BASE COMMUNICATOR PROTOCOLS
for ROM Version 1.0
Mike Saari 2/15/84
File: bcspec10.tp2

This document will deal strictly with base communicator (BaseComm or B.C.) protocols. Commands passing between the host computer and Topo are covered in a separate document, the Topo Command Set specification. This specification applies to B.C. ROMS 1.00, 1.02 and 1.03.

All numbers are decimal unless otherwise specified.

IR PROTOCOLS

Packet Format

All messages are sent over the IR link in packets. The packet format is one bit every 256 usec, starting with 5 on's and 1 off (the "start sequence"), then characters coded as 8 data bits (LSB first) followed by 1 parity bit - odd parity. This gives a start sequence time of 1.5 msec, a character time of 2.3 msec and a packet time of 20.0 msec. A high bit is a 5 usec IR pulse, a low bit is no IR pulse sent. All packets in the system are always 8 characters, except for short ACK's which are only a single character. The 8 characters in the packet are ordered as follows:

ch#, proc#, cmd#, d,d,d,d, cksum .

d,d,d,d are 4 characters of data; ch# is the channel number to indicate which device should receive the message; proc# is the on-board process number; and cmd# is the command for the given process. The cksum is calculated by first calculating the twos complement of the sum of all the previous characters, and then setting the low nibble (four low-order bits) to the value D(hex).

All packetizing, retries, and IR error handling is handled by the B.C.

IR Timing Specs

For timing purposes, "packet start" occurs on the first bit of the start sequence. "Packet end" occurs at the parity bit for the last byte transmitted.

-Packet end to packet start "dead time" = 2 msec minimum

-Packet end to response packet start = 50 msec maximum

-Packet end to saywhat? start = 55 msec minimum

As implemented, packet end to saywhat? start is actually 60 msec. Also, if the B.C. receives a valid start sequence and a valid ch# (which also indicates the length of the forthcoming message) before the 60 msec timeout, then the B.C. will wait until the expected end of the message before sending a saywhat? - even if the total wait time exceeds the 60 msec timeout time.

ACK0,ACK1, and Saywhat Protocol

Every message received by any given Topo (except for public channels), is acknowledged by an ACK0 or an ACK1. These must alternate to ensure valid transmissions. The MSB of the channel# being transmitted by the B.C. indicates whether Topo should send an ACK0 or ACK1 - 0 means ACK0, 1 means ACK1. Similarly, the return ACK is indicated by the MSB of the returning channel#. The B.C. is required to keep track of the current ACK0/ACK1 status for each private channel.

A valid ACK should be received by the B.C. within 50 ms of the end of the message being sent. If no ACK is received within that time, or a garbled transmission or the wrong ACK is detected, then a "saywhat?" command (see CONFIGURE BASECOMM) is given, telling the Topo to repeat its last ACK. Receiving the correct ACK means to continue normally, i.e. the original message has been correctly handled. Receiving the wrong ACK means the original message was mishandled and should be resent. Receiving nothing or a garbled reply should prompt another "saywhat?". After 5 or more saywhat's without a valid reply, the appropriate QUERY error flag (Topo not responding) is set while the saywhat's continue. The error flag will only be reset after the message is eventually received, or by doing a RESTART BASECOMM.

If Topo is ever unable to determine the answer to a status request within the ACK timeout time, it simply sends the appropriate ACK with no message. The host computer may then deal with the error of "message not received". This timeout handler could change with future product revisions.

When Topo sends a message to the B.C. (generally a status requested by the host computer), the format is:

B.C. channel #,
0 (don't care)
0 (don't care)
d,d,d,d,
cksum.

For a short ACK, the message is a single character, simply:

short ACK channel #

IR Carrier

A carrier signal is transmitted on regular intervals, so that their absence can tell any Topos if they go out of range of the B.C. The B.C. is required to send out some command at least once every 300 msecs. If no command has been sent after 300 msecs has elapsed, the B.C. must send out a dummy carrier message (see CONFIGURE BASECOMM). Any Topo which hears no valid commands for 1.28 seconds will PARK (abort the current motion command and flush the motion command queue). This 1.28 timeout value can be changed - see Topo Command Set spec.

HOST/B.C. PROTOCOLS

8-bit Data and Nibbling

All data in the system, whether on the host or onboard Topo, consists of full 8-bit words. There is a problem, however, in that most serial cards deal with 7-bit ASCII, and also trap certain control codes. To avoid this, all data sent between the host and B.C. will be broken up into 4-bit nibbles, sent through the serial card, and then reassembled. (The full 8-bits are sent over the IR.) The 4-bit nibbles are encoded as ASCII 0-9 and A-F. For example, data 10100011 would be sent as A3 (hex 41, hex 33). A letter G (hex 47) would be sent as 47 (hex 34, hex 37).

Host/B.C. Command Syntax

Handshaking commands between the host and B.C. are not broken up into nibbles. All commands and responses are standard ASCII characters, excluding 0-F and excluding all control codes. The actual range of values is hex 50-7F.

Host/B.C. Serial Bus Communication

Although the baud rates are setable on both the serial card and the B.C., the standard configuration is: 9600 baud, no parity, 1 start bit, 8 data bits, and 1 stop bit. The baud rate is set on the B.C. with four dip switches. The following options are available:

	4	3	2	1
9600 baud -	on,	on,	on,	on
4800 baud -	on,	on,	on,	off
2400 baud -	on,	on,	off,	on
1200 baud -	on,	on,	off,	off
19.2 Kbaud-	on,	off,	on,	on

Any other combination will default to 9600 baud.

Host/B.C. Handshaking

A partial handshaking system is used between the host and the B.C. Before sending any message or command to the B.C., the host should first send the command:

QUERY (ASCII Q).

The B.C. must respond with one of several answers to indicate readiness to receive the message. The required response time is within 3 ms + 1 character time (dependent on the baud rate). (This protocol will run at 9600 baud, which gives a character time of 1 ms.) The possible responses are:

Bit 3 - BUSY - host may not send more data yet
2 - MESSAGE WAITING (from Topo to host)
1 - ERROR - Topo not responding
0 - ERROR - invalid message from host to B.C.
all off - READY - OK to send

The response is a single character, where the indicated bit is active. The four high-order bits are always 1110, so responses are E0-EF (ASCII lowercase blank,a,b...m,n,o). Thus MESSAGE WAITING is 11100100, or hex E4, and BUSY is 11101000 or hex E8 .

The B.C. may interrupt a long message coming in by sending an INTERRUPT (ASCII U) back to the host, which must then stop sending and QUERY until a READY is received. (The host always checks before sending any character to see if an INTERRUPT has been received.)

(The B.C. will accept data even if MESSAGE WAITING is set. However, the host will generally read the message first.)

Host/B.C. Command Formats

The standard message format from host to B.C. is:

MESSAGE START (ASCII S)
Topo process number (nibbled)
Topo command number (nibbled)
d,d,d ... d (nibbled)
MESSAGE END (ASCII Z).

This is used for every message which is to be sent from the host to Topo. If the B.C. detects an extra MESSAGE START, an extra MESSAGE END, a MESSAGE END before proc# and cmd# have been received, a framing error or other serial error, or a MESSAGE END after only half of a character (one nibble) has been sent, then the invalid message error code is set for the next query. Remaining characters at the end of a message, which fall short of filling a packet, are sent out in a final packet padded with trailing spaces (hex 20). If the data for a message spans more than one packet, then the Topo process number and command number will be duplicated for each packet.

In the middle of a message the only legal B.C. commands are QUERY and RESTART BASECOMM, otherwise just data and END-OF-MESSAGE (after READY has been received.) In fact, whenever an INTERRUPT or BUSY has been received, the only legal commands are QUERY and RESTART BASECOMM.

GET LAST RESPONSE (ASCII R) will cause the B.C. to return (0,0,d,d,d,d) (in nibbles) that was last received back from Topo. The 0's are don't care characters returned with the packet. They may be used in the future as a response identifier, or for other information.

CONFIGURE PACKET (ASCII P) tells the B.C. the channel # for further packets, and a public/private flag to indicate whether to expect a return ACK. When the public flag is set, no ACK's are expected. The format is:

CONFIGURE PACKET (ASCII P)
20 channel # (nibbled)
00 public flag - 1=public, 0=private (nibbled)

The left column shows the power-on default values (in hex).

The response is a single character, where the indicated bit is active. The four high-order bits are always 1110, so responses are E0-EF (ASCII lowercase blank,a,b...m,n,o). Thus MESSAGE WAITING is 11100100, or hex E4, and BUSY is 11101000 or hex E8 .

The B.C. may interrupt a long message coming in by sending an INTERRUPT (ASCII U) back to the host, which must then stop sending and QUERY until a READY is received. (The host always checks before sending any character to see if an INTERRUPT has been received.)

(The B.C. will accept data even if MESSAGE WAITING is set. However, the host will generally read the message first.)

Host/B.C. Command Formats

The standard message format from host to B.C. is:

MESSAGE START (ASCII S)
Topo process number (nibbled)
Topo command number (nibbled)
d,d,d ... d (nibbled)
MESSAGE END (ASCII Z).

This is used for every message which is to be sent from the host to Topo. If the B.C. detects an extra MESSAGE START, an extra MESSAGE END, a MESSAGE END before proc# and cmd# have been received, a framing error or other serial error, or a MESSAGE END after only half of a character (one nibble) has been sent, then the invalid message error code is set for the next query. Remaining characters at the end of a message, which fall short of filling a packet, are sent out in a final packet padded with trailing spaces (hex 20). If the data for a message spans more than one packet, then the Topo process number and command number will be duplicated for each packet.

In the middle of a message the only legal B.C. commands are QUERY and RESTART BASECOMM, otherwise just data and END-OF-MESSAGE (after READY has been received.) In fact, whenever an INTERRUPT or BUSY has been received, the only legal commands are QUERY and RESTART BASECOMM.

GET LAST RESPONSE (ASCII R) will cause the B.C. to return (0,0,d,d,d,d) (in nibbles) that was last received back from Topo. The 0's are don't care characters returned with the packet. They may be used in the future as a response identifier, or for other information.

CONFIGURE PACKET (ASCII P) tells the B.C. the channel # for further packets, and a public/private flag to indicate whether to expect a return ACK. When the public flag is set, no ACK's are expected. The format is:

CONFIGURE PACKET (ASCII P)
20 channel # (nibbled)
00 public flag - 1=public, 0=private (nibbled)

The left column shows the power-on default values (in hex).

RESTART BASECOMM (ASCII X) cancels the current packet being attempted and resets all four QUERY status flags. Channel settings and other parameters are not affected. The restart occurs immediately (version 1.02 only had a delayed restart).

CONFIGURE BASECOMM (ASCII Y) tells the B.C. its own IR channel #'s, the content of the carrier dummy message, and the content of the "saywhat" message. The format is:

CONFIGURE BASECOMM (ASCII Y)
10 long ACK B.C. channel # (nibbled)
0F short ACK B.C. channel # (nibbled)
1F carrier null channel # (nibbled)
50 carrier proc# - don't care (nibbled)
41 carrier cmd# - don't care (nibbled)
82 "saywhat" proc# (nibbled)
FF "saywhat" cmd# (nibbled)

The left column shows the power-on default values (in hex).

GET BASECOMM REVISION (ASCII V) causes the B.C. to return two don't care bytes, then four bytes (in nibbles) of the B.C.'s version number and prom number, i.e. (0,0,V.V,P.P). Each byte has values 0-99. The third byte is integral version, the fourth is fractional version, i.e. 1.00, 2.05, 99.99 . The fifth and sixth bytes are integral and fractional prom numbers. The B.C. has the same number for both version number and revision number, i.e. 1.00, 1.02, etc.

Handshake Command Summary

P - configure Packet. Takes (ch#, public flag) in nibbles.
Q - Query basecomm. Returns E0-EF (four status bits).
R - get last Response. Returns (0,0,d,d,d,d) in nibbles.
S - Start message.
U - interrUpt character to host.
V - get basecomm reVision. Returns (0,0,V.V,P.P) in nibbles.
X - restart basecomm.
Y - configure basecomm. Takes (B.C. ch#, short ACK#, carrier ch#, carrier proc#, carrier cmd#, saywhat proc#, saywhat cmd#) in nibbles.
Z - message end.

Status Bit Values

Bit 3 - BUSY - host computer may not send more data yet
2 - MESSAGE WAITING (from Topo to host computer)
1 - ERROR - Topo not responding
0 - ERROR - invalid message from host computer to B.C.
all off - READY - OK to send

TOBS PROTOCOLS Version 1.0
Mike Saari and
Ed Wischmeyer 2/6/84
File: tobs.tp2

TOBS (Topo Onboard Bus System) is a four-wire serial communication bus, controlled by the TopoComm board, the bus master. Connected to the bus master by TOBS are any number of slave boards. The bus master broadcasts, and allows slaves to broadcast, messages to which all boards have access.

All values are in decimal unless otherwise specified.

Message Specifications and Formats

The message format specifications per se are simple: each message is a 7 byte long packet, and the first byte is the destination process of the message. However, all messages to date conform to one of two formats.

The most common format is:

process#, command#, spare, data1, data2, data3, data4

This means that the given process should execute the given command with as many of the 4 data bytes as required. All other processes should ignore this message. (There is also an "all call" process number (0) to which most processes should respond. See the Topo Command Set spec. for more details on process and command assignments.)

The other currently used format is for status messages being relayed from TOBS via the bus master to the IR system for transmission as IR messages. The format for these return messages is:

IR channel#, dest. process#, dest. command#, data1-4

In the current implementation, the only valid IR destination (the base communicator) does not use destination process# or command#, so this return format is equivalent to:

IR channel#, spare, spare, data1, data2, data3, data4

All status requests to date use the first three data values of the requesting message to specify the "return address" for the response. In other words, the first three bytes (the destination) of the return message are given by the first three data bytes of the requesting message. Thus, a status request of the form

process#, command#(request), spare, X, Y, Z, spare

would be answered with a message of the form

X, Y, Z, data1, data2, data3, data4

For offboard requests, X is then the IR return channel. For onboard requests, X is the return process# and Y is the return command#. Note that in all message formats currently used, the last 4 bytes are always data.

Timing and Signal Characteristics

All messages transmitted over TOBS are exactly 7 bytes long. Each byte sent has 11 bits: a start bit (0), 8 data bits (LSB first), a programmable ninth data bit, and a stop bit (1). The ninth bit is set only in the first byte in each message, and indicates message start. The ninth bit is cleared for all other bytes. The baud rate is 12 MHz (system clock) / (16 X 36), or 20.833 Kbaud, giving one byte in 528 usec. This is set up by configuring the 8031 serial interface in Mode 3, SMOD = 1, timer 1 in mode 2 with reload value of FD hex. At present, bytes are sent on 256 usec intervals, so one byte is actually sent every 768 usec. For standard 7-byte messages, this gives a transmit time of 5376 usec/message. 4 extra cycles are allowed as margin in the system, for a standard message time of 6400 usec.

TOBS Wiring

Two of the wires on TOBS are for serial data per se - bus master transmit (for sending data from master to slave), and bus master receive (for sending data from slave to master). In addition, logic is supplied on the bus master board allowing it to tie the transmit and receive lines together. The bus master always ties the send and receive lines together whenever it grants a bus request. This allows one slave to send data directly to all other slaves.

The two TOBS control lines are named priority_OK_bar and bus_busy_bar. Both are active low. With these lines high, the state is referred to as priority not OK and bus not busy. Bus busy is a single wire common to the bus master and all slaves. It is configured as a wire-OR, normally pulled high by a resistor (bus not busy), but any board can pull it low into the bus busy condition. Priority OK is daisy-chained starting with the bus master at the "top", and lower-priority slaves in series "below" it (see Figure 1). Any board may set priority not OK, but only the boards "below" it are affected. Every slave board has a bus request line (active high) which it uses to request a TOBS access. Activating bus request does two things: it will activate bus busy, and it will set priority not OK for all "lower" slaves boards.

How it Works

A slave board that wishes to send data over TOBS must first wait for a bus not busy condition (see Figure 2). (Priority will always be not OK during bus not busy.) The slave then asserts bus request (a), which causes a bus busy condition on all boards (and also forces priority not OK for all lower boards). The bus master, seeing a request (by bus not busy becoming bus busy), grants the bus by setting priority OK (b). The bus master also sets bus busy, so now both master and slave are asserting bus busy.

The slave, seeing priority OK, transmits its message. Then the slave releases bus request (c), but bus remains held busy by the bus master, so no further accesses are allowed. One standard message time (6400 usec) after granting the bus, the bus master disables TOBS by setting priority not OK (d). (This may occur sometime after the timeout time, if the communication controller board is busy elsewhere.) After another delay of 15 cycles, or 3840 usec, the bus master releases bus busy (e), and the TOBS is free to accept another request. (This 15 cycles is in the present implementation to insure that boards have a chance to process received TOBS messages.) Thus, the total time for consecutive messages is 40 cycles, or 10240 usec (10.24 msec), plus any delay from one message to the next.

The slave should check priority OK before sending every byte. If it ever goes not OK (say if the slave takes too long to send its message, although this should never happen), the slave must immediately relinquish the bus.

If two slaves both wish to send a message, the first one to request will set bus busy, locking out the other slave. If a simultaneous request occurs, priority OK will only make it to the higher priority slave, since its bus request will not allow priority OK for the lower slave. The lower slave, after waiting and not receiving priority OK within the prescribed time of 512 usec, must release its bus request and try again later.

When the bus master board wants to use the bus, it must first wait for any message in progress to complete. It then sets priority not OK and bus busy, and sends its message. When done, it sets bus not busy.

Since the bus master cannot monitor the TOBS bus while handling speech data (since the serial port used for TOBS must be reset to the speech board baud rate), the bus master will disable TOBS while passing data to the speech board.

Other

There is a fifth line routed with the TOBS bus, a system reset line. This is asserted by the communication controller board, and performs a power-on reset of all slave boards.

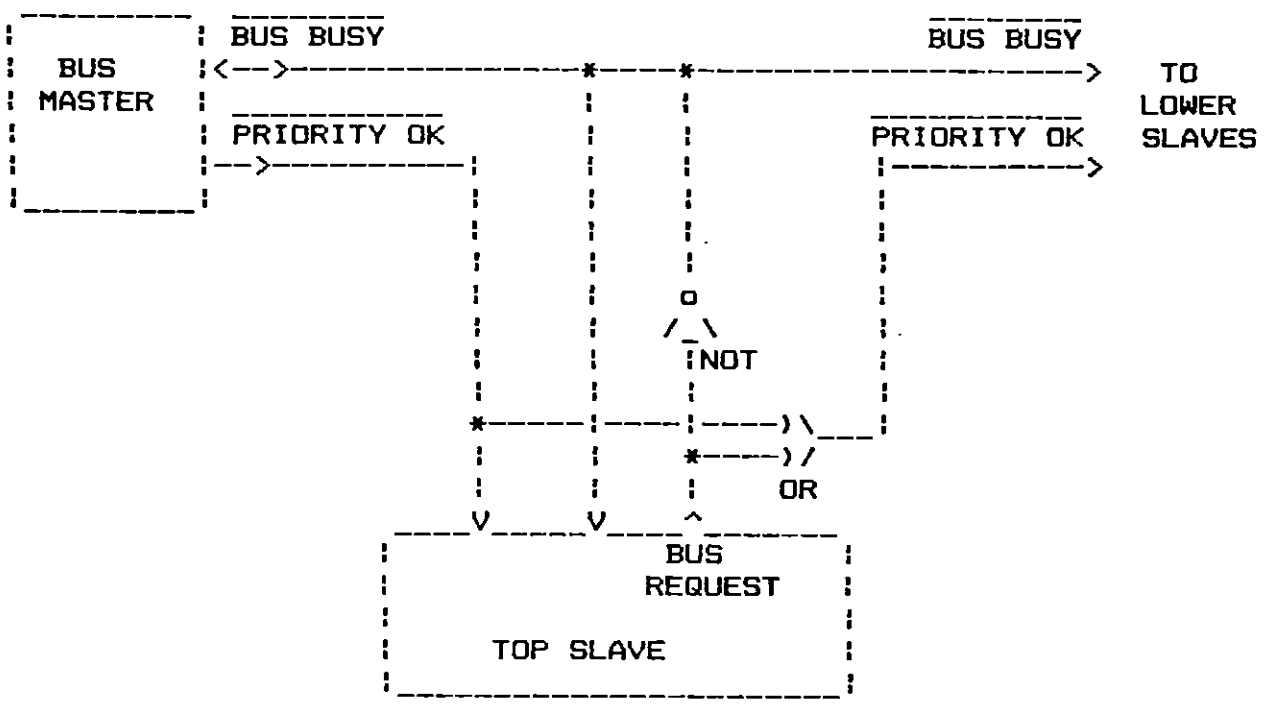


Figure 1.
TOBS Control Structure

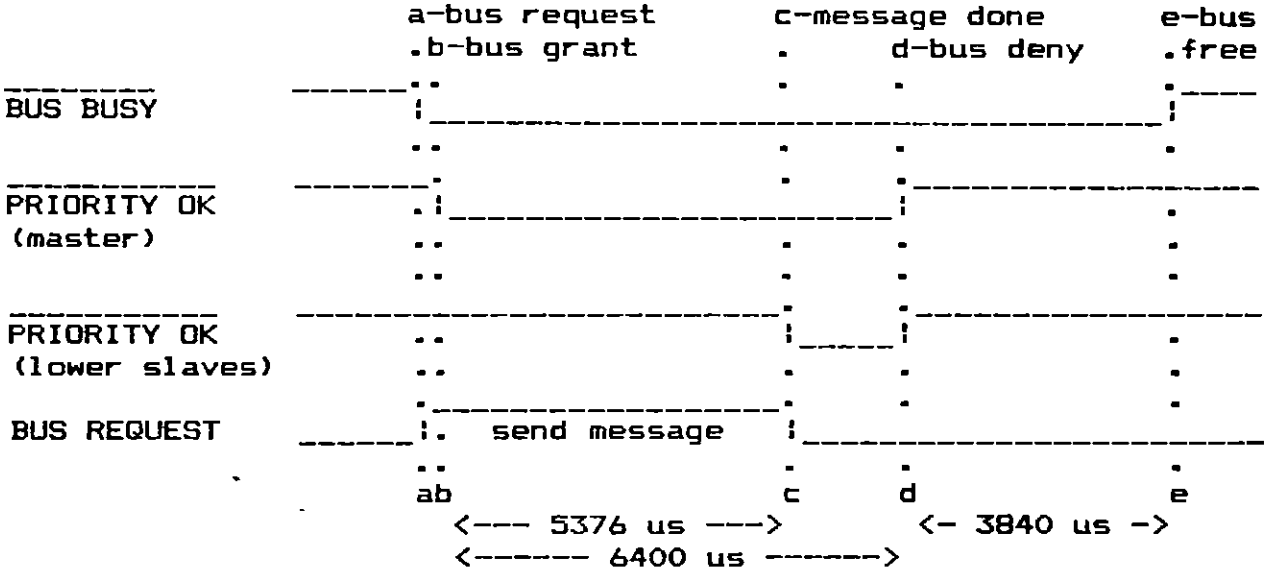


Figure 2.
TOBS Timing

1 of 2

Nolan bumpswitch words

Mike Saari
12-15-83

```
: GET-BUMPSWITCH ( ; → word )
(word is bits 0-15, where bits 0-13 are one bit for each of)
(14 bump sensors - 1 = deflected, 0 = off)
Proc# switches  cmd# reg bumpsw  fetch-2-word-msg  drop ;
   ($ 81)          ($ BE)

: Test-bumpswitch ( word ; position=0-13 → flag)
  2** OVER AND BOOL ;


: 2** ( ; n → 2^n ) (valid for n=0-15 )
  DUP
  IF 1 SWAP 0 DO DUP + LOOP
  ELSE DROP 1
  THEN ;
```

Typical use:

```
Get-bumpswitch
0 Test-bumpswitch IF PARK THEN
1 Test-bumpswitch IF ... THEN
  :
13 Test-bumpswitch IF ... THEN
DROP
```

Nolan - I know this is slightly terse and/or cryptic. Questions - give me a call.
x262 work 272-3952 home, or 972-2053 Mike S

2 of 2

Nolan bumpswitch words

#4 stuck on, #C stuck on

Mike Saari
12-15-83

use the command LOAD-SWITCH to load commands into switch buffers.

LOAD-SWITCH (; SW#, proc#, cmd#, word1, word2)

Examples: ($n means "hex" n)

Park when sw#3 hit (on)

sw# = $83, proc# = $F0 (motion), cmd# = $07 (motion-stop)

word1, word2 = don't care

: $83 $F0 $07 0 0 LOAD-SWITCH

Say "on" when sw#1 hit, say "off" when released

interpret word1, word2 each has 2 bytes, put in ASCII code for each letter (3 max). The fourth byte should be a carriage return ($0D), thus

O	N	CR	blank	O	F	F	CR
4F	4E	0D	20	4F	46	46	0D
$4F4E		$0D20		$4F46		$460D	
word1		word2		word1		word2	

sw# $81 (on), $91 (off)

proc# = $8C (speech), cmd# = $7F (SAY)

: $81 $8C $7F $4F4E $0D20 LOAD-SWITCH

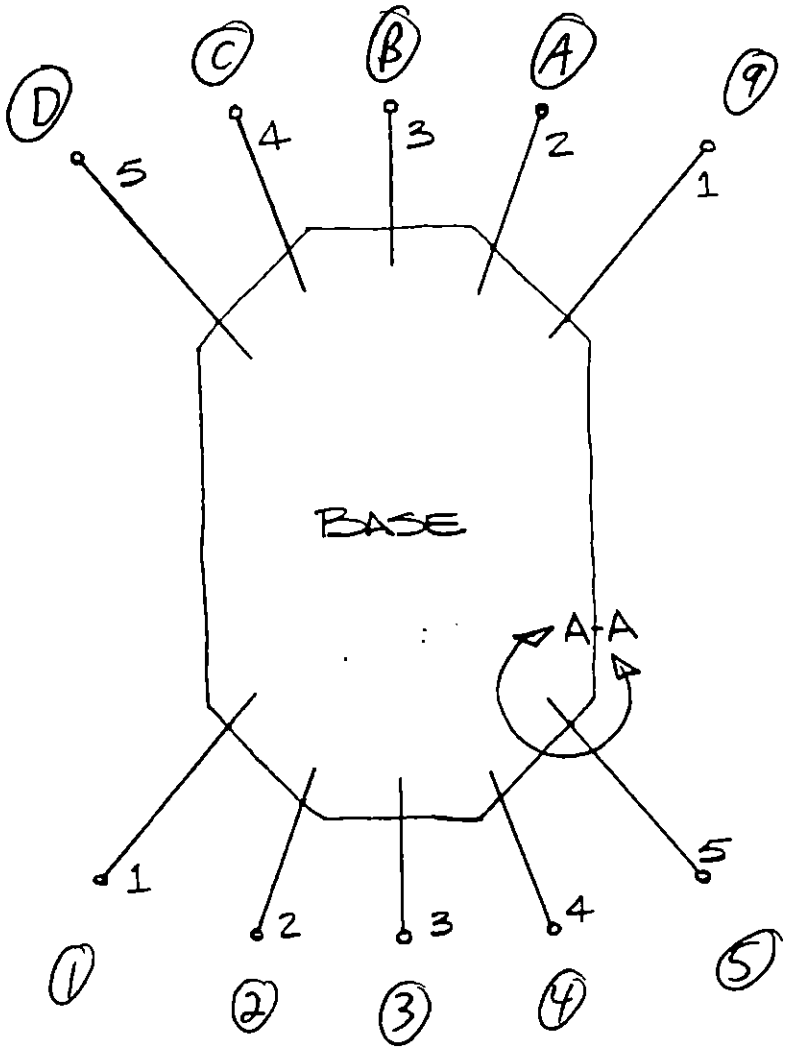
$91 $8C $7F $4F46 $460D LOAD-SWITCH

Do an 45 50 ARC when sw #9 is hit

sw# = $89, proc# = $F0 (motion), cmd# = $5E (ARC), word1 = 45, word2 = 50

: $89 $F0 $5E 45 50 LOAD-SWITCH

BACK



BF1 THRO
BF5

FF 1 THRO
FF5

FRONT

VIEW A-A

14 Dec 83

CURB FEELERS

COMMUNICATION BD.			CURB FEELER
	FRONT ~~BACK~~ CURB FEELERS		
✓ J3-2	PIN 1	81	FBF 1 CT
✓ J3-3			FBF 2 CT
✓ J3-4			FBF 3 CT
✓ J3-5			FBF 4 CT
✓ J3-6	PIN 5	85	FBF 5 CT
✓ J3-11			FBF 5 GS
✓ J3-12			FBF 4 GS
✓ J3-13			FBF 3 GS
✓ J3-14			FBF 2 GS
✓ J3-15			FBF 1 GS
	~~FRONT~~ BACK CURB FEELERS		
✓ J2-2	PIN 1	89	BFF 1 CT
✓ J2-3			BFF 2 CT
✓ J2-4			BFF 3 CT
✓ J2-5			BFF 4 CT
✓ J2-6	PIN 5	8D	BFF 5 CT
✓ J2-11			BFF 5 GS
✓ J2-12			BFF 4 GS
✓ J2-13			BFF 3 GS
✓ J2-14			BFF 2 GS
✓ J2-15			BFF 1 GS

Socket J3 = Part B

J2 = Part C

All values hex

Switch Number Assignments

Head Switches

1-FWD
2-LEFT
3-RIGHT
4-BACK

Bump Switches

0-7
and
8-13 (initial set of 6)

REQUEST-HEADSWITCH returns d1-d2 = 00000000 000dddd0
4321

REQUEST-BUMPSWITCH returns d1-d2 = 00dddddd dddddddd
DCBA98 76543210

LOADBUF commands use switch#'s:
Headswitches = 1-4
Bumpswitches (on) = 80-8D
Bumpswitches (off)= 90-9D

IR Channel Assignments

02	short ACK channel (one character only)
10	host datalink message return channel
20-2F	Topo channels (Topo #'s 0-15 decimal)
7C-7F	public channels P4-P1

Process Assignments

00	ALL CALL
01-7F	(IR return processes)
80	Comm. board null process
81	SWITCHES
82	IR CONTROL
83	(special IR relay, reserved)
84	UTILITY
8C	SPEECH
8D	(additional speech, reserved)
8E	(additional speech, reserved)
8F	(additional speech, reserved)
F0	MOTION
FF	NULL PROCESS

Command Number Summary

00	RESET	universal-no data
01	CANCEL	universal-no data
02	ABORT-REQUEST	universal-no data
03	TOBS-XMIT-OK	universal-no data

All values hex

04	TOBS-XMIT-NOT-OK	universal-no data	
05	SELF-TEST	universal-no data	
06	NO-OPERATION	universal-no data	
07	MOTION-STOP	universal-no data	
08-1F (open)			
20-38 (open)			
39	SET-HEADFOLLOW	switches	d1d2=flag (headfollow?)
3A	SET-SMOOTH	motion	d1d2=flag (smooth?)
3B	GO-TURN	motion	d1d2=turnrate
3C	GO-FWD	motion	d1d2=fwd. vel.
3D	SET-RAMP	motion	d1d2=ramp rate
3E	SET-SPEED	motion	d1d2=speed param.
3F	SET-PRIVATE	IR ctrl	d1d2=ch#
40-5C (open)			
5D	GO	motion	d1d2=turnrate, d3d4=fwd. vel.
5E	ARC	motion	d1d2=angle, d3d4=dist.
5F	SET-PUBLIC	IR ctrl	d1d2=ch#, d3d4=flag
60-7B (open)			
7C	LOADBUF-CHARS6-7	switches	d1=sw.#,d2d3=chars6-7
7D	LOADBUF-CHARS4-5	switches	d1=sw.#,d2d3=chars4-5
7E	LOADBUF-CHARS1-3	switches	d1=sw.#,d2-d4=chars1-3
7F	SAY	speech	d1d2d3d4=speech data
80-9F (open)			
A0	REQUEST-PROCESS#	universal-returns	d1d2=process#
A1	REQUEST-SELFTEST-STATUS	universal-returns	d1d2=result
A2-BC (open)			
BD	REQUEST-MAX-RAMP	motion	returns d1d2=ramp rate
BE	REQUEST-BUMPSWITCH	switches	returns d1d2=bits0-13
BF	REQUEST-HEADSWITCH	switches	returns d1d2=bits1-4
C0	REQUEST-REVISION	universal-returns	d1.d2=process vers.# d3.d4=prom #
C1-D9 (open)			
DA	REQUEST-CHANNEL-SETTINGS	IR ctrl	returns d1d2=default channel d3d4=current channel
DB	REQUEST-MAX-SPEED	motion	returns d1d2=max. turnrate, d3d4=max. velocity
DC	REQUEST-MOTION-QUEUESIZE	motion	returns d1d2=commands pending, d3d4=space remaining
DD	REQUEST-VELOCITY	motion	returns d1d2=turnrate, d3d4=fwd. velocity
DE	REQUEST-POSITION	motion	returns d1d2=current angle, d3d4=current distance
DF	REQUEST-SPEECH-STATUS	speech	returns d1d2=talking flag, d3d4=buffer full flag
E0	REQUEST-TYPE	universal-returns	dddd=process descript.
E1-FE (open)			
FF	SAYWHAT?	IR ctrl	returns repeat of last IR msg.

TOPO COMMAND SET version 1.0
Mike Saari 2/6/84
File: cmdset10.tp2

This document specifies all of the currently assigned channels, processes, and commands, along with their code assignments, being sent between the host computer and Topo. These code assignments should be generally transparent to the base communicator (see Base Communicator Protocols spec.) except for its default values for carrier, ACK's, saywhat's and the initial IR channel#.

This specification applies to motor control ROMs and TopoComm ROMs starting with version 1.0. ROMs which have actually been released to date (to this specification) are motor control ROMs 1.01 and 1.03, and TopoComm ROM 1.0.

(dddd) or (d1d2d3d4) represent four characters (bytes) of 8-bit data being transmitted. Numbers are twos complement unless otherwise specified. All data values generally occupy two bytes (0-FFFF), except where otherwise noted. A flag value of 0 means False; 1 or any other value means True. Counters, such as "characters 1-6", always start at 1 unless otherwise specified. Bits always start at 0, i.e. bits0-7.

All values are in hex unless otherwise specified.

The phrase "implemented but not used" means that the given command, while functional, is not actually called anywhere by TopoSoft or by any other part of the system. They are generally hooks for future product expansion.

Request Format

All REQUEST commands, which require an answer to be returned, are always in the range 80-FF (i.e. high bit is set). This bit is used by the system, specifically by the TopoComm IRctrl process, to indicate that a return answer should be forthcoming. The REQUEST message always includes the (d1d2d3d4) data field, which contains the necessary information to indicate where the reply should be sent. d1d2d3 are used as the first three bytes of the return message over TOBS, and also as the first three bytes of the IR return message. See the TOBS Protocols spec. for more details.

UNIVERSAL Command Rules

Universal commands are recognized by all processes. The same general functionality is implied for any given process, although the details of function and implementation will vary somewhat from process to process. For any case where the particular universal command is not applicable to a particular process, the process should still handle it gracefully, usually by performing a no-operation.

UNIVERSAL Commands

RESET	00-CANCEL the last command(s), and set all parameters to their initial (power-on) values.
CANCEL	01-stop processing of any previous command(s). Implemented but not used.
ABORT-REQUEST	02-cancel any previous status REQUEST(s).
TOBS-XMIT-OK	03-the addressed process(es) may talk to the TOBS. Implemented but not used.
TOBS-XMIT-NOT-OK	04-the addressed process(es) may NOT talk to the TOBS. Implemented but not used.
SELF-TEST	05-perform the internal self-test routine. Not currently implemented.
NO-OPERATION	06-null command. Implemented but not used.
MOTION-STOP	07-if the addressed process is a MOTION process, stop all motion and cancel any buffered commands. Otherwise, do nothing.
REQUEST-PROCESS#	A0-returns (d1d2). d1d2 is the process# of the answering process. This command is useful for determining the presence or absence of any given process, or can be used with an ALL-CALL process to poll all of the processes on the Topo. Implemented but not used.
REQUEST-SELFTEST-STATUS	A1-returns (d1d2). d1d2 (0-FFFF) is the result of a previous SELF-TEST command. 0 = fail, 1 = OK 2 = no test performed, 3 = not completed. (d3d4 are available for various error codes in the future.) Implemented (always returns value 2) but not used.
REQUEST-REVISION	C0-returns (d1.d2,d3.d4). d1.d2 is the version # (0.0 - 99.99 decimal) of the answering process. d3.d4 is the PROM # (0.0 - 99.99) of the answering process. Numbers are coded in BCD, so 98.76 decimal is coded as: 10011000 01110110.
REQUEST-TYPE	E0-returns (d1d2d3d4). d1-d4 is a 4-byte description of the answering process in ASCII. Current descriptions are: switches-SWBS (for headSWitches and BumpSwitches), IR control-IRLD (for IR with LeD's), utility-UTC1, speech-SPEC (for SPeech, ECho, motion-MTN1. Implemented but not used.

Switch Number Assignments

Head Switches

1-FWD
2-LEFT
3-RIGHT
4-BACK

Bump Switches
(Implemented but not used.)

0-13 decimal
(00-0D hex)

LOADBUF commands use switch#'s:

Headswitches = 1-4
Bumpswitches (on) = 80-8D
Bumpswitches (off)= 90-9D

The message buffer for each switch can be loaded with 7 characters which constitute a standard 7-character TOBS message (see TOBS PROTOCOLS spec.). The message is sent over TOBS whenever the particular switch is activated. (Only works for headswitches if SET-HEADFOLLOW is disabled.) Each bumpswitch has two separate message buffers - "on" and "off". "On" is activated when the bumpswitch is triggered, and "off" is activated when it is released.

SWITCH Control Commands

SET-HEADFOLLOW — 39-(d1d2). Enable or disable automatic head-follow, based on flag d1d2 (true means enable). The initial value is true. When enabled, the four switches being pressed result in the following:
1 - Say "forward" and send a "0 50 GO-FOREVER" cmd
2 - Say "left" and send a "-24 GO-TURN" command
3 - Say "right" and send a "24 GO-TURN" command
4 - Say "stop" and send a PARK command (All-Call Motion-Stop). Above speed values are in decimal.
When disabled, the four switches being pressed result in the message contained in the switch buffer being sent (see LOADBUF- commands).

SET-NO-IR — 7B-(d1d2,d3d4). Set Topo's actions after IR time-out has occurred. d1d2 is a flag which, if true, indicates that Topo should automatically revert to headfollow mode on IR timeout. Default is true. d3d4 sets the IR signal loss timeout time X 10ms (decimal). Valid values are 1-255 decimal and the default is 128 decimal.
This command will be superceded next revision.

LOADBUF-CHARS1-3 — 7E-(d1,d2d3d4). Load characters 1-3 of the message buffer associated with the indicated switch. d1 is the switch # (0-FF). d2-d4 are message characters 1-3. Initial values for headswitches are ALL-CALL MOTION-STOP (i.e. PARK). Initial values for bumpswitches contain a message to the null TopoComm. process, i.e. NO-OP.

LOADBUF-CHARS4-5 — 7D-(d1,d2d3). Load characters 4-5 of the message buffer associated with the indicated switch. d1 is the switch #. d2d3 are message characters 4-5.

LOADBUF-CHARS6-7 — 7C-(d1,d2d3). Load characters 6-7 of the message buffer associated with the indicated switch. d1 is the switch #. d2d3 are message characters 6-7.

REQUEST-BUMPSWITCH — BE-returns (d1d2). d1d2 uses 14 (decimal) bits representing 14 flags, one for each potential bumpswitch sensor. Implemented but not used.
d1-d2 = 00dddddd dddddddd
DCBA98 76543210

REQUEST-HEADSWITCH — BF-returns (d1d2). Bits 1-4 of d2 are four flags, one for each latched headswitch value.
d1-d2 = 00000000 000dddd0
4321

MOTION Commands

For all motion commands, positive velocity is forward, positive turns are clockwise.

SET-SMOOTH	3A-(d1d2). Set Topo's motion mode to smooth if the flag d1d2 is true. Set to exact mode (stop after each movement) if the flag is false. The initial value is true.
GO-TURN	3B-(d1d2). Command Topo to turn at the indicated turnrate, but do not alter the current forward velocity. d1d2 is the new turnrate, +/- 0-106 deg/sec (decimal). Used only by headfollow mode.
GO-FWD	3C-(d1d2). Command Topo to move forward at the indicated velocity, but do not alter the current turnrate. d1d2 is the new forward velocity, +/- 0-50 cm/sec (decimal). Implemented but not used.
SET-RAMP	3D-(d1d2). Set the motion ramp rate parameter to the indicated value. d1d2 is the new ramp rate value, 0-100 cm/sec/sec (decimal). Initial value is 10 (decimal).
SET-SPEED	3E-(d1d2). Set the target velocity parameter to the indicated value. d1d2 is the new value, 0-50 cm/sec (decimal). The initial value is 30 (decimal).
GO-FOREVER	5D-(d1d2,d3d4). Directly set the wheel speeds. d1d2 is the turnrate, +/- 0-106 deg/sec (decimal). d3d4 is the forward velocity, +/- 0-50 cm/sec (decimal).
ARC	5E-(d1d2,d3d4). Start a distance movement, using the given values for angle and distance and using previously set values for velocity and ramp rate. d1d2 is the total turn angle, +/- 0-7FFF deg. d3d4 is the distance covered, +/- 0-7FFF cm.
REQUEST-MAX-RAMP	BD-returns (d1d2). Gets the maximum allowed value for the motion ramp rate parameter (0-FF). Implemented but not used.
REQUEST-MAX-SPEED	DB-returns (d1d2,d3d4). Gets the maximum allowed turnrate and velocity. d1d2 is the max. turnrate, current value 106 deg/sec (decimal). d3d4 is the max. velocity, current value 50 deg/sec (decimal). Implemented but not used.

REQUEST-MOTION-QUEUESIZE	DC-returns (d1d2,d3d4). Gets the current status of the motion queue. d1d2 is the count of commands pending, 0-16 (decimal). d3d4 is the count of space remaining, 0-16 (decimal).
REQUEST-VELOCITY	DD-returns (d1d2,d3d4). Gets the current velocity values from the velocity control sequencer, i.e. the current attemped Topo speed. d1d2 is the current turnrate, +/- 0-106 deg/sec (decimal). d3d4 is the current velocity, +/- 0-50 cm/sec (decimal).
REQUEST-POSITION	DE-returns (d1d2,d3d4). Gets the current Topo position, relative to 0,0 at the beginning of the latest motion command. d1d2 is the current angle, +/- 0-7FFF deg. d3d4 is the current distance, +/- 0-7FFF cm.

IR Channel Groupings

01-0F	-	Ack's
10-1F	-	Other channels
20-2F	-	TOPO private channels
30-3F	-	open
40-4F	-	open
50-5F	-	open
60-6F	-	open
70-7F	-	public channels

Process Groupings

0	-	All Call
80-8F	-	TopoComm.
90-9F	-	open
A0-AF	-	open
B0-BF	-	open
C0-CF	-	open
D0-DF	-	open
E0-EF	-	Sensor (reserved)
F0-FE	-	Motion
FF	-	Null Process

Command Number Groupings

Command numbers (values are 0-FF) are grouped into the following classifications:

00-1F	command, no data parameters
20-3F	command, single data parameter
40-5F	command, two data parameters
60-7F	command, special (or multiple data parameters)
80-9F	request, unallocated
A0-BF	request, return single parameter
C0-DF	request, return two parameters
E0-FF	request, special (or return multiple parameters)

Universal commands start numbering from the bottom of any group, i.e. 20, 21, 22, etc. Process-specific commands start numbering from the top of any group, i.e. 3F, 3E, 3D, etc.

IR Channel Assignments

0F	short ACK channel (one character only)
10	base communicator message return channel
1F	null channel (used for carrier)
20-2F	Topo channels (Topo #'s 00-0F, respectively)
7C-7F	public channels P4-P1, respectively

Process Assignments

00	ALL CALL
01-7F	(IR return processes)
80	TopoComm null process (Implemented but not used.)
81	SWITCHES
82	IR CONTROL
83	(special IR relay, reserved but not used.)
84	UTILITY (Implemented but not used.)
8C	SPEECH
F0	MOTION
FF	NULL PROCESS (Implemented but not used.)

Command Number Summary

Cmd #	Name	Process # name	Data Format
00	RESET	universal-no data	
01	CANCEL	universal-no data	
02	ABORT-REQUEST	universal-no data	
03	TOBS-XMIT-OK	universal-no data	
04	TOBS-XMIT-NOT-OK	universal-no data	
05	SELF-TEST	universal-no data	
06	NO-OPERATION	universal-no data	
07	MOTION-STOP	universal-no data	
08-1F (open)			
20-38 (open)			
39	SET-HEADFOLLOW	81 switch	d1d2=flag (headfollow?)
3A	SET-SMOOTH	F0 motion	d1d2=flag (smooth?)
3B	GO-TURN	F0 motion	d1d2=turnrate
3C	GO-FWD	F0 motion	d1d2=fwd. vel.
3D	SET-RAMP	F0 motion	d1d2=ramp rate
3E	SET-SPEED	F0 motion	d1d2=speed param.
3F	SET-PRIVATE	82 IRctrl	d1d2=private ch#
40-5C (open)			
5D	GO-FOREVER	F0 motion	d1d2=turnrate, d3d4=fwd. vel.
5E	ARC	F0 motion	d1d2=angle, d3d4=dist.
5F	SET-PUBLIC	82 IRctrl	d1d2=public ch#, d3d4=flag

Cmd #	Name	Process # name	Data Format
60-7A (open)			
7B	SET-NO-IR	81 switch	d1d2=flag (auto-headfollow?) d3d4=timeout X 10ms (decimal)
** 7B to be superceded		next revision **	
7C	LOADBUF-CHARS6-7	81 switch	d1=sw.#,d2d3=chars6-7
7D	LOADBUF-CHARS4-5	81 switch	d1=sw.#,d2d3=chars4-5
7E	LOADBUF-CHARS1-3	81 switch	d1=sw.#,d2-d4=chars1-3
7F	SAY	8C speech	d1d2d3d4=speech data
80-9F (open)			
A0	REQUEST-PROCESS#	universal-returns	d1d2=process#
A1	REQUEST-SELFTEST-STATUS	universal-returns	d1d2=result
A2-BC (open)			
BD	REQUEST-MAX-RAMP	F0 motion returns	d1d2=max. ramp
BE	REQUEST-BUMPSWITCH	81 switch returns	d1d2=bits00-0D
BF	REQUEST-HEADSWITCH	81 switch returns	d1d2=bits1-4
C0	REQUEST-REVISION	universal-returns	d1.d2=process vers.# d3.d4=prom #
C1-D9 (open)			
DA	REQUEST-CHANNEL-SETTINGS	82 IRctrl returns	d1d2=initial channel d3d4=current channel
DB	REQUEST-MAX-SPEED	F0 motion returns	d1d2=max. turnrate, d3d4=max. velocity
DC	REQUEST-MOTION-QUEUESIZE	F0 motion returns	d1d2=commands pending, d3d4=space remaining
DD	REQUEST-VELOCITY	F0 motion returns	d1d2=turnrate, d3d4=fwd. velocity
DE	REQUEST-POSITION	F0 motion returns	d1d2=current angle, d3d4=current distance
DF	REQUEST-SPEECH-STATUS	8C-speech returns	d1d2=talking flag, d3d4=buffer full flag
E0	REQUEST-TYPE	universal-returns	dddd=process descript.
E1-FE (open)			
FF	SAYWHAT?	82 IRctrl returns	repeat of last IR msg.

APPENDIX A

TOPOSOFT COMMAND SUMMARY

MOTION COMMANDS

n FWD	(dist--)	PARK	(--)
n BACK	(dist--)	JOYSTICK	(--)
n LEFT	(angl--)	TILL-STOPPED	(--)
n RIGHT	(angl--)	MOVE-SMOOTH	(--)
n n ARC	(angl,dist)	MOVE-EXACT	(--)
n n GO-FOREVER	(turnrate,speed--)	GET-POSITION	(--angl,dist)
n SET-SPEED	(speed--)	GET-VELOCITY	(--turnrate,speed)
n SET-RAMP	(ramp--)		

SPEECH COMMANDS

SAY"	(--)	TALK-LEVEL	(--)
PHON"	(--)	TALK-WAVY	(--)
SAY-LATER"	(--)	TALK-FAST	(--)
SAY-IT	(--)	TALK-SLOW	(--)
n SAY#	(number--)		
n SET-PITCH	(pitch--)	SAY-LETTERS	(--)
n SET-VOLUME	(vol--)	SAY-WORDS	(--)
SPEECH-FULL?	(--flag)	SAY-SOME-PUNC	(--)
TALKING?	(--flag)	SAY-MOST-PUNC	(--)
TILL-SILENT	(--)	SAY-ALL-PUNC	(--)

HEADSWITCH COMMANDS

GET-HEADSWITCH	(--bits1234)	ENABLE-HEADFOLLOW	(--)
		DISABLE-HEADFOLLOW	(--)

CHANNEL COMMANDS

n OPEN-CHANNEL	(channel--)	n ENABLE-PUBLIC	(publicchannel--)
n CHANGE-CHANNEL	(privatechannel--)	n DISABLE-PUBLIC	(publicchannel--)
TEST-CHANNELS	(--)	TOPO-ON?	(--flag)

RESET COMMANDS

RESET-MOTION	(--)	RESET-SPEECH	(--)
RESET-TOPO	(--)		

Mike Saari 1/14/84
Androbot Engineering Dept.
File: demos.tp2

This is a summary of the demonstration commands on the marketing demo boot disk for Topo. Type the command word (in capitals) to perform the particular demo. (The source is on screens 50-66).

SPEECH DEMONSTRATIONS

SPEECHDEMO - A demonstration of Topo's full range of speech capabilities. The wolf-whistle exists separately as WOLF.

FORDEMO - Foreign language capability demo. Says "Hello, my name is Topo" in French, Spanish, Italian, and Japanese.

Bad Robot Jokes (four) - Type Q1 for the question, R1 for the response. Similar for Q2,R2, Q3,R3, Q4,R4. The answers also include a laugh which can be called with the word LAUGH.

MOTION DEMONSTRATIONS

SPEEDDEMO - A demonstration of Topo's speed range capabilities.

SQUARE - Topo demonstrates geometry by moving in a square.

SINGING DEMONSTRATIONS

DAISYDEMO - Performs a "Daisy, daisy" song and dance.

ANTHEM - Topo sings a brief chorus from the "Androanthem".

BEATLES - Topo sings a short excerpt from the song "When I'm Sixty Four".

n BOTTLES - Sings "n Bottles of Beer on the Wall" for any value of n, i.e. 5 BOTTLES (recommended value). Gets more drunk around 2 bottles left. Demonstrates a simple Forth LOOP, and also shows what can happen when you LOOP one time too many!

Other words already included as part of the original DEMO word in TopoSoft (and thus on the demo disk as well) include: RIBBIT - do a frog croak; WHEE - spin in a circle and say "wheeeeeeeee!"; and HERES - spin and say "da da da da da...heeeeeeerrrrrrrres Topo!"

MULTIPLE TOPOS DEMONSTRATION

INIT2TOPOS and 2TOPOSDEMO - Used only if two Topos are available, to demonstrate the ability to control multiple Topos. Type INIT2TOPOS first and follow the instructions to initialize the two Topos onto channels 0 and 1, and then type 2TOPOSDEMO to see the demo. Start with Topos side-by-side facing the listener for best results.