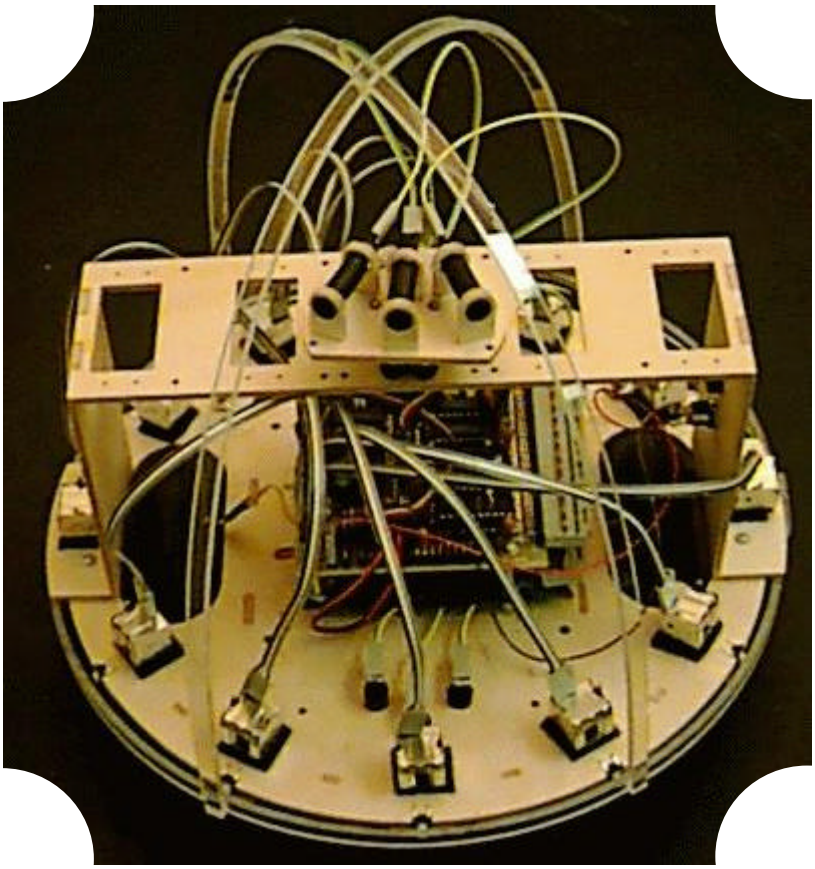
TALRIK^{II} Users Manual

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. You 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™.

1. **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 accept 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.

NETWORK SITE: http://www.mekatronix.com/
TECHNICAL SUPPORT: ***tech@mekatronix.com***

MANIFESTO

Mekatronix™ espouse the view that the personal autonomous agent 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!

TABLE of CONTENTS

LIST of FIGURES

LIST of TABLES

1. SAFETY AND HANDLING

1.1 TALRIK^{II}™'s Static Sensitive Parts

Some of TALRIK^{II}™'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 TALRIK^{II}™, exciting, enjoyable, amusing, and fascinating. TALRIK^{II}™'s small size minimizes its[1] threat to organic life, although pets and small children might find TALRIK^{II}™ either an interesting playmate or an alien to avoid. A child's hands, feet, and mouth probing TALRIK^{II}™ may lead to minor injury to both child and robot, so, to eliminate adverse reactions with pets, children, or nervous adults, be sure TALRIK^{II}™ operates under supervision by someone who knows how to turn it (him!) off. Remember, TALRIK^{II}™ is an autonomous machine and, as such, becomes "out of control"[2] once set in motion.

1.3 Holding, Carrying, and Transporting TALRIK^{II}™

TALRIK^{II}™'s small size and portability make it ideal for safe demonstrations and presentations of machine intelligence algorithms. Porting TALRIK^{II}™ from desk-top platform to floor and back, the most frequent carrying operation, requires handling TALRIK^{II}™ carefully. The recommended carrying technique is to firmly hold TALRIK^{II} on both sides at the base of the bridge supports next to the wheels with fingers curled underneath the robot and thumbs on top. Other carrying techniques can be devised by users at their own risk.

To transport TALRIK^{II}™ for long distances, place it securely padded into a suitably sized box.[3] Do not expose TALRIK^{II}™ to extreme cold or heat in an automobile during winter and summer for more than a few hours. Although TALRIK^{II}™ has survived a hot automobile for 12 hours in *40° C*, such treatment cannot help but reduce its lifetime.

TALRIK^{II}™ has flown in airline luggage compartments across the Atlantic, ridden in automobiles, carried, exposed, by hand across the University of Florida Campus on a bright Spring day, and passed around in class to a group of electrical and computer engineering students, all without ill effects! The following caution, however, is our official advisement.

Caution: *TALRIK^{II}™ 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 TALRIK 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 TALRIK is hard to resist and we cannot guarantee that personal pronouns usage will not appear!

[2] Future versions will have remote control for remote starting and stopping and data transfer.

[3] An accessory TALRIK™ carrying case will be offered soon.

1.4 Storage

Place TALRIK^{II}™ 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. GETTING TALRIK^{II}™ READY

This section tells you how to unpack and prepare TALRIK^{II}™ for operation.

2.1 Unpacking TALRIK^{II}™

1. TALRIK^{II}™ Robot

Carefully unwrap TALRIK^{II}™ from the bubble wrap being careful not to catch wires or drop the robot.

2. 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. In the case of the IR detectors, the black side of the cable should line up with the edge of the case. Check the connectors under the robot, the LED connectors are keyed so just replace them the way they were. Make sure the battery snap is firm. Check the bumper to see if it is pushing any switches. If it is, just gently shift the bumper away from the switch. Latter program testing will reveal any specific problems.
3. Remove the Software disks and any other documentation.

You may have also purchased the following (recommended)

4. A wall plug adapter for recharging the robot ,
5. An MB2325 serial communications board and cable,
6. 8 Nickel Cadmium rechargeable batteries.

2.2 Charging TALRIK^{II}™

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 battery charged.

2.2.1 If you purchased Charger and Batteries:

1. After the robot is unpacked plug the AC adapter into a power socket and insert the 12 volt DC output power plug into the power jack on the left bridge support of the robot.
2. Place the mode switches on the robot in *POWER-SAVE* and *DOWN-LOAD* mode.

*2.2.2 Using your own batteries (**IMPORTANT: NiCads only**)*

1. Place the robot switch into *POWER-SAVE* mode.
2. Flip TALRIK^{II}™ over, being careful of the bumper. The preferred method is to place TALRIK^{II}™ between your legs so the top of the bumper is not resting on anything.
3. Unhook the 9v snap connector and the single connector on the side of the battery pack. Unfasten the Velcro™ battery pack strap and remove the pack.
4. Place your NiCad batteries into the pack oriented with proper voltage polarity as labeled on the pack. Failure to do so will destroy all the electronics.

5. Replace the Velcro™ strapping tightly. Replace the 9v snap connector and the single line connector.

3. TALRIK^{II}™ OVERVIEW

This section provides the user general orientation to TALRIK^{II}™ .

3.1 What Can TALRIK^{II} DO?

TALRIK^{II} provides a 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 TALRIK^{II} and other 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 to the seventh generation, these days will seem primitive indeed.

3.2 Operating Environment

TALRIK^{II}™ is designed to operate in an office or home environment with smooth tiled floors. Although TALRIK^{II}™ 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 TALRIK^{II} and should be avoided.

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

Caution: *TALRIK^{II}™ 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).*

3.3 Physical Orientation

TALRIK^{II}'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^{II}" label appears on the front of the robot.

3.4 IO Memory Addresses and Sensor Selection

The MRSX01 generates four Input Register Select signals (IS0, IS1, IS2, IS3) and four Output Register Select signals (OS0, OS1, OS2, OS3). The Memory Addresses of these signals and their assigned function appears in Table 1. Only OS0 and OS1 have predefined functions. The others can be used to expand the sensory and actuation capability of the TALRIK robot even further. OS1 controls the digital outputs DIG_OUT[1..8] that drive the IR emitters through current limiting resistors. OS0 controls the Multiplexer/Motor_Direction Register (MMR) whose eight bits from the most significant to the least significant are

MMR

7	6	5	4	3	2	1	0
MOTOR1	MOTOR0	OUT1	SEL4	SEL3	SEL2	SEL1	SEL0

OUT1 is simply a Digital Output Bit that appears on IOHEADER[2]. MOTOR0 controls the direction of the right motor and MOTOR1 controls the direction of the left motor. The five select lines SEL0 to SEL4 control the three 8-to-1 Analog Multiplexers that feed into PE0 and PE1.

indicates how the SEL bits select the multiplexer inputs. Table 3 indicates which sensor input is selected by the SEL bits and the corresponding A/D_Channel/PortE_Pin on which the sensor data appears. Observe that the 8-pin header ANALOG provides six unallocated analog inputs to PE1 as well as ground (ANALOG[7]) and power (ANALOG[8]). The ANALOG header allows you to readily expand to six more analog sensor inputs.

Table 1 TALRIK IO Memory Addresses

IO Enable Line	Memory Address	Function
OS0	FFB8	Load MMR from Processor Data Bus (PortC).
IS0	FFB8	Input Select 0: IOHEADER[Pin 4]
OS1	FFB9	IR LED DRIVER
IS1	FFB9	Input Select 1: IOHEADER[Pin 5]
OS2	FFBA	Output Select 2: IOHEADER[Pin 3]
IS2	FFBA	Input Select 2: IOHEADER[Pin 6]
OS3	FFBB	Output Select 3: IDC60 Header Pin 55
IS3	FFBB	Input Select 3: IDC60 Header Pin 56

Table 2 Multiplexer Control with MMR SEL Bits

SEL4	SEL3	SEL2	SEL1	SEL0	Enables Multiplexer Input
1	0	0	0	0	MUX0, MUX16
1	0	0	0	1	MUX1, MUX17
1	0	0	1	0	MUX2, MUX18
1	0	0	1	1	MUX3, MUX19
1	0	1	0	0	MUX4, MUX20
1	0	1	0	1	MUX5, MUX21
1	0	1	1	0	MUX6, MUX22
1	0	1	1	1	MUX7, MUX23
0	1	0	0	0	MUX8
0	1	0	0	1	MUX9
0	1	0	1	0	MUX10
0	1	0	1	1	MUX11
0	1	1	0	0	MUX12
0	1	1	0	1	MUX13
0	1	1	1	0	MUX14
0	1	1	1	1	MUX15

Table 3 Sensors Selected by MMR SEL Bits

SEL4	SEL3	SEL2	SEL1	SEL0	Sensor (AN0/PE0)	Sensor (AN1/PE1)
1	0	0	0	0	Charge Voltage Detect	Rear Bumper
1	0	0	0	1	Battery Power Level	Front Bumper
1	0	0	1	0	IRDT7	ANALOG[1]
1	0	0	1	1	IRDT8	ANALOG[2]
1	0	1	0	0	IRDT9	ANALOG[3]
1	0	1	0	1	IRDT10	ANALOG[4]
1	0	1	1	0	IRDT11	ANALOG[5]
1	0	1	1	1	IRDT12	ANALOG[6]
0	1	0	0	0	IRDT13 (Optional)	
0	1	0	0	1	IRDT14 (Optional)	
0	1	0	1	0	CDS1	
0	1	0	1	1	CDS2	
0	1	1	0	0	CDS3	
0	1	1	0	1	CDS4	
0	1	1	1	0	CDS5	
0	1	1	1	1	CDS6	

The first six IRDT input sensor feed directly into the A/D channels AN2 through AN7 on PE2 to PE7. For convenience this information appears below in tabular form:

IRDT6	IRDT5	IRDT4	IRDT3	IRDT2	IRDT1
AN6/PE7	AN5/PE6	AN4/PE5	AN3/PE4	AN2/PE3	AN1/PE2

3.5 Sensor Layout

Figure 1 schematizes the standard layout of TALRIK^{II}'s sensor suite. Table 4 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.

Table 4 TALRIK™'s Sensor Suite and MRSX01 Sensor Connectors

TALRIK Label	**MRSX01 Connector**	**Name**	**Function**
IRDL	IRDT12	Infrared Detector Left Side	Left Side Proximity Sensor
IRDFL, IRDFML, IRDFM, IRDFMR, IRDFR	IRDT2, IRDT3, IRDT4, IRDT5 IRDT1	Front Infrared Detector (Left, Middle-Left, Middle, Middle-Right, Right)	Front Proximity Sensors
IRDR	IRDT7	Infrared Detector Right Side	Right Side Proximity Sensor
IRDBL, IRDBML, IRDBM, IRDBMR, IRDBR	IRDT11, IRDT10, IRDT6, IRDT9 IRDT8	Back Infrared Detector (Left, Middle-Left, Middle, Middle-Right, Right)	Back Proximity Sensors
IRDLW (Optional)	IRDT13	Infrared Detector Left Wheel	Edge Detection under the left wheel.
IRDRW (Optional)	IRDT14	Infrared Detector Right Wheel	Edge Detection under the right wheel.
NCDSL NCDSM NCDSR	CDS6 CDS5 CDS4	Nose Cadmium Sulfide Photoresistors (Left, Middle, Right)	Line following, Edge Detection
ECDSL ECDSM ECDSR	CDS3 CDS2 CDS1	Sensor Head Cadmium Sulfide Photoresistors	Light measurement, beacon tracking, phototaxis
BSWFL, BSWFML, BSWFM, BSWFMR, BSWFR	F_BUMP4 F_BUMP2 F_BUMP1 F_BUMP3 F_BUMP5	Front Bumper Switch (Left, Middle-Left, Middle, Middle-Right, Right)	Front contact Sense
BSWBL, BSWBML, BSWBM, BSWBMR, BSWBR	R_BUMP2 R_BUMP2 R_BUMP1 R_BUMP3 R_BUMP3	Back Bumper Switch (Left, Middle-Left, Middle, Middle-Right, Right)	Rear contact Sense

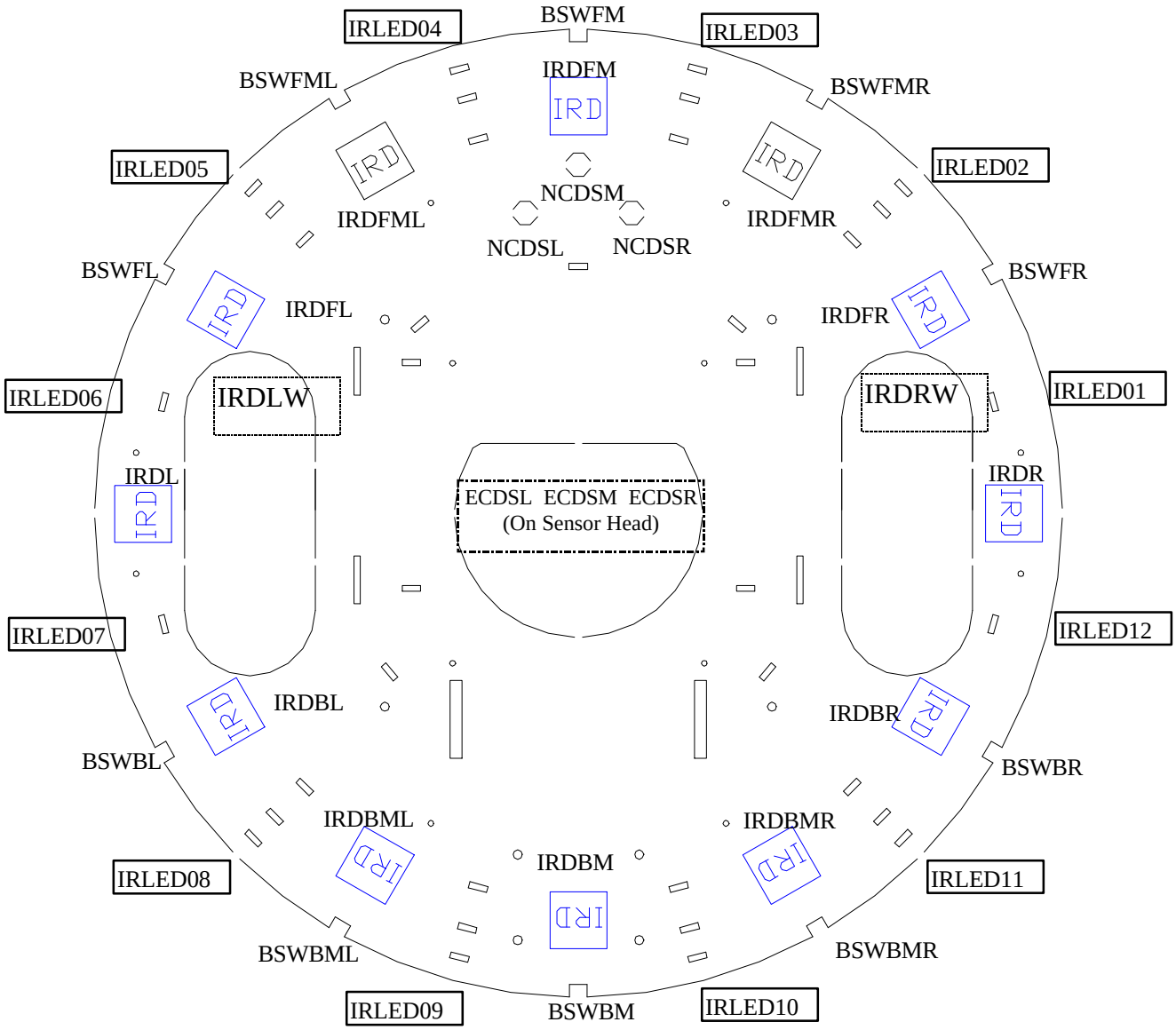


Figure 1 Sensor layout on top of the TALRIK plate. The lead symbols are B= Bumper, N=Nose, IRD = Infrared Detector, IRLED = Infrared LED. The following symbols mean: B = Back, F= Front, SW = Switch, L= Left, M= Middle, R= Right, CDS= Cadmium Sulfide photoresistor, W= wheel. For example, BSWBMR = Bumper-Switch-Back-Middle-Right and IRDFL = IR Detector Front Left; NCDSM = Nose-Cadmium-Sulfide-Middle; IRDLW=IRD-Left-Wheel. Note that IRDLW and IRDRW are on the underside of TALRIK™'s plate while the other IRDs are on the top side. ECDSL, ECDSM, and ECDSR mount on the Sensor Head (TALSHD01)

While not strictly binary, the voltage gaps between the individual bump switch sensor voltages are large enough to permit TALRIK™ to test for any combination of simultaneous bump switch closures. This feature provides TALRIK™ with quite a capable somatic ability.

3.6 Sensor Map and Program Names

Column one of Table 5 relates the TALRIK^{II} standard sensor labels to the corresponding *#define constants* of column two declared in the header file `icctk/Include/muxsentk.h` of the TALRIK^{II} distribution software. The hexadecimal replacement values of the *#define constants*

Table 5 The mapping of TALRIK Sensor Labels, #define labels, and Program Data

TALRIK Label	#define constant	MMR SEL Field (Hex)	Associated Program Data Names: Automatic, Programmed
IRDL	IRDT12	0x17	AtoD[0][4] , IRDT[11]
IRDFL	IRDT2	N.A.[1]	AtoD[3] , IRDT[1]
IRDFML	IRDT3	N.A.	AtoD[4] , IRDT[2]
IRDFM	IRDT4	N.A.	AtoD[5] , IRDT[3]
IRDFMR	IRDT5	N.A.	AtoD[6] , IRDT[4]
IRDFR	IRDT1	N.A.	AtoD[2] , IRDT[0]
IRDR	IRDT7	0x12	AtoD[0] , IRDT[6]
IRDBL	IRDT11	0x16	AtoD[0] , IRDT[10]
IRDBML	IRDT10	0x15	AtoD[0] , IRDT[9]
IRDBM	IRDT6	N.A.	AtoD[7] , IRDT[5]
IRDBMR	IRDT9	0x14	AtoD[0] , IRDT[8]
IRDBR	IRDT8	0x13	AtoD[0] , IRDT[7]
IRDLW	IRDT13	0x08	AtoD[0] , IRDT[12]
IRDRW	IRDT14	0x09	AtoD[0] , IRDT[13]
NCDSL	CDS6	0x0f	AtoD[0] , CDS[5]
NCDSM	CDS5	0x0e	AtoD[0] , CDS[4]
NCDSR	CDS4	0x0d	AtoD[0] , CDS[3]
ECDSL	CDS3	0x0c	AtoD[0] , CDS[2]
ECDSM	CDS2	0x0b	AtoD[0] , CDS[1]
ECDSR	CDS1	0x0a	AtoD[0] , CDS[0]
Charge_Volts	Charge_Volt	0x10	AtoD[0], charge_volts()
Battery_Level	Batt_Volt	0x11	AtoD[0], battery_level()
Rear_Bumper	R_Bump	0x10	AtoD[1] , rear_bumper()
Front_Bumper	F_Bump	0x11	AtoD[1] , front_bumper(
(Reserved)[3]	Xanalog1	0x12	AtoD[1] , Xanalog[0]
(Reserved)	Xanalog2	0x13	AtoD[1] , Xanalog[1]
(Reserved)	Xanalog3	0x14	AtoD[1] , Xanalog[2]
(Reserved)	Xanalog4	0x15	AtoD[1] , Xanalog[3]
(Reserved)	Xanalog5	0x16	AtoD[1] , Xanalog[4]
(Reserved)	Xanalog6	0x17	AtoD[1] , Xanalog[5]

[1] Not Applicable. [2] No external connector. [3] Reserved for future Mekatronix standard sensors, but may be used for customer applications. [4]AtoD[ ] data structure loaded automatically every 2 milliseconds by TOC5_isr in `clocktk.c`. The values loaded into AtoD[0] and AtoD[1] depend upon the SEL field in the MMR register.

appear in column three. These values indicate which multiplexer selection field of the MMR register enables that particular sensor. The MMR register SEL field enables the corresponding sensor, designated in the first column, to be converted by the AD every two milliseconds and stored in the eight integer vector AtoD as specified by column four.

To load the MMR register with a hex multiplexer selection found in column three, simply execute the C-function *mux_sel(constant*). For example, *mux_sel(IRDT12) = mux_sel(0x17)* loads the bit pattern *10111* into the SEL field of the MMR register. The values stored into AtoD[0] every 2 msec. will come from the IR detector connected to IRDT12 and is labeled as IRDL, the IR detector on the left side of the robot.

3.7 TALRIK^{II}'s™ Bridge

TALRIK^{II}'s™ horizontal cross piece, referred to as the *bridge*, supports a servo driven sensor head with three cadmium sulfide cells. These photoresistors allow TALRIK^{II}™ to sense different ambient light levels or follow beacons. You can use the head to mount other sensors or design modular heads which you can take on and off easily.

The bridge allows extensive sensory and servo expansion capabilities. The bridge can easily support the following arrangements of servos and single-chip computers: 1) five servos, 2) four servos and an MSCC11, 3) three servos and two MSCC11s, or 4) two servos and three MSCC11s. Too run more than two servos, however, requires the addition of an MSCC11 board.

The bridge offers the user a convenient platform from which to mount other sensors of choice. For example, the bridge can be used to support a pan-head sensor array, or a pan-tilt head sensor array.[4]

3.8 Switches

TALRIK^{II}™ control switches on the right bridge support provides easy access.

The red *RESET* button resets the MC68HC11 microcomputer.

The *POWER-SAVE/ON* mode switch in the *POWER-SAVE* position supplies power only to the memory and the memory enable logic. A fully charged battery pack will last about 12 hours in *POWER-SAVE* mode before being discharged.

In *DOWN-LOAD* mode the *DOWN-LOAD/RUN* mode switch permits the user to download programs using either Image Craft C or Interactive C. In *RUN* mode, a press of the reset button will invoke the start-up program the user has loaded into TALRIK^{II}™.

[4] Future accessories.

3.9 Batteries

TALRIK^{II}™ requires a minimum of eight AA Nickel-Cadmium rechargeable batteries (Mekatronix™ recommends Energizer™ 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. TALRIK^{II}™ 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 TALRIK^{II}™ really slowing down, it is time to recharge it!

3.10 Recharger

TALRIK^{II}™'s charger must be at 12 volts and be able to supply 300 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: When not in use, keep TALRIK^{II}™ connected to the charger with the *POWER-SAVE/ON* switch at *POWER-SAVE*. Otherwise, the charger must simultaneously supply the electronics as well as recharge the batteries, thus, extending the charge time considerably.

Note: If TALRIK^{II}™'s *POWER-SAVE/ON* switch is in the *POWER-SAVE* mode and TALRIK^{II}™ is not connected to the charger, the batteries will discharge slowly. A fully charged battery pack will last about 12 hours in *POWER-SAVE* mode before being discharged.

3.11 Can you Physically Damage TALRIK with Bad Software?

The answer is YES! A program that commands the robot to oscillate between 100% forward and 100% backward for 30 seconds or more will overheat, and possibly burn out, the motor driver chip. Anytime your program controls servos and motors, you should take extra care to insure this type of behavior does not occur.

WARNING!

Rapid oscillations of motor direction and speed for extended periods of time will burn out the motor control chip.

There is another case where software can damage the TALRIK. Interactive-C (IC) uses the MC68HC11 "output compare four" (OC4) pin to control its multi-tasking timing. We use OC4 to

control *servo2*. The rapid changes of OC4 will actually burn out any servo attached to the servo1 header. Clarification: servo0 header on the MRSX01 board drives a servo with the software name *servo1* and servo1 header drives a servo with the software name *servo2*.

WARNING!

IC code thrashes servo2 and could easily destroy the servo in a short amount of time. servo2 connected to the servo1-header on the MRSX01 board must be disconnected when TALRIK^{II}™ runs IC code.

3.11.1 Controlling TALRIK's Motors

When the batteries are low, the robot may stop after making a series of rapid moves. This behavior usually indicates that the robot has reset because of high motor current surges have drawn the battery voltage down so low that the voltage regulator on the processor drops out. For example, if your program changes both wheel motors simultaneously from full speed forward to full speed backward, there might be a chance of reset when the batteries are low. These reset-type conditions can be avoided by more careful motor control. Change motor speeds, hence, currents, gradually. What is "gradual" to the motor, however, will hardly be observable to you if done in tens of milliseconds. When you program smooth motor control, the robot motion becomes smooth and looks more "organic".

3.12 Viking Helmet Bumper[5]

TALRIK^{II}™'s bumper resembles an abstract *Viking Helmet*.[6] While helmets provide wide coverage and protection, they are not perfect. TALRIK^{II}'s™ helmet does present vulnerable attack points.[7] Small objects lower than TALRIK^{II}'s™ platform will not be seen. Pointed objects projecting above the platform can strike the bridge between the acrylic, vertical protection strips. Such obstacles do occasionally 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 could burn out if the surface has high friction. Our recommendation is either to 1) TALRIK^{II}™ 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 as vision flow detector. We recommend 3), although we have not done it yet! The idea behind 3) is that if TALRIK^{II} does not detect any forward motion, he assumes he is stuck and backs up, turns around, and leaves the area.

[5] *Viking Helmet Bumper* © Mekatronix™
[6] OK, so it takes some imagination! If you speak Swedish, you will get the connection.
[7] Even helmets made for humans have vulnerable points!

3.12.1 Description of the Bumper

A clear, carbon acrylic bumper girdles TALRIK^{II}™'s waist and two, crossing, curved, acrylic suspenders provide protection to TALRIK^{II}™'s bridge and, at the same time, supply the restoring spring force to the waist bumper.

Behind the acrylic bumper, ten push button switches surround TALRIK^{II}™'s waist. TALRIK^{II}™ can be programmed to distinguish any subset of button closures activated by the bumper.

Caution: DO NOT CARRY TALRIK^{II}™ by the acrylic suspenders.

3.13 Serial Communication

The MB2325 serial communications board permits the user to download and upload code and data to TALRIK^{II}™ via a 6-wire serial communications link. The six wire communications line connects into the MB2325 (Figure 2) bi-directional serial communications board at one end (lower-right corner in Figure 2) and to TALRIK^{II}™'s serial interface at the other. The MB2325 connects directly into COM1 of your personal computer 25 pin D-connector. If your computer only has a 9-pin D-connector, you will need to acquire a 25 to 9-pin plug converter.

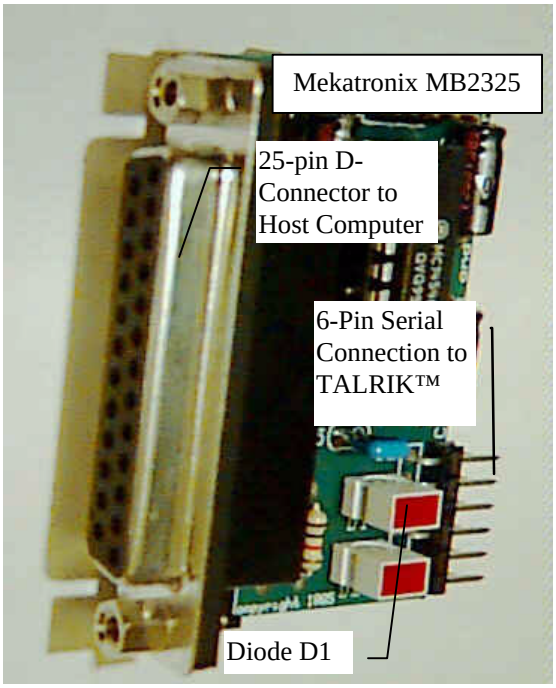


Figure 2 MB2325 Communications Board

3.14 Where is TALRIK^{II} During Program Development?

The user will spend considerable time developing and testing programs to run on TALRIK^{II}™. 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 TALRIK^{II}™ on a test platform that elevates the drive wheels off the floor or desk top so that the wheels turn freely without moving the robot. To maintain fresh batteries during these long development sessions, keep TALRIK^{II}™ connected to the charger. Also, keep the serial communications cable to TALRIK^{II}™ connected to minimize plugging and unplugging it. When the user is ready to test TALRIK^{II}™ on the floor, disconnect both the charger and the serial communications cable from TALRIK^{II}™.

3.15 Halting a Moving TALRIK^{II}

Chasing down a moving TALRIK^{II} takes more skill than transporting it. A recommended procedure is to chase it from behind and reach in to either turn the *POWER-SAVE/ON* mode switch to *POWER-SAVE* or to switch *DOWN-LOAD/RUN* to *DOWN-LOAD*. Either maneuver will stop the robot. The user can also program clever behaviors which halt the robot 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.

Future enhancements will include infrared or radio control for data acquisition and remote control.

4. TALRIK^{II} SET-UP

If you are not familiar with TALRIK^{II}, we recommend you read TALRIKII™ OVERVIEW, starting at page 9, before continuing.

4.1 Computer Requirements

Program development for the TALRIK^{II}™ 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 and use the Mekatronix *High Speed Serial Downloader* (HSSDL11) to download programs into the MC68HC11 microcontroller at a blistering 115.2KBaud.

4.1.1 Basic System (DOS)

The following computer system will meet all the basic hardware support for developing TALRIK^{II}™ 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
4. Optional, but very useful, a terminal program such as *Kermit* for DOS.

4.1.2 A Better 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. **NO** modem (see the section titled *Com port problems to avoid*, page 20)
3. A mouse on COM2
4. A printer
5. Access to the *Internet*, especially the *World Wide Web*,
6. *Windows* terminal (*terminal.exe*)

4.1.3 Com port problems to avoid

Both Randy Sargent's Interactive C for DOS (*IC-DOS*) and Motorola's *Pcbug11* , both freeware, use 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. Therefore, 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 *Pcbug* running. The modem used the same interrupt as the robot COM port and disrupted the robot serial communication.

4.2 Serial Communication between Host Computer and TALRIK^{II}™

This section assumes TALRIK^{II}™ 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 TALRIK^{II}™ via a 6-wire serial communications link. Whenever testing programs on TALRIK^{II}™, the user will connect the host computer with TALRIK^{II}™ 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.

4.2.1 Plugging MB2325 into the host computer

The MB2325 serial communications board (Figure 2, page 18) has a 25 pin, female, type D-connector. On most computers, COM1 is a 25pin 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..

4.2.2 Connecting the Host computer to TALRIK^{II}™

After the MB2325 is connected to the computer, take the 6-pin rainbow colored cable and insert either end onto J2 of 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 MB2325. Now, place the *DOWN-LOAD / RUN* switch to the *DOWN-LOAD* position. Place the *POWER- SAVE / ON* switch to *ON* position. At this point the 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. When *RESET* is released, D2 on the MB2325 turns off.

4.2.3 Verify proper operation

The lights on the MB2325 let you know that the basic connections are working. If on power up, the power light on the top circuit board of the robot did not light, 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. If the lights on the MB2325 board do not function as specified, check that no other programs or devices are using the serial port.

4.3 Software Language Support

TALRIK^{II}™ programs can be developed under Image Craft C (ICC11), the compiled version of C for the MC68HC11, or under Interactive C (IC), the C interpreter with multi-tasking developed by Randy Sargent for MIT's 6.270 course under Fred Martin's direction.

ICC11 comes in a DOS and a WINDOWS version. The WINDOWS version offers an Integrated Development Environment (IDE). Mekatronix™ sells both versions of ICC11 through its distributors. This manual does not discuss the IDE as it is explained in the ICC11 manual sent with its purchase.

IC-DOS is freeware, is supplied at no charge, and can be downloaded from the mekatronix web site (www.mekatronix.com). IC for WIN95 also can be purchased from a Mekatronix distributor. IC for WIN95 allows you to use IC in a WINDOWS environment and is much more convenient and easier to use than the IC-DOS version. Either version of IC lacks some application flexibility on the TALRIK as compared to ICC11.

ICC11 is the preferred programming language for extensive program development on TALRIK^{II}™ as it gives you complete control over all the existing hardware resources and even ones that you add yourself.

IC is useful and easy to use for simple applications. In many cases, the user may prefer to develop robot behaviors on IC. We have attempted to make TALRIK^{II}™ compatible with IC for this reason. We were not wholly successful. IC uses the MC68HC11 "output compare four" (OC4) pin to control its multi-tasking timing. We use OC4 to control *servo1*. The rapid changes of OC4 will actually burn out any servo attached to the servo1 header. Clarification: servo1 header on the MRSX01 board drives a servo with the software name *servo0* and servo2 header drives a servo with the software name *servo1*.

WARNING!

IC code thrashes servo1 and could easily destroy the servo in a short amount of time. servo1 must be disconnected when TALRIK^{II}™ runs IC code.

5. INSTALLATION OF ICC11 AND TALRIK II™ SOFTWARE

If you purchased the ICC11 software this section provides guidance in its installation and integration with the TALRIK™ 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 TALRIK 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:\icctk`.

6. 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:\icctk` 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:\icctk`
  - `Bin`
  - `Examples`
  - `Include`
  - `Lib`

- `Libsrc`

6.1 IDE Compiler and Linker Setup for TALRIK

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* window select the Linker tab.
4. In the *Linker* setup window enter the following:

a. Text section: 0x8000
b. Stack: 0x7fff
c. Data Section:
d. Additional Libraries: libtk

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

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

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

will be the default.

5. Select Preprocessor under *Compiler*:

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

Include Paths: `c:\icctk\include`

will be the default.

6. Close *Compiler* window.

Continue immediately to the next section.

6.2 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. Expand it to fill your screen. At the bottom of the window, you will see several selection buttons. These buttons allow us to configure for bootstrapping.

5. Select *Bootstrap Options*. The *HC11* window opens.
6. In *Bootloader Programming* select "External RAM".
7. Set baud rate to default (1200)
8. Close *HC11* window.

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

6.3 Integrating TALRIK™ Software with ICC11 for Windows

Integrating TALRIK™ software amounts to copying files into `icctk` directory at the appropriate places. Refer to Section ??????

7. 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:\icctk\tkcode\avoidtk.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. From *Window* on the main IDE menu bar, select *Tile* 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, correct your program in the edit window, and repeat Steps 3 and 4.

8. DOWNLOADING WITH THE HSSDL11

The Mekatronix product, the extremely low cost High-Speed-Serial–Downloader-11(HSSDL11), provides an ultra fast (115.2Kbaud) serial download rate from your PC to the robot. 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 TALRIK II. The HSSDL11 operates in WIN95 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 such.

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 before HSSDL11 can download.*

8.1 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.

8.2 HSSDL11 Setup

The HSSDL11 remembers its setup, so you will only have to perform this procedure once or whenever you want to make changes. For more details refer to Section 11.3.2.

1. Open the HSSDL11.

2. Confirm default *Settings*:
   Clock: 8MHz
   Port: COM1

3. Click on *Config* button and verify the default settings:

   CONFIG: DO NOT CHECK ANY ITEMS HERE! Default: NO item is checked.

   Features: Check only High Speed option.
   Some older computers will not allow you to communicate at 115Kbaud and you may have to "unclick this option". The baud rate then defaults to 9600.

4. Click on *Config* button again to leave configuration mode.

8.3 Download Procedure

Be sure the robot power switch is on, the robot 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 TALRIK ".s19" file of interest. Select *Open* to open the desired .s19 file. For example, `c:\icctk\tkcode\taltst.s19`.

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

3. 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.

4. Select *Download* in the HSSDL menu.
   First, the 256 bytes of the bootloader file loads at 1200 baud. Next, your program loads at 115.2Kbaud into the robot.

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

8.4 HSSDL11 Operation Tips

Often the download will fail the very first time you try it. In fact, it may take two or three tries initially. Thereafter, downloads usually succeed on the first attempt.

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.

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

If the Download prints 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 on the robot or to place the robot in download mode. 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. Typically, after a failure, the HSSDL11 with fail a second time, even with everything correct. Do not worry. Press reset and repeat the *Download* command. For some reason the HSSDL11 sometimes remembers immediate past failures and then recovers on the second attempt, usually. Of course, this problem is avoided altogether if you don't make a mistake with the switches.

9. DOWNLOADING USING IDE

You must carry out the setup procedures in Section 6 before using IDE to download any ".s19" 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 **Error! Reference source not found.**.

Open 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 8).
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. TALRIK DISTRIBUTION SOFTWARE

The distribution software includes files with important instructions and information. This information, with some additional comments and elaboration, is repeated here for your convenience. Being familiar with the contents of this package will help you find the information you need to install and use it with your TALRIK.

11.1 File Name: install_instructions.txt

11.1.1 Installation Instructions

This installation assumes that your C-Compiler has already been installed into the directory `icctk` and that the batch file installtk.bat will executes from the director holding the TALRIK Distribution Software. Execute the installation batch file with an argument specifying the root directory where the TALRIK files will be loaded. If no directory is specified, c:\icctk is assumed. The latter is recommended as all Mekatronix batch files assume the root is c:\icctk.

11.1.2 File Name: installtk.bat

This batch file integrates the TALRIK libraries and code with the compiler installation. See below for a listing of this file.

`installtk.bat`

```
@echo off

if "%1"=="" goto noargument

copy talsetup.bat %1
copy talreadme.txt %1
copy diskcont.txt %1
copy IDEsetup.txt %1

copy bintk\*.* %1\bin\*.*
copy includetk\*.* %1\include\*.*
copy libtk\*.* %1\lib\*.*

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

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

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

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

```
goto end

: noargument

copy talsetup.bat c:\icctk
copy talreadme.txt c:\icctk
copy diskcont.txt c:\icctk
copy IDEsetup.txt c:\icctk

copy bintk\*.* c:\icctk\bin\*.*
copy includetk\*.* c:\icctk\include\*.*
copy libtk\*.* c:\icctk\lib\*.*

if not exist c:\icctk\libsrctk md c:\icctk\libsrctk
copy libsrctk\*.* c:\icctk\libsrctk\*.*

if not exist c:\icctk\tkcode md c:\icctk\tkcode
copy tkcode\*.* c:\icctk\tkcode\*.*

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

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

: end
```

11.2 File Name: diskcont.txt

This file summarizes the contents of the distribution software package.

File Contents:

TALKRIK DISTRIBUTION SOFTWARE
==========
The distribution software includes the following directories:

<Assembler> Freeware assembler for the M68HC11
<Bintk> Batch files to compile and load TALRIK programs
<Includetk> TALRIK include files used specifically by TALRIK programs
<Libtk> TALRIK library files
<Libsrctk> Source code for TALRIK library files
<Pcbug11> PCBug11 monitor for the M68HC11, Motorola Freeware
<tkcode> Applications Source code for TALRIK

Assembler directory

This directory contains as11.exe, a freeware assembler for the M68HC11

Bintk directory

This directory contains the batch files comptk.bat, loadtk.bat, cltk.bat which let you 1. Compile a TALRIK program, 2. Download a TALRIK program in Motorola S19 file format into the robot, 3. Do both a compile and load of a TALRIK program into the robot, respectively. Refer to the "talreadme" file for details.

Includetk directory

This directory contains all the header files <file.h> required by TALRIK applications written in C. As you add new applications you will add new header files to this directory.

Libtk directory

This library directory contains the TALRIK library file "libtk.a"

Libsrctk directory

TALRIK library source files.

Pcbug11 directory

This is Motorola's monitor program for the HC11. Also included is the batch file MSCC11.bat, which will automatically load a program into a M68HC11E2 chip. Mrc11.bat is used by loadtk.bat

---

tkcode directory

This MEKATRONIX directory contains a very simple collision avoidance program "avoidtk.c" and the important "xsensetk.c" program. "xsensetk.c" outputs the sensor readings to your PC screen in a TALRIK disk pattern. It is an excellent program for you to familiarize yourself with the robot's sensory capabilites and to verify sensory operation. Source code is provided in C. The S19 files "avoidtk.s19" and "xsensetk.s19" can be downloaded and executed directly, even if you do not have a C compiler.

---

talreadme.txt

A brief description on how to compile and load TALRIK programs.

11.3 File Name: talreadme.txt

Contents
I. Compile a C-Program in DOS
II. Downloading a File using the High Speed Serial Down Loader (HSSDL11)
III. Downloading PCBUG11
IV. Brief Program Descriptions

11.3.1 Compile a C-Program in DOS

1. Open a DOS window and change the directory to c:\icctk\
2. Execute `talsetup.bat` to set up parameters for the compiler and linker.
3. Type the batch command

```
comptk  <program name without .c extension>
```

4. Provided there are no errors, the compiler will place the .s19 file, .lis file, a .lst file, an .mp file, a .s file, and an .o file. If you desire fewer output files you can change the compiler settings. Refer to the ICC11 manual for the definition of these files and how to control their production.

5. With an error free .s19 file you are ready to download.

11.3.2 Downloading a File using the High Speed Serial Down Loader (HSSDL11)

On the TALRIK Robot:

1. Make sure the batteries are connected.
2. Flip the DownLoad/Run switch to DownLoad.
3. Flip the PowerOn/PowerSave switch to PowerOn.

The red PowerOn LED must light. If not, check power connections.

4. Press red Reset button switch.

Diode D2 next to the edge of the Mekatronix MB2325 board will flash on each time you press the reset button and then go out when your release the

button. If it does not perform this way, unplug the 6-wire cable at one end, rotate the plug 180 degrees and replug. When D2 is off and D1 on, the robot is ready for downloading.

In Win95 select HSSDL11 icon and double click. Say 'OK' to enable the program.

5. Settings:
Clock 8MHz
Port COM1

5. Config: Click Config. button or type C.
Check only High-Speed option. Leave all other items unchecked.

Some older computers will not allow you to communicate at 115KBaud and you may have to "unclick this option". The baud rate then defaults to 9600.

7. Click Config again to get back to HSSDL main window.

8. Select Program and browse to find and open c:\icctk\tkcode. Make file type selection the Motorola .s19 files, then, only these files will appear in the program window. Select an .s19 file with a click and then select "Open". The HSSDL11 returns to its main window. The program you selected is featured in the top bar of the HSSDL11 window.

9. Make sure that you have performed steps 1 through 4.

10. Press D or click on the Download button (Diode D2 must be off and D1 on). The HSSDL11 first loads the bootloader (1200 baud) and then your program (115Kbaud).

11. If the Download fails to load the bootloader, it probably means that the serial cable is plugged in wrong, not connected, or you forgot to press reset (diode D2 must be off and D1 on).

If the Download prints 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.

You cannot use PCBUG11 and the HSSDL11 together (and you shouldn't want to!). PCBUG11 is an old program that locks the COM port. If you have used PCBUG11 during a session, you must restart your computer before you can use the HSSDL11.

After sucessful download

On the TALRIK Robot:

CAUTION! Before doing the next step, make sure TALRIK has a safe place to move. Place TALRIK on the floor or suspend the wheels so they do not contact any surface.

12. Flip the DownLoad/Run switch to Run.
13. Press Reset -- TALRIK IS ALIVE!!!!

11.3.3 Downloading using PCBUG11

Open a DOS window, or startup in DOS.

On the TALRIK Robot:

1. Make sure the batteries are connected.
2. Flip the DownLoad/Run switch to DownLoad.
3. Flip the PowerOn/PowerSave switch to PowerOn.

The red PowerOn LED must light. If not, check power connections.

4. Press red Reset button switch.

Diode D2 next to the edge of the Mekatronix MB2325 board will light each time you press the reset button. If it does not light, unplug the 6-wire cable at one end, rotate the plug 180 degrees and replug.

In DOS

5. Change the directory to the PCBUG11 directory.

6. Type the following

mrc11 [path]avoidtk.s19

Note: use xsensetk.s19 for displaying sensory readings.

If the download fails, repeat steps 4 and 6.

After successful download

CAUTION! Before doing the next step, make sure TALRIK has a safe place

to move. Place TALRIK on the floor or suspend the wheels so they do not contact any surface.

7. Flip the DownLoad/Run switch to Run.
9. Press Reset -- TALRIK IS ALIVE!!!!

11.4 Brief Descriptions of TALRIK programs in directory tkcode

The ".s19" files for the programs described below can be loaded directly into your robot and executed. You can find further descriptions of the files in the ".c" versions. To really understand the details of each program, however, read the ".c" code itself!

`avoidtk.c`

1. Back bumper must be pressed to start robot.
2. Performs obstacle avoidance.
3. If forward bumpers are pressed during obstacle avoidance mode, Talrik backs up, turns a random amount of time and then goes forward again.

`avoid2tk.c`

1. Back bumper must be pressed to start robot.
2. Performs obstacle avoidance.
3. If forward bumpers are pressed during obstacle avoidance mode, Talrik moves his head to face the direction of the bumper press and then backs up, turns a random amount of time and then goes forward again.

taltst.c

This program allows you to test all of TALRIK's basic features:

IR Detectors
CdS photoresistors
Motors
Servos

Select the appropriate test from the startup menu. When finished, press reset on TALRIK and select the next test.

`Xsensetk.c`

This module reads all the sensors and updates the appropriate variables and displays them on the screen in a pattern similar to their geometric position on the TALRIK plate.

Xserv1tk.c

Press and hold the rear bumper down and TALRIK's Head servo rotates in one direction. Press and hold the front bumper down and the Head rotates in the other direction. The PWM pulse width (1000<PW<5000) is output to the terminal. By momentarily pressing the bumpers you can step the PWM pulse width by a fixed number of processor cycle increments in either direction.

NOTE: SOME SERVOS HAVE INVERTED CONTROL AND WILL MOVE IN THE OPPOSITE DIRECTION OF OTHER SERVOS. YOU CAN CORRECT FOR THIS IN THE SOFTWARE.

Xserv2tk.c

Press anywhere on the front bumper. TALRIK will turn his head in the direction of the bump.

NOTE: SOME SERVOS HAVE INVERTED CONTROL AND WILL MOVE IN THE OPPOSITE DIRECTION OF OTHER SERVOS. YOU CAN CORRECT FOR THIS IN THE SOFTWARE.

11.5 TALRIK Library Code in directory Libsrctk

Before using any of the TALRIK library functions the appropriate initialization programs must be executed. The init_<program> for the files below are:

```
  init_analog();
  init_clocktk();
  init_motortk();
/* No init routine in muxsentk.c */
  init_serial();
  init_servos();
/* No init routine in Vectors.c */
```

The TALRIK library functions contain these programs. A brief description of the TALRIK library functions appears below.

`analog.c`

Defines init_analog() and analog() functions. analog() reads the analog to digital converter output on ADR1.

`clocktk.c`

Generates msec, seconds, minutes, hours, days. Also, variable timertk is a msec free running counter which can be used. The function wait() permits your programs to execute time waits.

`motortk.c`

Drivers for TALRIK motors.

`muxsentk.c`

Defines all the symbols used with the TALRIK sensors, sensor name aliases, and sensor multiplexor select field values. Provides all the sensor read functions: read_IR, read_CDS, front_bumper(), rear_bumper(), etc. Refer to the file in directory "Libsrctk" or to muxsentk.h in the "Include" directory.

`serialtk.c`

Defines the Mekatronix serial functions. Use <stdio.h> whenever possible, but the function read_int() here is quite useful. Also, executing init_serial() will initialize appropriately for <stdio.h> as well.

`servotk.c`

Drivers for Servo_1 and Servo_2 on the MRSX01 board.

`Vectors.c`

Defines all the MC68HC11 interrupt vectors generically. User and future Mekatronix interrupt service routine names must equate to the predefined generic names.

11.6 talsetup.bat file and the Talrik Library Libtk.a

This file defines the paths to the Include and Lib directories as well as establish the necessary linker options. Observe that the code begins at 0x8000 and the stack at 0x7fff. Note that the MC68HC11 stack grows toward lower memory addresses. If you have a stack that fills up for whatever reason, the processor registers will be affected. Such an errant program could cause the robot to do nothing at best and really weird behavior and possible motor damage at worst. So be careful in programming and ready to turn the robot off immediately if it seems to oscillating rapidly and out of control.

The library option `-ltk` is essential and refers to `Libtk.a`. The library facility of ICC11 compiler versions 5.0 and greater is not compatible with the same facility of versions less than 5.0. A safe rule is that an ICC11 compiler will compile TALRIK code if `Libtk.a` has been compiled on that same version.

If you have the DOS version of the compiler, you can remake the library `Libtk.a`. with `Libtkmake.bat`.

Contents of `talsetup.bat`:

```
doskey
PATH %PATH%;C:\ICCTK\bin
set ICC_INCLUDE=C:\ICCTK\include
set ICC_LIB=C:\ICCTK\lib
set ICC11_LINKER_OPTS=-btext:0x8000 -ltk -dinit_sp:0x7fff -dheap_size:0
```

11.7 Generation of TALRIK's Library Libtk.a with DOS Script Libtkmake.bat

IMPORTANT!

If Libtk.a has been corrupted or has not been compiled with your version of the compiler, then you must recompile the programs in the directory Libsrctk and remake Libtk.a. You will not be able to compile any TALRIK code until this is done.

For the DOS version use the batch file **`Libtkmake.bat`** (listed below). For the Windows IDE version, duplicate the process indicated by the batch file.

Contents of `Libtkmake.bat`:

```
echo off
echo This script file makes a library file for the TALRIK Routines
echo Title              Make TALRIK Libraries
echo Programmer             Keith L. Doty
echo Date               July 28, 1998

icc11 -c vectors.c
icc11 -c serialtk.c
icc11 -c motortk.c
icc11 -c clocktk.c
icc11 -c analog.c
icc11 -c muxsentk.c
icc11 -c -e servotk.c

del ..\lib\libtk.a

ilib -a libtk.a  serialtk.o
ilib -a libtk.a  motortk.o
ilib -a libtk.a  clocktk.o
```

```
ilib -a libtk.a  analog.o
ilib -a libtk.a  muxsentk.o
ilib -a libtk.a  vectors.o
ilib -a libtk.a  servotk.o

ilib -t libtk.a
copy libtk.a ..\lib\*.*
```

12. TALRIK^{II}™ PLAY

To get TALRIK^{II}™ going immediately, we have provided six programs for you to use and expand upon. Follow the downloading procedures explained earlier. You potentially have three choices for downloading

1. HSSDL11 (preferred),
2. IDE *Terminal* program,
3. PCBUG11, least desirable to load <*filename*>*.s19* files.

The files in ICC11 have the suffixes in Table 6. Refer to the ICC11 manual for other suffixes.

Table 6 ICC11 file suffixes.

<filename>.c	C source code file
<filename>.o	C object code file
<filename>.s	MC68HC11 Assembly source code file
<filename>.s19	Motorola S19 object file input to MC68HC11 ROM boot loader.

12.1.1 Checking out TALRIK with `taltst.s19`

The first program you may want to load on TALRIK^{II}™ is our test program *`taltst.s19`*. This program tests all of TALRIK's features and uses serial IO. TALRIK can communicate with IDE *Terminal* or with *HyperTerminal* in you WIN95 environment or with some other terminal simulator that simulates a VT100 terminal. Run the terminal simulator with the following settings:

baud=9600,	8 data bits,	1 stop bit,	Parity = none,	Flow control = none

Read *`taltst.c`* for more details on what the TALRIK Test program does. The program displays a menu on your terminal screen and provides three options, 1) display the sensors, 2) sweep the servos or 3) run the motors. The first choice causes the robot to display all of TALRIK's sensor values on the screen. The second choice will move sweep the servos back and forth. The last choice will run the motors through a sequence of speeds between full backward and full forward.

The sensors are displayed in rough proportion to their geometric position on the TALRIK. This helps you to visually identify the sensor under test. Press a front bumper switch and you will get a front bumper reading. Press a back one and you will observe the back bumper data change. Try

different combination of bumper switch closures. Notice the back bumper does not differentiate points of bumping with the same precision as the front bumper. Place your hand about a foot away from an IR detector and move slowly towards the robot. Notice the changes in the IR readings. Cover a CdS photoresistors with a black object and observe the change in displayed value. Point the robot in different directions and notice how sensitive the CdS photoresistors are.

12.1.2 Get TALRIK Moving

Load *avoidtk.s19*. This program demonstrates the IR collision avoidance and bumpers. Although the collision avoidance algorithm in *avoidtk.c* is simple, TALRIK^{II}™ will avoid most large obstacles. For smaller objects, such as chair legs, the front bumper detects the collision. The IR emitters underneath TALRIK's top plate emit considerable amounts of light. This will make the robot "shy" in crowded environments. If you crowd TALRIK with obstacles all about him, he will just spin around in a circle trying, but unable, to get away. You can make him less shy, by making a copy of this program and changing the threshold for turning away from an object.

12.1.3 Optional Ranging Program (Not Included)

A servo, mounted at one end of the bridge, holds an IR emitter. A servo mounted at the other end of the bridge holds a collimated IR Detector. This system can be used to range out to about 1 meter with 10% accuracy. The program *ranger.c* uses TALRIK^{II}'s™ IR ranging system (a TALRIK^{II}™ option) to collect data from the ranging device. This data can be captured from a serial program to a text file and graphed or analyzed. *ranger.c* allows you to examine real ranging data which may prove useful in the design of algorithms using the *ranger*. See your Mekatronix distributor for details.

12.2 Tips for Writing programs

1. Make sure you have set the linker and compiler options and pathways to `Bin, Lib` and `Include` correctly for you configuration.

2. In your program make sure that include files are referenced like this:

   ```
   #include <mil.h>
   #include <hc11.h>
   etc.
   ```

3. For ICC11 specify the function prototypes at the beginning of the program as shown in the sample code.

4. For IC drop all references to prototypes!!!

5. In ICC11 DOS, compile your program using the batch file: `comptk.bat`

comptk.bat Contents:

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

cd tkcode
icctk  -e -v -l -m %1.c
cd ..\
```

-e C++ comment style accepted (i.e., // comment)
-v verbose (gives details during the compilation)
-l listing
-m map file

You can turn off all these options except –e, since some of Mekatronix code has mixed comment styles. Mix comment styles help when debugging because a // comment can be used to comment out any statement without creating a common bug that involves embedded comments.

You may also want to delete all the `echo` statements and the `pause` statement, especially after you have established everything the way you want it and you have it working.

12.3 Writing your own Interrupt Service Routines for TALRIK using ICC11

You should not attempt to write interrupt service routines unless you have extensive experience with the MC68HC11 and have great familiarity with how interrupts work.

Mekatronix strongly advises you not to change existing interrupt service routines.

Caution!!!

If you change any of the standard TALRIK drivers, Mekatronix applications and system software will not run correctly.

Use the example interrupt service routines provided in the directory `Libsrctk` as guides to constructing your own. Be sure to `#define` your interrupt service routine name equivalent to the Mekatronix predefined names which are given for all possible MC68HC11 interrupt service routines in `Libsrctk/vectors.c`. For those routines that you do construct be sure to enable installation of the interrupt vector by the compiler through a command sequence similar to the C-language construct in `servotk.c`, namely,

```
 #pragma interrupt_handler servo_hand
… … …

//install interrupt handler on OC1
*((void (**)())0xffe8)  = servo_hand;
```

A second, earlier, method which is not as "elegant" , but maybe a little easier to understand, is illustrated by `motortk.c` , first

```
#define motor0 TOC3_isr  /*motor0 is the user name of the interrupt routine*/
#define motor1 TOC2_isr
…
…
…
#pragma interrupt_handler TOC2_isr, TOC3_isr;
```

Next, to prevent `Include/isrtk.h` from destroying the pointers created by the `#pragma`, you must add the lines

```
/* TALRIK II Motor1 handler */
#define TOC2_isr

/* TALRIK II Motor0 handler */
#define TOC3_isr
```

at the beginning of `Include/isrtk.h` . These `#defines` establish the existence of the interrupt service routine names but assign nothing to those names (this has already been done!). The conditional compilations that follow will recognize that the interrupt service routine names already exist and will not create a null function for their interrupt vectors!

13. INTERACTIVE C

Interactive C provides a command line interactive environment for programming the TALRIK robot. First, you most load a PCODE interpreter into the TALRIK memory. The PCODE interpreter is a virtual C-machine that interprets byte-encoded instructions generated by IC from your C-programs. Other than IC translating C into a tightly encoded PCODE form, no compilation takes place. The PCODE virtual machine interprets highly encoded C language statements directly. PCODE programs take less storage than the actual C code. Execution time is slower than compiled code, but, in most cases, this has no robot performance degrading consequences. IC comes in a freeware DOS version and a commercial grade version that runs in WIN95. IC is not standard C, but it comes close enough to make it easy to use. For further *IC DOS* information download the IC manual free from www.mekatronix.com.

Note: The freeware version of IC is not supported by Mekatronix.

13.1 Unplug Servo1 when running IC

Interactive C (IC) does not support 2 servos on the TALRIK. If you have a TALRIK^{II}™ with two servos, disconnect the servo connected to servo1 on the MRSX01 sensor board, the top circuit board in the stack, before proceeding.

WARNING!

A servo plugged into MRSX01 header servo1, will be destroyed if you run IC.

13.2 Getting IC running on TALRIK^{II}™

To get IC to run on TALRIK you must first download the PCODE, particularly, `pcode_rw.s19`, and then execute ic.exe from its root directory.

13.3 Loading PCODE

You can load the PCODE (`pcode_rw.s19`), the C-interpreter portion of IC, using the three methods described earlier for any .s19 file.

1. HSSDL11
2. IDE *Terminal Bootstrap Loader*
3. PCBUG11
   You must boot up in DOS to use PCBUG11 for loading the PCODE. A batch file called `init_tj.bat` will assist in performing this task. After loading PCODE, reboot into WIN95, if you have a WIN95 machine. Executing `ic` from a DOS window in WIN95 is much more convenient.

After loading PCODE, flip *Download/Run* to *Run*, press *Reset* button on the TALRIK and execute `ic.exe`, either the DOS or WINDOWS version. The DOS version can be run in a DOS WINDOW in WIN95.
An IC window pops up in the WINDOWS version and the IC prompt, `C>`, appears in the DOS version. Ask a Mekatronix distributor about sample IC programs and a tutorial manual.

13.4 Writing IC Programs

We have tried to keep the *ICC11* and *IC* programs similar but here are some differences.

1. Motors are controlled in *IC* with motor(0,100.0). In *ICC11* the equivalent function is motor(0,100) (the second number is an interger not a floating point number). *IC* does not require init_motors(), since the motor routine is part of *IC*'s main interpreter system, pcode_rw.
2. As noted above, servo1 header cannot be used with *IC*.
3. *IC* is a multitasking system. In *ICC11* the user would have to write their own multi-tasking system.

14. PROGRAMMING BEHAVIOURS

14.1 Scope

Programming behaviors is what autonomous mobile robots is all about, or, at least a substantial part of what its 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 TALRIK^{II}™ 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 TALRIK^{II}™. But, with one or more MSCC11 single chip computers controlling the additional sensors and servo driven manipulation devices, a TALRIK^{II}™ can often be expanded to meet contest requirements.

14.2 Some Possible Behaviors

TALRIK^{II}™ 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 7.

Table 7 Primitive Behaviors

Collision Avoidance	Collision Detection	Line following	Light following
Light avoidance	Pushing	Collision detection	Shy behavior
Aggressive behavior	Exploring behavior	Wall following	Beacon tracking

With shaft encoders and ranging sensors, TALRIK^{II}™ can develop measurement behaviors (Table 8). These behaviors allow TALRIK^{II}™ to quantify some of its environment. While the measurements taken by these sensors are not precise, TALRIK^{II}™ can be programmed to perform a number of surprising tasks with the information provided.

Table 8 Measurement Behaviors

Relative headings	Range data	Circumnavigating an object
Length of a path	Wheel speed	Mean free path distance

From the primitive and measurement behaviors, one might, with great skill, have TALRIK^{II}™ perform the behaviors in Table 9. These behaviors are still research issues, so if you come up with something that really works well, let the rest of the robot community know about your results!

Table 9 Complex Behaviors

Make maps	Navigate its own map
Learn about its environment	Develop curiosity about unexplored environments

14.3 Advice on Developing Behaviors

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

14.3.1 Vulcan Mind Meld

To effectively program a behavior for TALRIK^{II}™, or, quite possibly, any autonomous robot, and gain insight into the problems you face in developing your concept, you should play Vulcan to the robot and imagine 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 receiving 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.

14.3.2 Virtual Mind Meld©[8]

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.

14.3.3 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

[8] Virtual Mind Meld © Mekatronix™

calibrate themselves in some fixed environment. For example, place a cardboard box in front of TALRIK^{II}™ and 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. The difference in these readings might materially affect some behaviors, so a routine to relative calibrate the responses of the front sensors would provide behavior algorithms a balanced sensor view upon which to act.

14.3.4 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 TALRIK^{II}™ 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.

14.3.5 Create simple behaviors

The beginning robot practitioner usually formulates behaviors too complicated to implement directly. With experience, the virtue of simple, direct behaviors become 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.

14.3.6 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 to align "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!

14.4 Integrating Behaviors

More complex behaviors may require the combination of primitive behaviors in a way not well understood. Neural network activation, 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.

15. TECHNICAL NOTES

This section is to remind the user of not readily apparent features of TALRIK^{II}™.

Disconnect Servo2 when Running IC

When running *IC* one can neither run *servo2* or leave it connected. The connection to *servo2* is on the top printed circuit board, the MrSX01. This must be disconnected if the user chooses to run *IC*.

IR Detector Connectors: Black 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. However, we recommend avoid doing so. Future connectors will be keyed and not permit connecting them incorrectly.

IR Emitters Shine

How can you tell if the IR emitters emit IR? Construct a 300mm (1 foot) 3 wire extension cable for an IR detector, male at one end and female at the other. Dismount an IR detector and place the extension cable between it and the TALRIK^{II} connector. Write your own test program, or execute *tksen.s19* in the TALRIK^{II} directory under ICC11, to determine if the IR detector responds when placed in front of each of TALRIK^{II}'s IR emitters.

CdS Photoresistor Usage

While extremely sensitive over a wide dynamic range, CdS cell characteristics vary widely and two or more cells, typically, do not match. A rule of thumb is to measure the resistance of the cell exposed to nominal lighting conditions for your application and use a resistor of that value in the cell's voltage divider circuit. The robot is supplied with fixed values of resistors which are

socketed for your convenience. You can easily swap in resistor values required for your application.

In programming, try not make the robot behavior dependent upon absolute values of the CdS cell readings, otherwise the code becomes brittle and will easily fail if lighting conditions change. A better approach is to have behaviors based upon differences in values at sequential time samples.

16. TALRIK^{II} TECHNICAL SPECIFICATIONS

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

16.1 Mechanical Structure

1. All of TALRIK™'s body parts are made from beautiful, strong, durable, 1/8 inch, 5-ply, birch model airplane plywood.
2. TALRIK™ fits into a right circular cylinder 10inches in diameter by 10 inches high. (Volume approximately 785 cubic inches or 0.45 cubic feet)

16.2 Power Requirements

1. Eight AA rechargeable Nickel-Cadmium batteries (sold separately) (ENERGIZER™ or EVEREADY™ recommended), 600 ma-hr, 7.2 volts (Discharged) to 9.6 volts (Nominal Charge).

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

2. Recharger, 12 volts D.C. rated greater than or equal to 200ma (Sold separately).

16.3 Actuation

A gearhead DC motor drives each wheel. The characteristics are

1. Battery Voltage input to the motors,
2. 100 -120 ma under load, 80 ma no-load,
3. 1.5 revolutions/sec at 9.6 volts (full battery charge). Speed decreases as the voltage drops.

Pulse Width Modulation (PWM) on PA5 controls the right wheel motor (Motor0) and PWM on PA6 controls the left wheel motor (Motor1). The directions of the motors are controlled by two bits in the MMR register, which is discussed in Section 3.4.

Pulse Width Modulation (PWM) on PA4 controls one servo (SERVO1) and PWM on PA7 controls another servo (SERVO2). The servos have a turning range of approximately 180 degrees. The pan head servo on the TALRIK bridge is connected to SERVO1. You can add another bridge servo without any further circuitry. Simply connect the servo input to the connector. If both servos are driven at the same time while the robot is moving, the

batteries can be pulled down enough by the current surges of changing motor speeds to actually reset the processor. Judicious operation of the motors can avoid this problem.

PA3 can drive a piezo speaker connected to the two pin male header PEIZO.

16.4 MRC11 Robot Controller

The MEKATRONIX Robot Controller, MRC11, enables execution of machine intelligence programs on TALRIK. The MRC11 comes with

1. Motorola MC68HC11 processor,
2. Two 32KB Memory sockets for either RAM or ROM, 64KB total,
3. 5 Volt regulator,
4. Low voltage inhibit reset circuit,
5. Power on LED,
6. 60-Pin Male Header processor / IO bus.

16.5 Sensor Expansion Board

The MRSX01 mates with the MRC11 to provide extensive sensor and control capabilities. The two board stack, MRSX01 and MRC11, furnishes the circuitry that supports all the sensory, motor and cognitive functions for the TALRIK autonomous mobile robot.

The MRSX01 features:

1. Twenty Analog Inputs,
2. Two Digital Inputs,
3. Eight Memory Mapped Digital Outputs
4. Four Memory Addressable Input Device Selects,
5. Three Memory Addressable Output Device Selects,
6. High Memory Select(Address >= b"1111 1111 1011 1xxx"),
7. Two Pulse Width Modulated Outputs for Motor Control,
8. Two Pulse Width Modulated Outputs for Servo Control,
9. One Pulse Width Modulated Output assigned to an optional piezo speaker,
10. A Battery charge circuit (100ma at 12 volts-DC),
11. Battery Voltage Sensor (Analog Input),
12. Charge Current Sensor (Analog Input),
13. Front Bumper Sensor (Analog Input),
14. Rear Bumper Sensor (Analog Input),
15. Forty KHz square wave generator,
16. Processor Data Bus and Ports brought out to a 60 pin Header
17. Battery-Power-In and Battery-Power-Out Headers

Fourteen analog inputs lead out to 3-pin male headers: (pin-1,pin-2, pin-3)=(Signal, 5 Volts, Ground). TALRIK™ uses 12 of these 14 analog inputs to sense IR detector signals. The other two can be used for additional IR or other analog sensors requiring 3-pin connections. Six other

analog inputs terminate on 2-pin male headers:(Signal, 5Volts). TALRIK™ employs these 6 analog inputs for light detection with photoresistors.

16.6 Basic Sensor Suite

TALRIK senses its environment with

1. Twelve IR Emitters, wavelength equals 940nm,
2. Twelve IR Detectors for 40KHz modulated 940nm IR,
3. Five Front Bumper Momentary Tactile Switches,
4. Five Rear Bumper Momentary Tactile Switches,
5. Six CdS Photoresistors.

The hardware information required for software access of these sensory inputs is provided in Section 3.4.

16.7 Sensor Expansion

Up to two MSCC11 boards (optional, not included in base kit) mount on the bridge. You can establish communications between the MSCC11 and the MRC11 via either the Serial Peripheral Interface or the Serial Communications Interface. Each MSCC11 provides TALRIK™ with complete MC68HC11E2 processor functionality, for example, 8 analog IO channels, 8 digital outputs, 8 digital inputs, 5 pulse-width modulation controls, and four input signal captures among others. This functionality permits extensive sensory expansion.

16.8 Switches and LED Power-On Indicator

1. Reset push button
2. Toggle switch: Download Program and Run Program
3. Toggle switch: Off-On,
4. Power on Indicator on TALRIK™ Body

16.9 System Support Software

TALRIK programs can be written in MC68HC11 Assembly Language, Interactive C(IC), C, or BASIC.

1. Sensor and motor routines provided in C and S19 files.
2. PCBUG11 freeware for downloading Motorola S19 files.
3. Freeware MC68HC11 Assembly Language.

Freeware version of IC and BASIC exist, but they cannot take full advantage of all of TALRIK's capabilities. Commercial versions of IC and BASIC are also available.

Separate purchase of a commercial C compiler is available from MEKATRONIX. A high speed RAM Downloader (115 Kbaud) that bypasses the 9.6Kbaud PCBUG11 download rate, speeds program development considerably and is also available from MEKATRONIX™.

16.10 Applications Software

MEKATRONIX™ provides a C program that allows TALRIK to explore his environment and avoid bumping into things, most of the time! If TALRIK does bump into something, his bumpers tell him and he moves away.

16.11 Serial Communication Hardware

TALRIK's serial communications require logic signals (5 volts and Ground). Downloading your software applications to TALRIK from a PC requires voltage conversion of the RS232C levels to the TALRIK 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.