F. APPENDICES

Copyright 1984 All rights reserved.

APPENDIX 1

Provided RCL Tasks

```
 TASKS
 Name        (Status)

 ASSIGN A MOTOR CODE
*BEGIN A COUNTED LOOP called <1> beginning at <2> ending at <3>
*BEGIN A LIMITED LOOP
*BEGIN A LOOP
*BUILD a program from the robot task <1>
*BUILD AND LOAD
*CALCULATE variable <1> = <2> (+,-,*,/) <3>  <4>
*CALL this robot task line <1>
 CLEAR ALL ITEMS
*DEFINE A VARIABLE called <1>
 END HERE
 END OF PROGRAM
*EXIT IF ANY BUMPER TOUCHED
*EXIT IF FRONT+REAR BUMPER PRESS
*EXIT IF SONAR distance value is less than <1>
*EXIT IF THE BATTERY IS LOW
*EXIT IF THE CHARGER IS TOUCHED
*EXIT IF THE TAPE IS SENSED
*EXIT IF THIS BUMPER is touched <1>
*EXIT THIS LOOP
*FOLLOW TAPE
*GO CLOCKWISE
*GO COUNTERCLOCKWISE
*HONK the horn for this many seconds <1>
 INITIALIZE MEMORY
 INITIALIZE VARIABLES
*JUMP to the line called <1>
*LIGHTS ROUTINE
*LIST THE PROGRAM
*LOAD
*LOAD THE PHRASE called: <1> with phonemes: <2>
*MAINTAIN CHARGE
*MOVE BACKWARD
*MOVE DISTANCE BACKWARD for this many feet: <1>
*MOVE DISTANCE FORWARD for this many feet: <1>
*MOVE FORWARD
*MOVE FORWARD TIL TAPE NOT SENSED
*MOVE RANDOMLY
*MOVE THE ARM FROM SHOULDER (UP,DOWN,IN,OUT): <1> for this many degrees <2>
*MOVE THE FOREARM (IN,OUT): <1> for this many degrees <2>
*MOVE THE HAND (OPEN,CLOSE): <1> for this (1-100) %: <2>
*MOVE TIMED BACKWARD for this many seconds <1>
*MOVE TIMED FORWARD for this many seconds <1>
*MOVE WITH BETA INTELLIGENCE
 OTHERWISE DO
*PICK A RANDOM DIRECTION
*PIVOT ON LEFT CLOCKWISE
*PIVOT ON LEFT COUNTERCLOCKWISE
*PIVOT ON RIGHT CLOCKWISE
*PIVOT ON RIGHT COUNTERCLOCKWISE
*PREPARE THE ROBOT
*PREPARE THE VOICE
 REMARK that: <1>
*REPEAT THE LIMITED LOOP unless <1> is (<,>,=,<>) <2> to <3>
*REPEAT THIS COUNTED LOOP for the counter <1>
*REPEAT THIS LOOP
 REPORT ASSIGN ERROR
```

```
 TASKS
 Name        (Status)

 REPORT DELETE ERROR
 REPORT GET ERROR  (Undefined)
 REPORT LOAD ERROR
 REPORT MATH ERROR
 REPORT REPLACE ERROR
 REPORT SAVE ERROR
*RUN
*SET the variable <1> equal to <2>
 SHOW ERRORS
 SHOW THE PHONEME DICTIONARY
*SPEAK the phrase called:  <1>
*SPIN AROUND CLOCKWISE this many times: <1>
*SPIN AROUND COUNTERCLOCKWISE this many times: <1>
*SPIN CLOCKWISE this many degrees: <1>
*SPIN COUNTERCLOCKWISE this many degrees: <1>
*SPIN LEFT 90 DEGREES
*SPIN RIGHT 90 DEGREES
 STARTUP
*STOP ALL MOTION
*TEST IF the variable <1> is (=,<,>,<>) <2> compared to <3>
*TURN OFF LED number <1>
*TURN OFF THE FLASHING LIGHTS
*TURN OFF THE HORN
*TURN OFF THE INFRARED LED
*TURN ON LED number <1>
*TURN ON THE FLASHING LIGHTS
*TURN ON THE HORN
*TURN ON THE INFRARED LED
*TURN THE WRIST (CW, CCW): <1> for this many degrees <2>
*WAIT this many seconds <1>
*WAIT RANDOMLY up to this many seconds <1>
*X ALPHA
*X ALPHA W/SONAR
*X BETA
*X BETA W/SONAR
*X BUMPER ACTIVATED MOTIONS
*X CHARGER FINDER
*X LOAD THE INTRODUCTION
*X SIMPLE SIMON
 Z COMPILE the BASIC statement <1> which means <2>
 Z COMPUTE 2 TO THE N-1 , with N = <1>
 Z DELAY for this many loops:  <1>
 Z DO CHECK if item <1> is in folder <2>
 Z DO NUMBER TEST on <1>
 Z EMPTY THE PROGRAM
 Z ERROR CHECK IF/UNTIL STATEMENT
 Z ERROR CHECK MATH FUNCTION
 Z ERROR CHECK STRUCTURES
 Z ERROR CHECK THE FOR STATEMENT
 Z ERROR CHECKING?
 Z FIND the label called <1>
 Z GET PREVIOUS BEGIN
 Z GET PREVIOUS IF OR ELSE
 Z INCLUDE at <1> the statement <2> which means <3>
 Z INCREMENT the <1>
 Z INCREMENT STATEMENT NUMBER
 Z INITIALIZE LEGAL VARIABLES
```

```
TASKS
Name        (Status)

Z INSURE ARM PULSE SUBROUTINE IS
Z INSURE BETA SUBROUTINE IS
Z INSURE END OF PROGRAM IS THERE
Z INSURE PORT 7801 SUBROUTINE IS
Z INSURE RANDOM TURN SUBROUTINE
Z INSURE RANDOM WAIT SUBROUTINE
Z INSURE VOICE SUBROUTINE IS
Z INSURE WAIT SUBROUTINE IS
Z LOAD SONAR
Z LOAD VOICE SUBROUTINE
Z LOOK FOR BEGIN
Z LOOK FOR MATCHING STATEMENT
Z OUTPUT PAGE NUMBER
Z OUTPUT PROGRAM HEADING
Z PATCH UP END STATEMENTS
Z PATCH UP EXIT STATEMENTS
Z PATCH UP JUMP STATEMENTS
Z SAVE FOLDER ERROR
Z SAVE NUMBER OR FOLDER ERROR
Z SAVE STRUCTURE ERROR missing the statement <1> to go with the statment <2> a
statement number <3>
ZZ AVAIL TASK
ZZ AVAIL TASK #1
ZZ AVAIL TASK #10
ZZ AVAIL TASK #11
ZZ AVAIL TASK #12
ZZ AVAIL TASK #13
ZZ AVAIL TASK #14
ZZ AVAIL TASK #2
ZZ AVAIL TASK #3
ZZ AVAIL TASK #4
ZZ AVAIL TASK #5
ZZ AVAIL TASK #6
ZZ AVAIL TASK #7
ZZ AVAIL TASK #8
```

APPENDIX 2

Detail of Provided RCL Tasks

```
ASSIGN A MOTOR CODE (a Task)

 1 Does Z COMPILE the BASIC statement "IF Y=251 THEN P=9" which means "FORWARD
 2  and Z COMPILE the BASIC statement "IF Y=254 THEN P=8" which means "RIGHT FI
WARD"
 3  and Z COMPILE the BASIC statement "IF Y=247 THEN P=5" which means "ROTATE I
GHT"
 4  and Z COMPILE the BASIC statement "IF Y=127 THEN P=7" which means "RIGHT RI
ERSE"
 5  and Z COMPILE the BASIC statement "IF Y=239 THEN P=6" which means "REVERSE
 6  and Z COMPILE the BASIC statement "IF Y=191 THEN P=2" which means "LEFT REV
RSE"
 7  and Z COMPILE the BASIC statement "IF Y=223 THEN P=10" which means "ROTATE
EFT"
 8  and Z COMPILE the BASIC statement "IF Y=253 THEN P=1" which means "LEFT FOR
ARD"
 9  and Z COMPILE the BASIC statement "@M=P" which means "ASSIGN THE MOTOR CODE

10  and END   (Task is 32% full.)

BEGIN A COUNTED LOOP called <1> beginning at <2> ending at <3>  (a Function)

 1 Does Z ERROR CHECK THE FOR STATEMENT
 2  and PASTE the "FOR " in front of <1>
 3  and PASTE the FRONT in front of "="
 4  and PASTE the FRONT in front of <2>
 5  and PASTE the FRONT in front of " TO "
 6  and PASTE the FRONT in front of <3>
 7  and Z COMPILE the BASIC statement FRONT which means "BEGIN A COUNTED LOOP (
OR/NEXT""
 8  and COPY from "FOR" to STRUCTURE TYPE
 9  and SAVE new page in STRUCTURES
10  and END   (Task is 9% full.)

BEGIN A LIMITED LOOP (a Task)

 1 Does Z COMPILE the BASIC statement "DO" which means "BEGIN A LIMITED LOOP"
 2  and COPY from "DO" to STRUCTURE TYPE
 3  and SAVE new page in STRUCTURES
 4  and END   (Task is 4% full.)

BEGIN A LOOP (a Task)

 1 Does Z COMPILE the BASIC statement "REM START A LOOP" which means "BEGIN A L
OP"
 2  and COPY from "BEGIN" to STRUCTURE TYPE
 3  and SAVE new page in LOOPS
 4  and SAVE new page in STRUCTURES
 5  and END   (Task is 5% full.)
```

```
BUILD a program from the robot task <1>  (a Function)

 1 Does CLEAR the page
 2  and COPY from "BUILD" to ASSIGNMENT
 3  and ASSIGN the <1> to ASSIGNED TASK
 4  and Z ERROR CHECKING?
 5  and CLEAR the page
 6  and Z EMPTY THE PROGRAM
 7  and DISPLAY the "Now building robot task "
 8  and DISPLAY the <1>
 9  and DISPLAY the "...one moment, please."
10  and CARRIAGE return
11  and DO the assigned task
12  and Z PATCH UP JUMP STATEMENTS
13  and Z INSURE END OF PROGRAM IS THERE
14  and COPY from EMPTY to ASSIGNMENT
15  and IF the ERROR CHECK  IS  YES then
16   Do     COPY from 199 to ERROR NUMBER
17  and     Z ERROR CHECK STRUCTURES
18  and     GET the page indexed by FIRST in folder ERRORS
19  and     IF the PAGE  IS NOT  EMPTY then
20   Do        SHOW ERRORS
21  and        END of test
22  and    END of test
23  and END    (Task is 13% full.)

BUILD AND LOAD (a Task)

 1 Does CLEAR the page
 2  and OUTPUT from "*** Build and load an RB5X robot task ***" using format DE
AULT
 3  and CARRIAGE return
 4  and CARRIAGE return
 5  and OUTPUT from "Please enter the name of the robot task:  " using format D
FAULT
 6  and INPUT into HERE
 7  and BUILD a program from the robot task HERE
 8  and CLEAR the page
 9  and OUTPUT from "** Build completed, now loading into the RB5X **" using
mat DEFAULT
10  and LOAD
11  and END    (Task is 16% full.)
```

```
CALCULATE variable <1> = <2> (+,-,*,/) <3>  <4>  (a Function)

 1 Does Z ERROR CHECK MATH FUNCTION
 2  and PASTE the <1> in front of "="
 3  and PASTE the FRONT in front of <2>
 4  and PASTE the FRONT in front of <3>
 5  and PASTE the FRONT in front of <4>
 6  and Z COMPILE the BASIC statement FRONT which means "MATH FUNCTION"
 7  and END   (Task is 5% full.)

CALL this robot task line <1>  (a Function)

 1 Does PASTE the "REM LABEL THIS LINE " in front of <1>
 2  and Z COMPILE the BASIC statement FRONT which means "LABEL THE LINE"
 3  and PASTE the "L" in front of <1>
 4  and COPY from FRONT to STRUCTURE TYPE
 5  and SAVE new page in LOOPS
 6  and END   (Task is 6% full.)

CLEAR ALL ITEMS (a Task)

 1 Does Z COMPILE the BASIC statement "CLEAR" which means "CLEAR VARIABLES"
 2  and END   (Task is 3% full.)

DEFINE A VARIABLE called <1>  (a Function)

 1 Does BEGIN at page indexed by FIRST in folder LEGAL VARIABLES
 2  and START a loop
 3   Do    GET the page indexed by NEXT in folder LEGAL VARIABLES
 4  and    IF the USED  IS  NO then
 5   Do       LEAVE this loop
 6  and       END of test
 7  and    REPEAT this loop
 8  and COPY from <1> to VARIABLES ID
 9  and COPY from LEGAL ID to VAR NAME
10  and SAVE new page in VARIABLES
11  and COPY from YES to USED
12  and REPLACE this page in folder LEGAL VARIABLES
13  and END   (Task is 5% full.)

END HERE (a Task)

 1 Does Z COMPILE the BASIC statement "REM END HERE" which means "FALL THROUGH
O HERE"
 2  and Z PATCH UP END STATEMENTS
 3  and COPY from "END" to STRUCTURE TYPE
 4  and SAVE new page in STRUCTURES
 5  and END   (Task is 5% full.)
```

```
END OF PROGRAM (a Task)

 1 Does Z INCLUDE at 2,999 the statement "GOTO 3400" which means "GOTO END"
 2  and Z INCLUDE at 3,600 the statement "REM END OF PROGRAM" which means ""
 3  and END   (Task is 6% full.)

EXIT IF ANY BUMPER TOUCHED (a Task)

 1 Does Z COMPILE the BASIC statement "Y=@#7800" which means "TEST FOR BUMPER
NTACT"
 2  and Z COMPILE the BASIC statement "IF Y<255 GOTO EXIT" which means "EXIT
ANY CONTACT"
 3  and COPY from "EXIT" to STRUCTURE TYPE
 4  and SAVE new page in LOOPS
 5  and SAVE new page in STRUCTURES
 6  and END   (Task is 10% full.)

EXIT IF FRONT+REAR BUMPER PRESS (a Task)

 1 Does Z COMPILE the BASIC statement "Y=@#7800" which means "TEST FOR BUMPER
NTACT"
 2  and Z COMPILE the BASIC statement "IF Y=235 GOTO EXIT" which means "EXIT
FRONT+REAR BUMPER PRESS"
 3  and COPY from "EXIT" to STRUCTURE TYPE
 4  and SAVE new page in LOOPS
 5  and SAVE new page in STRUCTURES
 6  and END   (Task is 11% full.)

EXIT IF SONAR distance value is less than <1>  (a Function)

 1 Does Z COMPILE the BASIC statement "D=0" which means "ZERO DISTANCE"
 2  and Z COMPILE the BASIC statement "LINK #1800" which means "CALL SONAR"
 3  and SPLIT the number <1>
 4  and PASTE the "IF D<" in front of INTEGER
 5  and PASTE the FRONT in front of " GOTO EXIT"
 6  and Z COMPILE the BASIC statement FRONT which means "EXIT IF LESS THAN VA
"
 7  and COPY from "EXIT" to STRUCTURE TYPE
 8  and SAVE new page in LOOPS
 9  and COPY from YES to SONAR NEEDED
10  and SAVE new page in STRUCTURES
11  and END   (Task is 14% full.)
```

```
EXIT IF THE BATTERY IS LOW (a Task)

 1 Does Z COMPILE the BASIC statement "Z=@#7802 AND #10" which means "TEST IF
TTERY LOW"
 2  and Z COMPILE the BASIC statement "IF Z=0 GOTO EXIT" which means "EXIT IF
TTERY IS LOW"
 3  and COPY from "EXIT" to STRUCTURE TYPE
 4  and SAVE new page in LOOPS
 5  and SAVE new page in STRUCTURES
 6  and END   (Task is 10% full.)

EXIT IF THE CHARGER IS TOUCHED (a Task)

 1 Does Z COMPILE the BASIC statement "Q=@#7802 AND #20" which means "TEST FOR
HARGER CONTACT"
 2  and Z COMPILE the BASIC statement "IF Q=0 GOTO EXIT" which means "EXIT IF
ARGER IS TOUCHED"
 3  and COPY from "EXIT" to STRUCTURE TYPE
 4  and SAVE new page in LOOPS
 5  and SAVE new page in STRUCTURES
 6  and END   (Task is 11% full.)

EXIT IF THE TAPE IS SENSED (a Task)

 1 Does TURN ON THE INFRARED LED
 2  and Z COMPILE the BASIC statement "R=@#7802 AND #40" which means "TEST FOR
APE SENSE"
 3  and Z COMPILE the BASIC statement "IF R=0 GOTO EXIT" which means "EXIT IF
PE SENSED"
 4  and COPY from "EXIT" to STRUCTURE TYPE
 5  and SAVE new page in LOOPS
 6  and SAVE new page in STRUCTURES
 7  and END   (Task is 10% full.)

EXIT IF THIS BUMPER is touched <1>  (a Function)

 1 Does IF the <1>  IS LESS THAN OR EQUAL TO  8 then
 2   Do    IF the <1>  IS GREATER THAN OR EQUAL TO  1 then
 3  Do       Z COMPILE the BASIC statement "Y=@#7800" which means "TEST FOR BU
PER CONTACT"
 4  and      SPLIT the number <1>
 5  and      Z COMPUTE 2 TO THE N-1 , with N = INTEGER
 6  and      SUBTRACT the PRODUCT from 255
 7  and      PASTE the "IF Y=" in front of DIFFERENCE
 8  and      PASTE the FRONT in front of " GOTO EXIT"
 9  and      Z COMPILE the BASIC statement FRONT which means "EXIT IF CONTACT'
10  and      COPY from "EXIT" to STRUCTURE TYPE
11  and      SAVE new page in LOOPS
12  and      SAVE new page in STRUCTURES
13  and      END of test
14  and   END of test
15  and END   (Task is 15% full.)
```

```
EXIT THIS LOOP (a Task)

 1 Does Z COMPILE the BASIC statement "GOTO GOES HERE" which means "EXIT THIS
OP"
 2  and COPY from "EXIT" to STRUCTURE TYPE
 3  and SAVE new page in LOOPS
 4  and SAVE new page in STRUCTURES
 5  and END   (Task is 5% full.)

FOLLOW TAPE (a Task)

 1 Does BEGIN A LOOP
 2  and PREPARE THE ROBOT
 3  and CLEAR ALL ITEMS
 4  and BEGIN A LOOP
 5  and PIVOT ON LEFT COUNTERCLOCKWISE
 6  and EXIT IF THE TAPE IS SENSED
 7  and EXIT IF THE CHARGER IS TOUCHED
 8  and REPEAT THIS LOOP
 9  and EXIT IF THE CHARGER IS TOUCHED
10  and TURN ON THE FLASHING LIGHTS
11  and MOVE FORWARD TIL TAPE NOT SENSED
12  and PREPARE THE ROBOT
13  and CLEAR ALL ITEMS
14  and BEGIN A LOOP
15  and PIVOT ON RIGHT CLOCKWISE
16  and EXIT IF THE TAPE IS SENSED
17  and EXIT IF THE CHARGER IS TOUCHED
18  and REPEAT THIS LOOP
19  and EXIT IF THE CHARGER IS TOUCHED
20  and TURN ON THE FLASHING LIGHTS
21  and MOVE FORWARD TIL TAPE NOT SENSED
22  and REPEAT THIS LOOP
23  and END   (Task is 6% full.)

GO CLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#05" which means "SPIN CLOCKWIS
 2  and END   (Task is 3% full.)

GO COUNTERCLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#0A" which means "SPIN COUNTERCLO
KWISE"
 2  and END   (Task is 4% full.)
```

```
HONK the horn for this many seconds <1>  (a Function)

 1 Does TURN ON THE HORN
 2  and WAIT this many seconds <1>
 3  and TURN OFF THE HORN
 4  and END   (Task is 1% full.)

INITIALIZE MEMORY (a Task)

 1 Does Z COMPILE the BASIC statement "N=TOP" which means "INITIALIZE EXPERIE
BLOCK"
 2  and Z COMPILE the BASIC statement "O=TOP+#FF" which means "INITIALIZE INF
TION BLOCK"
 3  and Z COMPILE the BASIC statement "M=TOP+#200" which means ""
 4  and Z COMPILE the BASIC statement "FOR P=N TO M" which means ""
 5  and Z COMPILE the BASIC statement "@P=#FF" which means ""
 6  and Z COMPILE the BASIC statement "NEXT P" which means ""
 7  and END   (Task is 15% full.)

INITIALIZE VARIABLES (a Task)

 1 Does Z COMPILE the BASIC statement "N=TOP" which means ""
 2  and Z COMPILE the BASIC statement "O=TOP+#FF" which means ""
 3  and Z COMPILE the BASIC statement "M=N" which means ""
 4  and Z COMPILE the BASIC statement "FOR P=N TO O" which means "INITIALIZE I
ORY"
 5  and Z COMPILE the BASIC statement "@P=0" which means ""
 6  and Z COMPILE the BASIC statement "NEXT P" which means ""
 7  and END   (Task is 11% full.)

JUMP to the line called <1>  (a Function)

 1 Does PASTE the "JUMP TO LABEL " in front of <1>
 2  and Z COMPILE the BASIC statement "GOTO GOES HERE" which means FRONT
 3  and PASTE the "J" in front of <1>
 4  and COPY from FRONT to STRUCTURE TYPE
 5  and SAVE new page in LOOPS
 6  and END   (Task is 6% full.)

LIGHTS ROUTINE (a Task)

 1 Does TURN ON LED number 2
 2  and WAIT this many seconds .1
 3  and TURN ON LED number 3
 4  and WAIT this many seconds .1
 5  and TURN ON LED number 4
 6  and WAIT this many seconds .1
 7  and TURN ON LED number 5
 8  and WAIT this many seconds .1
 9  and TURN ON THE FLASHING LIGHTS
10  and WAIT this many seconds 1
11  and END   (Task is 7% full.)
```

LIST THE PROGRAM (a Task)

```
 1 Does BEGIN at page indexed by FIRST in folder SOURCE
 2  and COPY from 0 to PAGE COUNT
 3  and START a loop
 4   Do    CLEAR the page
 5  and    Z INCREMENT the PAGE COUNT
 6  and    Z OUTPUT PAGE NUMBER
 7  and    Z OUTPUT PROGRAM HEADING
 8  and    COPY from 5 to LINE COUNT
 9  and    START a loop
10   Do       Z INCREMENT the LINE COUNT
11  and       GET the page indexed by NEXT in folder SOURCE
12  and       IF the PAGE  IS  EMPTY then
13   Do          LEAVE this loop
14  and          END of test
15  and       OUTPUT from SPACES using format 5
16  and       OUTPUT from STATEMENT NUMBER using format "#4"
17  and       OUTPUT from SPACES using format 5
18  and       OUTPUT from BASIC TEXT using format 30
19  and       OUTPUT from SPACES using format 5
20  and       OUTPUT from MEANING using format 30
21  and       CARRIAGE return
22  and       IF the LINE COUNT  IS  55 then
23   Do          LEAVE this loop
24  and          END of test
25  and       REPEAT this loop
26  and    IF the PAGE  IS  EMPTY then
27   Do       LEAVE this loop
28  and       END of test
29  and    REPEAT this loop
30  and CARRIAGE return
31  and CARRIAGE return
32  and OUTPUT from SPACES using format 26
33  and OUTPUT from "** End of the Program **" using format DEFAULT
34  and CARRIAGE return
35  and END   (Task is 19% full.)
```

LOAD (a Task)

```
 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  10 then
 2   Do    REPORT LOAD ERROR
 3  and    END of test
 4  and CLEAR the page
 5  and RING bell
 6  and RING bell
 7  and RING bell
 8  and DISPLAY the "Ready to load into the RB5X...press any key to continue.
 9  and LIMIT the input for ANSWER to a maximum of 1
10  and CLEAR the page
11  and DISPLAY the "** Now loading...One moment please **"
12  and CARRIAGE return
13  and CARRIAGE return
14  and DISPLAY the "Now loading your robot task..."
15  and CARRIAGE return
16  and PRINT the ""
17  and OUTPUT from "@#7803=#98:@#7801=#7C" using format DEFAULT
18  and CARRIAGE return
19  and Z DELAY for this many loops:  10
20  and OUTPUT from "NEW #1000" using format DEFAULT
```

```
LOAD (a Task)

 21  and CARRIAGE return
 22  and Z DELAY for this many loops:  60
 23  and OUTPUT from "NEW" using format DEFAULT
 24  and CARRIAGE return
 25  and Z DELAY for this many loops:  5
 26  and BEGIN at page indexed by FIRST in folder SOURCE
 27  and IF the VOICE NEEDED  IS  YES then
 28   Do    Z LOAD VOICE SUBROUTINE
 29  and    DISPLAY the ""
 30  and    CARRIAGE return
 31  and    PRINT the ""
 32  and    END of test
 33  and IF the SONAR NEEDED  IS  YES then
 34   Do    Z LOAD SONAR
 35  and    DISPLAY the ""
 36  and    CARRIAGE return
 37  and    PRINT the ""
 38  and    END of test
 39  and DISPLAY the ""
 40  and CARRIAGE return
 41  and DISPLAY the "Now loading your Tiny BASIC robot task..."
 42  and CARRIAGE return
 43  and PRINT the ""
 44  and START a loop
 45   Do    GET the page indexed by NEXT in folder SOURCE
 46  and    IF the PAGE  IS  EMPTY then
 47   Do       LEAVE this loop
 48  and       END of test
 49  and    PASTE the STATEMENT NUMBER in front of " "
 50  and    PASTE the FRONT in front of BASIC TEXT
 51  and    OUTPUT from FRONT using format DEFAULT
 52  and    CARRIAGE return
 53  and    DISPLAY the "#"
 54  and    PRINT the ""
 55  and    Z DELAY for this many loops:  5
 56  and    REPEAT this loop
 57  and Z DELAY for this many loops:  15
 58  and PRINT the ""
 59  and OUTPUT from "RUN" using format DEFAULT
 60  and CARRIAGE return
 61  and DISPLAY the ""
 62  and RING bell
 63  and CLEAR the page
 64  and DISPLAY the "*** Your robot task has been successfully loaded ***"
 65  and END   (Task is 47% full.)
```

```
LOAD THE PHRASE called: <1> with phonemes: <2>  (a Function)

 1 Does COPY from 0 to SPEECH LENGTH
 2  and COPY from EMPTY to SPEECH ADDRESS
 3  and GET the page indexed by FIRST in folder SPEECH LOAD DATA
 4  and DISPLAY the PAGE
 5  and IF the PAGE  IS  EMPTY then
 6   Do    COPY from "TOP+500" to SPEECH ADDRESS
 7  and    COPY from "TEMP" to LABEL ID
 8  and    SAVE new page in SPEECH LOAD DATA
 9  and    GET the page indexed by "TEMP" in folder SPEECH LOAD DATA
10  and    END of test
11  and GET the page indexed by "TEMP" in folder SPEECH LOAD DATA
12  and Z INSURE VOICE SUBROUTINE IS
13  and START a loop
14   Do    IF the <2>  IS  EMPTY then
15   Do       LEAVE this loop
16  and       END of test
17  and    ADD the 1 and SPEECH LENGTH
18  and    COPY from SUM to SPEECH LENGTH
19  and    CUT the <2> after DELIMITER
20  and    COPY from BACK to <2>
21  and    SUBTRACT the 1 from LENGTH
22  and    TAKE this many characters DIFFERENCE from FRONT
23  and    WHEN folder PHONEMES  CONTAINS  FRONT then
24   Do       GET the page indexed by FRONT in folder PHONEMES
25  and       PASTE the "P=#" in front of PHONEME CODE
26  and       PASTE the FRONT in front of ":GOSUB 3500"
27  and       Z COMPILE the BASIC statement FRONT which means "VOICE SUBROUTI
"
28  and       COPY from YES to VOICE NEEDED
29  and    OTHERWISE
30   Do       IF the ERROR CHECK  IS  YES then
31   Do          Z COMPILE the BASIC statement "REM PHONEME ERROR" which means
"
32  and          Z INCREMENT the ERROR NUMBER
33  and          COPY from STATEMENT NUMBER to LINE NUMBER
34  and          PASTE the PHONEME ID in front of " is not a valid phoneme"
35  and          COPY from FRONT to ERROR MESSAGE
36  and          SAVE new page in ERRORS
37  and          END of test
38  and       END of test
39  and    REPEAT this loop
40  and GET the page indexed by LAST in folder SPEECH LOAD DATA
41  and COPY from <1> to LABEL ID
42  and SAVE new page in SPEECH LOAD DATA
43  and PASTE the SPEECH ADDRESS in front of "+"
44  and PASTE the FRONT in front of SPEECH LENGTH
45  and COPY from FRONT to SPEECH ADDRESS
46  and COPY from "TEMP" to LABEL ID
47  and COPY from SPEECH ADDRESS to HERE
48  and GET the page indexed by "TEMP" in folder SPEECH LOAD DATA
49  and COPY from HERE to SPEECH ADDRESS
50  and REPLACE this page in folder SPEECH LOAD DATA
51  and END   (Task is 38% full.)
```

```
MAINTAIN CHARGE (a Task)

 1 Does BEGIN A LOOP
 2  and BEGIN A LOOP
 3  and EXIT IF THE CHARGER IS TOUCHED
 4  and SET the variable "P" equal to 0
 5  and BEGIN A LOOP
 6  and Z COMPILE the BASIC statement "P=P+1" which means "INCREMENT LOOP COU

 7  and MOVE TIMED FORWARD for this many seconds .1
 8  and EXIT IF THE CHARGER IS TOUCHED
 9  and TEST IF the variable "P" is (=,<,>,<>) "=" compared to 5
10  and MOVE TIMED BACKWARD for this many seconds 1
11  and MOVE TIMED FORWARD for this many seconds 1.1
12  and SET the variable "P" equal to 0
13  and END HERE
14  and REPEAT THIS LOOP
15  and REPEAT THIS LOOP
16  and TURN ON LED number 2
17  and WAIT this many seconds 1
18  and PREPARE THE ROBOT
19  and REPEAT THIS LOOP
20  and END   (Task is 14% full.)

MOVE BACKWARD (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#06" which means "REVERSE"
 2  and END   (Task is 2% full.)

MOVE DISTANCE BACKWARD for this many feet: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by 3
 3  and    MOVE BACKWARD
 4  and    WAIT this many seconds PRODUCT
 5  and    STOP ALL MOTION
 6  and    END of test
 7  and END   (Task is 3% full.)

MOVE DISTANCE FORWARD for this many feet: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by 3.
 3  and    MOVE FORWARD
 4  and    WAIT this many seconds PRODUCT
 5  and    STOP ALL MOTION
 6  and    END of test
 7  and END   (Task is 3% full.)
```

```
MOVE FORWARD (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#09" which means "GO FORWARD"
 2  and END   (Task is 3% full.)

MOVE FORWARD TIL TAPE NOT SENSED (a Task)

 1 Does SET the variable "T" equal to 0
 2  and Z COMPILE the BASIC statement "DO" which means "BEGIN A LIMITED LOOP"
 3  and CALCULATE variable "T" = "T" (+,-,*,/) "+"  1
 4  and MOVE FORWARD
 5  and Z COMPILE the BASIC statement "R=@#7802 AND #40" which means "TEST FO
APE SENSE"
 6  and Z COMPILE the BASIC statement "UNTIL (R<>0) OR (T>=100)" which means "
ECK EXIT CONDITIONS"
 7  and WAIT this many seconds .1
 8  and END   (Task is 17% full.)

MOVE RANDOMLY (a Task)

 1 Does Z COMPILE the BASIC statement "GOSUB 3040" which means "PICK RANDOM DIR
CTION"
 2  and Z COMPILE the BASIC statement "@#7802=V" which means "GO RANDOM DIREC
N"
 3  and Z INSURE RANDOM TURN SUBROUTINE
 4  and END   (Task is 7% full.)

MOVE THE ARM FROM SHOULDER (UP,DOWN,IN,OUT): <1> for this many degrees <2>  (a
unction)

 1 Does IF the <1>  IS  IN then
 2   Do    COPY from "D=#7805:M=8:N=0:" to TEMP
 3  and OTHERWISE
 4   Do    IF the <1>  IS  OUT then
 5   Do       COPY from "D=#7805:M=12:N=4:" to TEMP
 6  and    OTHERWISE
 7   Do       IF the <1>  IS  DOWN then
 8   Do          COPY from "D=#7805:M=3:N=1:" to TEMP
 9  and       OTHERWISE
10   Do          IF the <1>  IS  UP then
11   Do             COPY from "D=#7805:M=2:N=0:" to TEMP
12  and             END of test
13  and          END of test
14  and       END of test
15  and    END of test
16  and IF the <2>  IS GREATER THAN OR EQUAL TO  1 then
17   Do    IF the <2>  IS LESS THAN OR EQUAL TO  360 then
18   Do       MULTIPLY the <2> by 40.38
19  and       SPLIT the number PRODUCT
20  and       COPY from INTEGER to PRODUCT
21  and       PASTE the "P=" in front of PRODUCT
22  and       PASTE the TEMP in front of FRONT
23  and       Z COMPILE the BASIC statement FRONT which means "SET MOTOR, ETC.
24  and       Z COMPILE the BASIC statement "GOSUB 3400" which means "MOVE THE A
RM"
25  and       Z INSURE ARM PULSE SUBROUTINE IS
26  and       GO to the line called "SKIP"
27  and       END of test
28  and    END of test
29  and CARRIAGE return
```

```
MOVE THE ARM FROM SHOULDER (UP,DOWN,IN,OUT): <1> for this many degrees <2>  (
unction)

30  and OUTPUT from "Shoulder Error:  degrees = " using format DEFAULT
31  and OUTPUT from <2> using format DEFAULT
32  and HALT and return to Ready
33  and LABEL this line "SKIP"
34  and END   (Task is 30% full.)

MOVE THE FOREARM (IN,OUT): <1> for this many degrees <2>  (a Function)

 1 Does IF the <1>  IS  IN then
 2   Do    COPY from "D=#7805:M=48:N=16:" to TEMP
 3  and OTHERWISE
 4   Do    IF the <1>  IS  OUT then
 5   Do       COPY from "D=#7805:M=32:N=0:" to TEMP
 6  and       END of test
 7  and    END of test
 8  and IF the <2>  IS GREATER THAN OR EQUAL TO  1 then
 9   Do    IF the <2>  IS LESS THAN OR EQUAL TO  360 then
10   Do       MULTIPLY the <2> by 20.16
11  and       SPLIT the number PRODUCT
12  and       COPY from INTEGER to PRODUCT
13  and       PASTE the "P=" in front of PRODUCT
14  and       PASTE the TEMP in front of FRONT
15  and       Z COMPILE the BASIC statement FRONT which means "SET THE MOTOR,
C."
16  and       Z COMPILE the BASIC statement "GOSUB 3400" which means "MOVE THE
RM"
17  and       Z INSURE ARM PULSE SUBROUTINE IS
18  and       GO to the line called "SKIP"
19  and       END of test
20  and    END of test
21  and CARRIAGE return
22  and OUTPUT from "Forearm Error:  degrees = " using format DEFAULT
23  and OUTPUT from <2> using format DEFAULT
24  and HALT and return to Ready
25  and LABEL this line "SKIP"
26  and END   (Task is 24% full.)

MOVE THE HAND (OPEN,CLOSE): <1> for this (1-100) %: <2>  (a Function)

 1 Does IF the <1>  IS  OPEN then
 2   Do    COPY from "D=#7806:M=242:N=240:" to TEMP
 3  and OTHERWISE
 4   Do    IF the <1>  IS  CLOSE then
 5   Do       COPY from "D=#7806:M=243:N=241:" to TEMP
 6  and       END of test
 7  and    END of test
 8  and IF the <2>  IS GREATER THAN OR EQUAL TO  1 then
 9   Do    IF the <2>  IS LESS THAN OR EQUAL TO  100 then
10   Do       MULTIPLY the <2> by 4.42
11  and       SPLIT the number PRODUCT
12  and       COPY from INTEGER to PRODUCT
13  and       SPLIT the number PRODUCT
14  and       PASTE the "P=" in front of INTEGER
15  and       PASTE the TEMP in front of FRONT
16  and       Z COMPILE the BASIC statement FRONT which means "SET MOTORS, ETC.

17  and       Z COMPILE the BASIC statement "GOSUB 3400" which means "MOVE THE
AND"
```

```
MOVE THE HAND (OPEN,CLOSE): <1> for this (1-100) %: <2>  (a Function)

18  and        Z INSURE ARM PULSE SUBROUTINE IS
19  and        GO to the line called "SKIP"
20  and        END of test
21  and     END of test
22  and CARRIAGE return
23  and OUTPUT from "Hand Error:  percent = " using format DEFAULT
24  and OUTPUT from <2> using format DEFAULT
25  and HALT and return to Ready
26  and LABEL this line "SKIP"
27  and END   (Task is 24% full.)

MOVE TIMED BACKWARD for this many seconds <1>  (a Function)

 1 Does MOVE BACKWARD
 2  and WAIT this many seconds <1>
 3  and STOP ALL MOTION
 4  and END   (Task is 1% full.)

MOVE TIMED FORWARD for this many seconds <1>  (a Function)

 1 Does MOVE FORWARD
 2  and WAIT this many seconds <1>
 3  and STOP ALL MOTION
 4  and END   (Task is 1% full.)

MOVE WITH BETA INTELLIGENCE (a Task)

 1 Does Z COMPILE the BASIC statement "GOSUB 3160" which means "GO TO BETA SUBR
UTINE"
 2  and Z COMPILE the BASIC statement "REM RETURN GOES HERE" which means "RE
 TO HERE"
 3  and COPY from STATEMENT NUMBER to RETURN STATEMENT NUMBER
 4  and Z INSURE BETA SUBROUTINE IS
 5  and END   (Task is 9% full.)

OTHERWISE DO (a Task)

 1 Does Z COMPILE the BASIC statement "REM ELSE" which means "DO THIS INSTEAD"
 2  and COPY from "ELSE" to STRUCTURE TYPE
 3  and SAVE new page in LOOPS
 4  and SAVE new page in STRUCTURES
 5  and END   (Task is 5% full.)
```

```
PICK A RANDOM DIRECTION (a Task)

 1 Does Z COMPILE the BASIC statement "GOSUB 3040" which means "PICK RANDOM DI
CTION"
 2  and Z INSURE RANDOM TURN SUBROUTINE
 3  and END    (Task is 4% full.)

PIVOT ON LEFT CLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#04" which means "RIGHT REVERSE"
 2  and END    (Task is 3% full.)

PIVOT ON LEFT COUNTERCLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#08" which means "RIGHT FORWARD"
 2  and END    (Task is 3% full.)

PIVOT ON RIGHT CLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#01" which means "LEFT FORWARD"
 2  and END    (Task is 3% full.)

PIVOT ON RIGHT COUNTERCLOCKWISE (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=#02" which means "LEFT REVERSE"
 2  and END    (Task is 3% full.)

PREPARE THE ROBOT (a Task)

 1 Does Z COMPILE the BASIC statement "@#7803=#98" which means "INITIALIZE I/O"
 2  and END    (Task is 3% full.)

PREPARE THE VOICE (a Task)

 1 Does Z COMPILE the BASIC statement "@#780B=#A7:S=TOP" which means "PREPARE T
E VOICE"
 2  and COPY from "YES" to VOICE NEEDED
 3  and END    (Task is 5% full.)

REMARK that: <1>  (a Function)

 1 Does PASTE the "REM " in front of <1>
 2  and Z COMPILE the BASIC statement FRONT which means "REMARK"
 3  and END    (Task is 2% full.)
```

```
REPEAT THE LIMITED LOOP unless <1> is (<,>,=,<>) <2> to <3>  (a Function)

 1 Does Z ERROR CHECK IF/UNTIL STATEMENT
 2  and PASTE the "UNTIL " in front of <1>
 3  and PASTE the FRONT in front of <2>
 4  and PASTE the FRONT in front of <3>
 5  and Z COMPILE the BASIC statement FRONT which means "CHECK EXIT CONDITION"
 6  and COPY from "UNTIL" to STRUCTURE TYPE
 7  and SAVE new page in STRUCTURES
 8  and END   (Task is 7% full.)

REPEAT THIS COUNTED LOOP for the counter <1>  (a Function)

 1 Does Z DO CHECK if item <1> is in folder "VARIABLES"
 2  and IF the RESULT  IS  "FAIL" then
 3   Do    Z SAVE FOLDER ERROR
 4  and OTHERWISE
 5   Do    GET the page indexed by <1> in folder VARIABLES
 6  and    COPY from VAR NAME to <1>
 7  and    END of test
 8  and PASTE the "NEXT " in front of <1>
 9  and Z COMPILE the BASIC statement FRONT which means "REPEAT THE COUNTED LO
(FOR/NEXT)"
10  and COPY from "NEXT" to STRUCTURE TYPE
11  and SAVE new page in STRUCTURES
12  and END   (Task is 11% full.)

REPEAT THIS LOOP (a Task)

 1 Does COPY from STATEMENT NUMBER to HERE
 2  and Z GET PREVIOUS BEGIN
 3  and DELETE the page indexed by STATEMENT NUMBER from folder LOOPS
 4  and PASTE the "GOTO " in front of STATEMENT NUMBER
 5  and COPY from HERE to STATEMENT NUMBER
 6  and Z COMPILE the BASIC statement FRONT which means "REPEAT THIS LOOP"
 7  and Z COMPILE the BASIC statement "REM EXIT TO HERE" which means ""
 8  and Z PATCH UP EXIT STATEMENTS
 9  and COPY from "REPEAT" to STRUCTURE TYPE
10  and SAVE new page in STRUCTURES
11  and END   (Task is 10% full.)

REPORT ASSIGN ERROR (a Task)

 1 Does CARRIAGE return
 2  and IF the ASSIGNMENT  IS  "BUILD" then
 3   Do    DISPLAY the "I'm sorry, I could not find the requested robot script."

 4  and    HALT and return to Ready
 5  and OTHERWISE
 6   Do    DISPLAY the "REPORT ASSIGN ERROR!"
 7  and    END of test
 8  and END   (Task is 10% full.)
```

```
REPORT DELETE ERROR (a Task)

 1 Does IF the ASSIGNMENT  IS  "BUILD" then
 2   Do    DISPLAY the ""
 3  and OTHERWISE
 4   Do    DISPLAY the "REPORT DELETE ERROR!"
 5  and    END of test
 6  and END   (Task is 4% full.)

REPORT GET ERROR  (Undefined) (a Task)

Undefined task!   (Task is 1% full.)

REPORT LOAD ERROR (a Task)

 1 Does CARRIAGE return
 2  and DISPLAY the "Sorry, there is not a program to load."
 3  and HALT and return to Ready
 4  and END   (Task is 4% full.)

REPORT MATH ERROR (a Task)

 1 Does DISPLAY the ""
 2  and END   (Task is 1% full.)

REPORT REPLACE ERROR (a Task)

 1 Does END   (Task is 1% full.)

REPORT SAVE ERROR (a Task)

 1 Does CARRIAGE return
 2  and DISPLAY the "REPORT SAVE ERROR!"
 3  and END   (Task is 2% full.)

RUN (a Task)

 1 Does PRINT the ""
 2  and OUTPUT from "RUN" using format DEFAULT
 3  and CARRIAGE return
 4  and END   (Task is 1% full.)

SET the variable <1> equal to <2>  (a Function)

 1 Does Z DO CHECK if item <1> is in folder "VARIABLES"
 2  and IF the RESULT  IS  "FAIL" then
 3   Do    Z SAVE FOLDER ERROR
 4  and OTHERWISE
 5   Do    GET the page indexed by <1> in folder VARIABLES
 6  and    COPY from VAR NAME to <1>
 7  and    END of test
 8  and Z DO NUMBER TEST on <2>
 9  and IF the RESULT  IS  "FAIL" then
10   Do    Z DO CHECK if item <2> is in folder "VARIABLES"
11  and    IF the RESULT  IS  "FAIL" then
12   Do       Z SAVE NUMBER OR FOLDER ERROR
13  and    OTHERWISE
14   Do       GET the page indexed by <2> in folder VARIABLES
15  and       COPY from VAR NAME to <2>
16  and       END of test
```

```
SET the variable <1> equal to <2>  (a Function)

17  and     END of test
18  and PASTE the <1> in front of "="
19  and PASTE the FRONT in front of <2>
20  and Z COMPILE the BASIC statement FRONT which means "INITIALIZE"
21  and END    (Task is 15% full.)

SHOW ERRORS (a Task)

 1 Does CLEAR the page
 2  and OUTPUT from "LINE NUMBER" using format DEFAULT
 3  and OUTPUT from SPACES using format 23
 4  and OUTPUT from "ERROR MESSAGE" using format DEFAULT
 5  and CARRIAGE return
 6  and CARRIAGE return
 7  and BEGIN at page indexed by FIRST in folder ERRORS
 8  and START a loop
 9   Do     GET the page indexed by NEXT in folder ERRORS
10  and     IF the PAGE  IS  EMPTY then
11   Do        LEAVE this loop
12  and        END of test
13  and     IF the ERROR NUMBER  IS  200 then
14   Do        CARRIAGE return
15  and        OUTPUT from SPACES using format 25
16  and        OUTPUT from "--- Structure Errors ---" using format DEFAULT
17  and        CARRIAGE return
18  and        CARRIAGE return
19  and        END of test
20  and     OUTPUT from LINE NUMBER using format 7
21  and     OUTPUT from SPACES using format 8
22  and     OUTPUT from ERROR MESSAGE using format DEFAULT
23  and     CARRIAGE return
24  and     REPEAT this loop
25  and CARRIAGE return
26  and END    (Task is 15% full.)

SHOW THE PHONEME DICTIONARY (a Task)

 1 Does CLEAR the page
 2  and OUTPUT from "Phoneme Symbol" using format DEFAULT
 3  and OUTPUT from SPACES using format 10
 4  and OUTPUT from "Phoneme Code" using format DEFAULT
 5  and CARRIAGE return
 6  and CARRIAGE return
 7  and BEGIN at page indexed by FIRST in folder PHONEMES
 8  and START a loop
 9   Do     GET the page indexed by NEXT in folder PHONEMES
10  and     IF the PAGE  IS  EMPTY then
11   Do        LEAVE this loop
12  and        END of test
13  and     OUTPUT from SPACES using format 5
14  and     OUTPUT from PHONEME ID using format 24
15  and     OUTPUT from PHONEME CODE using format 2
16  and     CARRIAGE return
17  and     REPEAT this loop
18  and END    (Task is 10% full.)
```

```
SPEAK the phrase called:  <1>  (a Function)

 1 Does BEGIN at page indexed by FIRST in folder SPEECH LOAD DATA
 2  and GET the page indexed by <1> in folder SPEECH LOAD DATA
 3  and COPY from SPEECH ADDRESS to HERE
 4  and PASTE the "S=" in front of HERE
 5  and PASTE the FRONT in front of ":LINK #2F50"
 6  and PASTE the FRONT in front of ":REM SPEAK"
 7  and PASTE the FRONT in front of <1>
 8  and Z COMPILE the BASIC statement FRONT which means "SPEAK A PHRASE"
 9  and END   (Task is 9% full.)

SPIN AROUND CLOCKWISE this many times: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by 5.53
 3  and    SPLIT the number PRODUCT
 4  and    TAKE this many characters 4 from FRACTION
 5  and    PASTE the INTEGER in front of FRONT
 6  and    GO CLOCKWISE
 7  and    WAIT this many seconds FRONT
 8  and    STOP ALL MOTION
 9  and    END of test
10  and END   (Task is 5% full.)

SPIN AROUND COUNTERCLOCKWISE this many times: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by 5.53
 3  and    SPLIT the number PRODUCT
 4  and    TAKE this many characters 4 from FRACTION
 5  and    PASTE the INTEGER in front of FRONT
 6  and    GO COUNTERCLOCKWISE
 7  and    WAIT this many seconds FRONT
 8  and    STOP ALL MOTION
 9  and    END of test
10  and END   (Task is 5% full.)

SPIN CLOCKWISE this many degrees: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by .01472
 3  and    SPLIT the number PRODUCT
 4  and    TAKE this many characters 4 from FRACTION
 5  and    PASTE the INTEGER in front of FRONT
 6  and    GO CLOCKWISE
 7  and    WAIT this many seconds FRONT
 8  and    STOP ALL MOTION
 9  and    END of test
10  and END   (Task is 5% full.)
```

```
SPIN COUNTERCLOCKWISE this many degrees: <1>  (a Function)

 1 Does IF the <1>  IS GREATER THAN  0 then
 2   Do    MULTIPLY the <1> by .01472
 3  and    SPLIT the number PRODUCT
 4  and    TAKE this many characters 4 from FRACTION
 5  and    PASTE the INTEGER in front of FRONT
 6  and    GO COUNTERCLOCKWISE
 7  and    WAIT this many seconds FRONT
 8  and    STOP ALL MOTION
 9  and    END of test
10  and END   (Task is 5% full.)

SPIN LEFT 90 DEGREES (a Task)

 1 Does WAIT this many seconds .5
 2  and GO COUNTERCLOCKWISE
 3  and WAIT this many seconds 1.38
 4  and STOP ALL MOTION
 5  and END   (Task is 2% full.)

SPIN RIGHT 90 DEGREES (a Task)

 1 Does WAIT this many seconds .5
 2  and GO CLOCKWISE
 3  and WAIT this many seconds 1.38
 4  and STOP ALL MOTION
 5  and END   (Task is 2% full.)

STARTUP (a Task)

 1 Does CLEAR the page
 2  and DISPLAY the "Robot Control Language (RCL-II) with SAVVY -- Version 1.0
of "
 3  and DISPLAY the VERSION
 4  and CARRIAGE return
 5  and CARRIAGE return
 6  and DISPLAY the "RCL-II is Copyright (C) 1983 by RB Robot Corporation.  Al
rights reserved."
 7  and CARRIAGE return
 8  and CARRIAGE return
 9  and END   (Task is 15% full.)
```

```
STOP ALL MOTION (a Task)

 1 Does Z COMPILE the BASIC statement "@#7802=0" which means "STOP ALL MOTION"
 2  and END   (Task is 3% full.)

TEST IF the variable <1> is (=,<,>,<>) <2> compared to <3>  (a Function)

 1 Does COPY from "TEST IF" to TASK NAME
 2  and Z ERROR CHECK IF/UNTIL STATEMENT
 3  and PASTE the "IF " in front of <1>
 4  and PASTE the FRONT in front of <2>
 5  and PASTE the FRONT in front of <3>
 6  and PASTE the FRONT in front of " GOTO "
 7  and Z COMPILE the BASIC statement FRONT which means "MAKE A COMPARISON"
 8  and COPY from "IF" to STRUCTURE TYPE
 9  and SAVE new page in LOOPS
10  and COPY from EMPTY to TASK NAME
11  and SAVE new page in STRUCTURES
12  and END   (Task is 9% full.)

TURN OFF LED number <1>  (a Function)

 1 Does IF the <1>  IS  1 then
 2   Do    Z COMPILE the BASIC statement "X=#FD" which means "TURN OFF LED 1"
 3  and OTHERWISE
 4   Do    IF the <1>  IS  2 then
 5   Do       Z COMPILE the BASIC statement "X=#FB" which means "TURN OFF LED 2

 6  and    OTHERWISE
 7   Do       IF the <1>  IS  3 then
 8   Do          Z COMPILE the BASIC statement "X=#F7" which means "TURN OFF LE
 3"
 9  and       OTHERWISE
10   Do          IF the <1>  IS  4 then
11   Do             Z COMPILE the BASIC statement "X=#EF" which means "TURN OFF
LED 4"
12  and          OTHERWISE
13   Do             Z COMPILE the BASIC statement "X=#DF" which means "TURN OFF
LED 5"
14  and             END of test
15  and          END of test
16  and       END of test
17  and    END of test
18  and Z COMPILE the BASIC statement "GOSUB 3120" which means " GO TURN OFF A
IT"
19  and Z INSURE PORT 7801 SUBROUTINE IS
20  and END   (Task is 23% full.)
```

```
TURN OFF THE FLASHING LIGHTS (a Task)

 1 Does Z COMPILE the BASIC statement "X=#BF" which means "TURN OFF FLASHING L
HTS"
 2  and Z COMPILE the BASIC statement "GOSUB 3120" which means " GO TURN OFF A
IT"
 3  and Z INSURE PORT 7801 SUBROUTINE IS
 4  and END   (Task is 7% full.)

TURN OFF THE HORN (a Task)

 1 Does Z COMPILE the BASIC statement "X=#7F" which means "TURN OFF HORN"
 2  and Z COMPILE the BASIC statement "GOSUB 3120" which means "GO TURN OFF A B
T"
 3  and Z INSURE PORT 7801 SUBROUTINE IS
 4  and END   (Task is 6% full.)

TURN OFF THE INFRARED LED (a Task)

 1 Does TURN OFF LED number 1
 2  and END   (Task is 1% full.)

TURN ON LED number <1>  (a Function)

 1 Does IF the <1>  IS  1 then
 2   Do    Z COMPILE the BASIC statement "X=#02" which means "TURN ON LED 1"
 3  and OTHERWISE
 4   Do    IF the <1>  IS  2 then
 5   Do       Z COMPILE the BASIC statement "X=#04" which means "TURN ON LED 2
 6  and    OTHERWISE
 7   Do       IF the <1>  IS  3 then
 8   Do          Z COMPILE the BASIC statement "X=#08" which means "TURN ON LE
3"
 9  and       OTHERWISE
10   Do          IF the <1>  IS  4 then
11   Do             Z COMPILE the BASIC statement "X=#10" which means "TURN ON
ED 4"
12  and          OTHERWISE
13   Do             Z COMPILE the BASIC statement "X=#20" which means "TURN ON L
ED 5"
14  and             END of test
15  and          END of test
16  and       END of test
17  and    END of test
18  and Z COMPILE the BASIC statement "GOSUB 3100" which means " GO TURN ON A
T"
19  and Z INSURE PORT 7801 SUBROUTINE IS
20  and END   (Task is 23% full.)
```

```
TURN ON THE FLASHING LIGHTS (a Task)

 1 Does Z COMPILE the BASIC statement "X=#40" which means "TURN ON FLASHING LI
TS"
 2  and Z COMPILE the BASIC statement "GOSUB 3100" which means " GO TURN ON A
T"
 3  and Z INSURE PORT 7801 SUBROUTINE IS
 4  and END   (Task is 7% full.)

TURN ON THE HORN (a Task)

 1 Does Z COMPILE the BASIC statement "X=#80" which means "TURN ON HORN"
 2  and Z COMPILE the BASIC statement "GOSUB 3100" which means "GO TURN ON A B
"
 3  and Z INSURE PORT 7801 SUBROUTINE IS
 4  and END   (Task is 6% full.)

TURN ON THE INFRARED LED (a Task)

 1 Does TURN ON LED number 1
 2  and END   (Task is 1% full.)

TURN THE WRIST (CW, CCW): <1> for this many degrees <2>  (a Function)

 1 Does IF the <1>  IS  CW then
 2   Do    COPY from "D=#7805:M=128:N=0:" to TEMP
 3  and OTHERWISE
 4   Do    IF the <1>  IS  CCW then
 5   Do       COPY from "D=#7805:M=192:N=64:" to TEMP
 6  and       END of test
 7  and    END of test
 8  and IF the <2>  IS GREATER THAN OR EQUAL TO  1 then
 9   Do    IF the <2>  IS LESS THAN OR EQUAL TO  360 then
10   Do       MULTIPLY the <2> by 19.88
11  and       SPLIT the number PRODUCT
12  and       COPY from INTEGER to PRODUCT
13  and       PASTE the "P=" in front of PRODUCT
14  and       PASTE the TEMP in front of FRONT
15  and       Z COMPILE the BASIC statement FRONT which means "SET THE MOTOR, E
C."
16  and       Z COMPILE the BASIC statement "GOSUB 3400" which means "TURN THE
RIST"
17  and       Z INSURE ARM PULSE SUBROUTINE IS
18  and       GO to the line called "SKIP"
19  and       END of test
20  and    END of test
21  and CARRIAGE return
22  and OUTPUT from "Wrist Error:  degrees = " using format DEFAULT
23  and OUTPUT from <2> using format DEFAULT
24  and HALT and return to Ready
25  and LABEL this line "SKIP"
26  and END   (Task is 24% full.)
```

```
WAIT this many seconds <1>  (a Function)

 1 Does IF the <1>  IS NOT  0 then
 2   Do    IF the <1>  IS LESS THAN OR EQUAL TO  1 then
 3   Do       MULTIPLY the <1> by 1,000
 4  and       PASTE the "DELAY " in front of PRODUCT
 5  and       Z COMPILE the BASIC statement FRONT which means "SHORT WAIT"
 6  and    OTHERWISE
 7   Do       SPLIT the number <1>
 8  and       PASTE the "T=" in front of INTEGER
 9  and       Z COMPILE the BASIC statement FRONT which means "NUMBER OF WHOL
ECONDS"
10  and       Z COMPILE the BASIC statement "GOSUB 3000" which means "GO WAIT"
11  and       Z INSURE WAIT SUBROUTINE IS
12  and       IF the FRACTION  IS NOT  0 then
13   Do          MULTIPLY the FRACTION by 1,000
14  and          TAKE this many characters 3 from PRODUCT
15  and          PASTE the "DELAY " in front of FRONT
16  and          Z COMPILE the BASIC statement FRONT which means "MILLISECOND
O WAIT"
17  and          END of test
18  and       END of test
19  and    END of test
20  and END   (Task is 21% full.)

WAIT RANDOMLY up to this many seconds <1>  (a Function)

 1 Does SPLIT the number <1>
 2  and PASTE the "W=" in front of INTEGER
 3  and Z COMPILE the BASIC statement FRONT which means "MAXIMUM SECONDS"
 4  and Z COMPILE the BASIC statement "GOSUB 3070" which means "GO WAIT RANDOML
"
 5  and Z INSURE RANDOM WAIT SUBROUTINE
 6  and END   (Task is 7% full.)

X ALPHA (a Task)

 1 Does PREPARE THE ROBOT
 2  and BEGIN A LOOP
 3  and PREPARE THE ROBOT
 4  and TURN ON THE FLASHING LIGHTS
 5  and BEGIN A LOOP
 6  and TURN ON THE FLASHING LIGHTS
 7  and MOVE FORWARD
 8  and EXIT IF ANY BUMPER TOUCHED
 9  and REPEAT THIS LOOP
10  and PREPARE THE ROBOT
11  and STOP ALL MOTION
12  and MOVE BACKWARD
13  and HONK the horn for this many seconds .5
14  and WAIT this many seconds 1
15  and STOP ALL MOTION
16  and MOVE RANDOMLY
17  and WAIT RANDOMLY up to this many seconds 5
18  and STOP ALL MOTION
19  and PREPARE THE ROBOT
20  and CLEAR ALL ITEMS
21  and REPEAT THIS LOOP
22  and END   (Task is 7% full.)
```

```
X ALPHA W/SONAR (a Task)

 1 Does PREPARE THE ROBOT
 2  and BEGIN A LOOP
 3  and PREPARE THE ROBOT
 4  and TURN ON THE FLASHING LIGHTS
 5  and BEGIN A LOOP
 6  and MOVE FORWARD
 7  and EXIT IF SONAR distance value is less than 95
 8  and EXIT IF ANY BUMPER TOUCHED
 9  and REPEAT THIS LOOP
10  and PREPARE THE ROBOT
11  and STOP ALL MOTION
12  and MOVE BACKWARD
13  and HONK the horn for this many seconds .5
14  and WAIT this many seconds 1
15  and STOP ALL MOTION
16  and MOVE RANDOMLY
17  and WAIT RANDOMLY up to this many seconds 5
18  and STOP ALL MOTION
19  and PREPARE THE ROBOT
20  and CLEAR ALL ITEMS
21  and REPEAT THIS LOOP
22  and END   (Task is 8% full.)

X BETA (a Task)

 1 Does PREPARE THE ROBOT
 2  and INITIALIZE MEMORY
 3  and TURN ON THE FLASHING LIGHTS
 4  and BEGIN A LOOP
 5  and EXIT IF ANY BUMPER TOUCHED
 6  and REPEAT THIS LOOP
 7  and CALL this robot task line "BEGIN BETA HERE"
 8  and PREPARE THE ROBOT
 9  and CLEAR ALL ITEMS
10  and TURN ON THE FLASHING LIGHTS
11  and BEGIN A LOOP
12  and MOVE FORWARD
13  and EXIT IF ANY BUMPER TOUCHED
14  and REPEAT THIS LOOP
15  and MOVE WITH BETA INTELLIGENCE
16  and JUMP to the line called "BEGIN BETA HERE"
17  and END   (Task is 8% full.)
```

```
X BETA W/SONAR (a Task)

 1 Does PREPARE THE ROBOT
 2  and INITIALIZE MEMORY
 3  and TURN ON THE FLASHING LIGHTS
 4  and BEGIN A LOOP
 5  and EXIT IF ANY BUMPER TOUCHED
 6  and REPEAT THIS LOOP
 7  and CALL this robot task line "BEGIN BETA HERE"
 8  and PREPARE THE ROBOT
 9  and TURN ON THE FLASHING LIGHTS
10  and CLEAR ALL ITEMS
11  and BEGIN A LOOP
12  and MOVE FORWARD
13  and EXIT IF SONAR distance value is less than 95
14  and EXIT IF ANY BUMPER TOUCHED
15  and REPEAT THIS LOOP
16  and MOVE WITH BETA INTELLIGENCE
17  and JUMP to the line called "BEGIN BETA HERE"
18  and END   (Task is 9% full.)

X BUMPER ACTIVATED MOTIONS (a Task)

 1 Does DEFINE A VARIABLE called "A"
 2  and PREPARE THE ROBOT
 3  and WAIT this many seconds 5
 4  and BEGIN A LOOP
 5  and HONK the horn for this many seconds 2
 6  and WAIT this many seconds 1.5
 7  and LIGHTS ROUTINE
 8  and MOVE TIMED FORWARD for this many seconds 1.5
 9  and MOVE TIMED BACKWARD for this many seconds 1.5
10  and GO CLOCKWISE
11  and WAIT this many seconds 1.5
12  and GO COUNTERCLOCKWISE
13  and WAIT this many seconds 1.5
14  and STOP ALL MOTION
15  and HONK the horn for this many seconds 2
16  and WAIT this many seconds 1.5
17  and BEGIN A COUNTED LOOP called "A" beginning at 1 ending at 16
18  and SET the variable "@#7802" equal to "A"
19  and WAIT this many seconds .5
20  and REPEAT THIS COUNTED LOOP for the counter "A"
21  and STOP ALL MOTION
22  and SET the variable "@#7801" equal to 0
23  and BEGIN A LOOP
24  and EXIT IF ANY BUMPER TOUCHED
25  and REPEAT THIS LOOP
26  and REPEAT THIS LOOP
27  and END   (Task is 17% full.)
```

```
X CHARGER FINDER (a Task)

 1 Does WAIT this many seconds 20
 2  and PREPARE THE ROBOT
 3  and INITIALIZE MEMORY
 4  and BEGIN A LOOP
 5  and PREPARE THE ROBOT
 6  and CLEAR ALL ITEMS
 7  and BEGIN A LOOP
 8  and MOVE FORWARD
 9  and EXIT IF ANY BUMPER TOUCHED
10  and EXIT IF THE TAPE IS SENSED
11  and REPEAT THIS LOOP
12  and TEST IF the variable "R" is (=,<,>,<>) "=" compared to 0
13  and EXIT THIS LOOP
14  and END HERE
15  and MOVE WITH BETA INTELLIGENCE
16  and REPEAT THIS LOOP
17  and FOLLOW TAPE
18  and MAINTAIN CHARGE
19  and END    (Task is 7% full.)

X LOAD THE INTRODUCTION (a Task)

 1 Does REMARK that: "RB5X say the introduction - new voice cards only!"
 2  and LOAD THE PHRASE called: "SAY THE INTRODUCTION" with phonemes: "H.EH1.EH
.L.O1.U1.AH1.EH3.I3.Y.PA1.PA1.AE1.EH3.M.THV.UH1.AH1.UH2.ER.B.E1.AY.Y.F.AH1.EH3.
.V.PA1.EH1.EH2.K.PA0.S.I1.N.T.EH1.EH3.L.I1.I3.D.J.EH1.EH3.N.T.R.O1.U1.B.AH1.UH
T.STOP."
 3  and END    (Task is 25% full.)

X SIMPLE SIMON (a Task)

 1 Does BEGIN A LOOP
 2  and PREPARE THE ROBOT
 3  and INITIALIZE VARIABLES
 4  and BEGIN A LOOP
 5  and EXIT IF FRONT+REAR BUMPER PRESS
 6  and REPEAT THIS LOOP
 7  and TURN ON THE FLASHING LIGHTS
 8  and HONK the horn for this many seconds .1
 9  and BEGIN A LOOP
10  and BEGIN A LOOP
11  and EXIT IF ANY BUMPER TOUCHED
12  and REPEAT THIS LOOP
13  and EXIT IF FRONT+REAR BUMPER PRESS
14  and WAIT this many seconds .2
15  and HONK the horn for this many seconds .1
16  and ASSIGN A MOTOR CODE
17  and TEST IF the variable "M" is (=,<,>,<>) ">" compared to "0"
18  and EXIT THIS LOOP
19  and END HERE
20  and CALCULATE variable "M" = "M" (+,-,*,/) "+"  1
21  and REPEAT THIS LOOP
22  and WAIT this many seconds 3
23  and BEGIN A COUNTED LOOP called "P" beginning at "N" ending at "M"
24  and SET the variable "@#7802" equal to "@P"
25  and WAIT this many seconds 1
26  and REPEAT THIS COUNTED LOOP for the counter "P"
27  and CLEAR ALL ITEMS
```

```
X SIMPLE SIMON (a Task)

28  and REPEAT THIS LOOP
29  and END    (Task is 16% full.)

Z COMPILE the BASIC statement <1> which means <2>  (a Function)

 1 Does Z INCREMENT STATEMENT NUMBER
 2  and COPY from <1> to BASIC TEXT
 3  and COPY from <2> to MEANING
 4  and SAVE new page in SOURCE
 5  and END    (Task is 2% full.)

Z COMPUTE 2 TO THE N-1 , with N = <1>  (a Function)

 1 Does COPY from 1 to PRODUCT
 2  and COPY from <1> to DIFFERENCE
 3  and START a loop
 4   Do     IF the DIFFERENCE  IS LESS THAN OR EQUAL TO  1 then
 5   Do        LEAVE this loop
 6  and     OTHERWISE
 7   Do        MULTIPLY the PRODUCT by 2
 8  and        SUBTRACT the 1 from DIFFERENCE
 9  and        END of test
10  and     REPEAT this loop
11  and END    (Task is 5% full.)

Z DELAY for this many loops:  <1>  (a Function)

 1 Does COPY from 0 to COUNTER
 2  and START a loop
 3   Do     Z INCREMENT the COUNTER
 4  and     IF the COUNTER  IS  <1> then
 5   Do        LEAVE this loop
 6  and        END of test
 7  and     REPEAT this loop
 8  and END    (Task is 3% full.)

Z DO CHECK if item <1> is in folder <2>  (a Function)

 1 Does ASSIGN the <2> to ASSIGNED FOLDER
 2  and WHEN folder ASSIGNED FOLDER  DOES NOT CONTAIN  <1> then
 3   Do     COPY from "FAIL" to RESULT
 4  and     COPY from <1> to ITEM NAME
 5  and     COPY from <2> to FOLDER NAME
 6  and OTHERWISE
 7   Do     COPY from "PASSED" to RESULT
 8  and     END of test
 9  and END    (Task is 5% full.)
```

```
Z DO NUMBER TEST on <1>  (a Function)

 1 Does ADD the 0 and <1>
 2  and IF the SUM  IS  ERROR then
 3   Do    COPY from "FAIL" to RESULT
 4  and    COPY from <1> to ITEM NAME
 5  and OTHERWISE
 6   Do    COPY from "PASSED" to RESULT
 7  and    END of test
 8  and END   (Task is 5% full.)

Z EMPTY THE PROGRAM (a Task)

 1 Does ERASE all pages in folder SOURCE
 2  and ERASE all pages in folder LOOPS
 3  and ERASE all pages in folder VARIABLES
 4  and ERASE all pages in folder ERRORS
 5  and ERASE all pages in folder STRUCTURES
 6  and ERASE all pages in folder SPEECH LOAD DATA
 7  and Z INITIALIZE LEGAL VARIABLES
 8  and COPY from 0 to STATEMENT NUMBER
 9  and COPY from NO to SONAR NEEDED
10  and COPY from 0 to ERROR NUMBER
11  and COPY from NO to CHECKED
12  and COPY from NO to VOICE NEEDED
13  and COPY from NO to SONAR NEEDED
14  and COPY from EMPTY to RESULT
15  and COPY from EMPTY to SPEECH ADDRESS
16  and END   (Task is 8% full.)

Z ERROR CHECK IF/UNTIL STATEMENT (a Task)

 1 Does Z DO CHECK if item <1> is in folder "VARIABLES"
 2  and IF the RESULT  IS  "FAIL" then
 3   Do    Z SAVE FOLDER ERROR
 4  and OTHERWISE
 5   Do    GET the page indexed by <1> in folder VARIABLES
 6  and    COPY from VAR NAME to <1>
 7  and    END of test
 8  and Z DO CHECK if item <2> is in folder "CONDITIONS"
 9  and IF the RESULT  IS  "FAIL" then
10   Do    Z SAVE FOLDER ERROR
11  and OTHERWISE
12   Do    GET the page indexed by <2> in folder CONDITIONS
13  and    IF the TASK NAME  IS  "TEST IF" then
14   Do       COPY from OPPOSITE to <2>
15  and       END of test
16  and    END of test
17  and Z DO NUMBER TEST on <3>
18  and IF the RESULT  IS  "FAIL" then
19   Do    Z DO CHECK if item <3> is in folder "VARIABLES"
20  and    IF the RESULT  IS  "FAIL" then
21   Do       Z SAVE NUMBER OR FOLDER ERROR
22  and    OTHERWISE
23   Do       GET the page indexed by <3> in folder VARIABLES
24  and       COPY from VAR NAME to <3>
25  and       END of test
26  and    END of test
27  and END   (Task is 19% full.)
```

```
Z ERROR CHECK MATH FUNCTION (a Task)

 1 Does Z DO CHECK if item <1> is in folder "VARIABLES"
 2  and IF the RESULT  IS  "FAIL" then
 3   Do    Z SAVE FOLDER ERROR
 4  and OTHERWISE
 5   Do    GET the page indexed by <1> in folder VARIABLES
 6  and    COPY from VAR NAME to <1>
 7  and    END of test
 8  and Z DO NUMBER TEST on <2>
 9  and IF the RESULT  IS  "FAIL" then
10   Do    Z DO CHECK if item <2> is in folder "VARIABLES"
11  and    IF the RESULT  IS  "FAIL" then
12   Do       Z SAVE NUMBER OR FOLDER ERROR
13  and    OTHERWISE
14   Do       GET the page indexed by <2> in folder VARIABLES
15  and       COPY from VAR NAME to <2>
16  and       END of test
17  and    END of test
18  and Z DO CHECK if item <3> is in folder "FUNCTIONS"
19  and IF the RESULT  IS  "FAIL" then
20   Do    Z SAVE FOLDER ERROR
21  and    END of test
22  and Z DO NUMBER TEST on <4>
23  and IF the RESULT  IS  "FAIL" then
24   Do    Z DO CHECK if item <4> is in folder "VARIABLES"
25  and    IF the RESULT  IS  "FAIL" then
26   Do       Z SAVE NUMBER OR FOLDER ERROR
27  and    OTHERWISE
28   Do       GET the page indexed by <4> in folder VARIABLES
29  and       COPY from VAR NAME to <4>
30  and       END of test
31  and    END of test
32  and END   (Task is 22% full.)

Z ERROR CHECK STRUCTURES (a Task)

 1 Does BEGIN at page indexed by FIRST in folder STRUCTURES
 2  and START a loop
 3   Do LABEL this line "START A LOOP"
 4  and    GET the page indexed by NEXT in folder STRUCTURES
 5  and    IF the PAGE  IS  EMPTY then
 6   Do       LEAVE this loop
 7  and       END of test
 8  and    IF the CHECKED  IS  YES then
 9   Do       GO to the line called "START A LOOP"
10  and       END of test
11  and    IF the STRUCTURE TYPE  IS  "REPEAT" then
12   Do       Z SAVE STRUCTURE ERROR missing the statement "BEGIN" to go with
e statment STRUCTURE TYPE at statement number STATEMENT NUMBER
13  and    OTHERWISE
14   Do       IF the STRUCTURE TYPE  IS  "END" then
15   Do          Z SAVE STRUCTURE ERROR missing the statement "IF OR ELSE" to
 with the statment STRUCTURE TYPE at statement number STATEMENT NUMBER
16  and       OTHERWISE
17   Do          IF the STRUCTURE TYPE  IS  "NEXT" then
18   Do             Z SAVE STRUCTURE ERROR missing the statement "FOR" to go wi
h the statment STRUCTURE TYPE at statement number STATEMENT NUMBER
19  and          OTHERWISE
20   Do             IF the STRUCTURE TYPE  IS  "UNTIL" then
```

```
Z ERROR CHECK STRUCTURES (a Task)

21   Do                    Z SAVE STRUCTURE ERROR missing the statement "DO" to go
ith the statment STRUCTURE TYPE at statement number STATEMENT NUMBER
22  and                 OTHERWISE
23   Do                    IF the STRUCTURE TYPE  IS  "EXIT" then
24   Do                       Z LOOK FOR BEGIN
25  and                    OTHERWISE
26   Do                       GET the page indexed by STRUCTURE TYPE in folder STA
MENT TYPES
27  and                       Z LOOK FOR MATCHING STATEMENT
28  and                       END of test
29  and                    END of test
30  and                 END of test
31  and              END of test
32  and           END of test
33  and       REPEAT this loop
34  and END    (Task is 25% full.)

Z ERROR CHECK THE FOR STATEMENT (a Task)

 1 Does Z DO CHECK if item <1> is in folder "VARIABLES"
 2  and IF the RESULT  IS  "FAIL" then
 3   Do     Z SAVE FOLDER ERROR
 4  and OTHERWISE
 5   Do     GET the page indexed by <1> in folder VARIABLES
 6  and     COPY from VAR NAME to <1>
 7  and     END of test
 8  and Z DO NUMBER TEST on <2>
 9  and IF the RESULT  IS  "FAIL" then
10   Do     Z DO CHECK if item <2> is in folder "VARIABLES"
11  and     IF the RESULT  IS  "FAIL" then
12   Do        Z SAVE NUMBER OR FOLDER ERROR
13  and     OTHERWISE
14   Do        GET the page indexed by <2> in folder VARIABLES
15  and        COPY from VAR NAME to <2>
16  and        END of test
17  and     END of test
18  and Z DO NUMBER TEST on <3>
19  and IF the RESULT  IS  "FAIL" then
20   Do     Z DO CHECK if item <3> is in folder "VARIABLES"
21  and     IF the RESULT  IS  "FAIL" then
22   Do        Z SAVE NUMBER OR FOLDER ERROR
23  and     OTHERWISE
24   Do        GET the page indexed by <3> in folder VARIABLES
25  and        COPY from VAR NAME to <3>
26  and        END of test
27  and     END of test
28  and END    (Task is 19% full.)
```

```
Z ERROR CHECKING? (a Task)

 1 Does CARRIAGE return
 2  and DISPLAY the "Would you like your robot script checked for errors? (Y/
"
 3  and LIMIT the input for ERROR CHECK to a maximum of 1
 4  and END   (Task is 7% full.)

Z FIND the label called <1>  (a Function)

 1 Does BEGIN at page indexed by FIRST in folder LOOPS
 2  and START a loop
 3   Do    GET the page indexed by NEXT in folder LOOPS
 4  and    IF the PAGE  IS  EMPTY then
 5   Do       IF the ERROR CHECK  IS  YES then
 6   Do          Z SAVE STRUCTURE ERROR missing the statement "CALL" to go wi
the statment "JUMP" at statement number TEMP STATEMENT NUMBER
 7  and          LEAVE this loop
 8  and          END of test
 9  and       LEAVE this loop
10  and       END of test
11  and    TAKE this many characters 1 from STRUCTURE TYPE
12  and    IF the FRONT  IS  "L" then
13   Do       IF the BACK  IS  <1> then.
14   Do          COPY from STATEMENT NUMBER to LABEL STATEMENT NUMBER
15  and          GET the page indexed by TEMP STATEMENT NUMBER in folder SOUR
16  and          CUT the BASIC TEXT after "GOTO "
17  and          PASTE the FRONT in front of LABEL STATEMENT NUMBER
18  and          COPY from FRONT to BASIC TEXT
19  and          REPLACE this page in folder SOURCE
20  and          LEAVE this loop
21  and          END of test
22  and       END of test
23  and    REPEAT this loop
24  and END   (Task is 14% full.)

Z GET PREVIOUS BEGIN (a Task)

 1 Does GET the page indexed by THIS in folder LOOPS
 2  and IF the STRUCTURE TYPE  IS NOT  "BEGIN" then
 3   Do    START a loop
 4   Do       GET the page indexed by PREVIOUS in folder LOOPS
 5  and       IF the PAGE  IS  EMPTY then
 6   Do          LEAVE this loop
 7  and          END of test
 8  and       IF the STRUCTURE TYPE  IS  "BEGIN" then
 9   Do          LEAVE this loop
10  and          END of test
11  and       REPEAT this loop
12  and    END of test
13  and END   (Task is 6% full.)
```

```
Z GET PREVIOUS IF OR ELSE (a Task)

 1 Does COPY from EMPTY to STRUCTURE TYPE
 2  and GET the page indexed by THIS in folder LOOPS
 3  and IF the STRUCTURE TYPE  IS NOT  "IF" then
 4   Do    IF the STRUCTURE TYPE  IS NOT  "ELSE" then
 5   Do       START a loop
 6   Do          GET the page indexed by PREVIOUS in folder LOOPS
 7  and          IF the PAGE  IS  EMPTY then
 8   Do             LEAVE this loop
 9  and          OTHERWISE
10   Do             IF the STRUCTURE TYPE  IS  "IF" then
11   Do                LEAVE this loop
12  and             OTHERWISE
13   Do                IF the STRUCTURE TYPE  IS  "ELSE" then
14   Do                   LEAVE this loop
15  and                   END of test
16  and                END of test
17  and             END of test
18  and          REPEAT this loop
19  and       END of test
20  and    END of test
21  and END   (Task is 10% full.)

Z INCLUDE at <1> the statement <2> which means <3>  (a Function)

 1 Does COPY from <1> to STATEMENT NUMBER
 2  and COPY from <2> to BASIC TEXT
 3  and COPY from <3> to MEANING
 4  and SAVE new page in SOURCE
 5  and END   (Task is 2% full.)

Z INCREMENT the <1>  (a Function)

 1 Does ADD the 1 and <1>
 2  and COPY from SUM to <1>
 3  and END   (Task is 1% full.)

Z INCREMENT STATEMENT NUMBER (a Task)

 1 Does ADD the STATEMENT NUMBER and STATEMENT NUMBER INCREMENT VALUE
 2  and COPY from SUM to STATEMENT NUMBER
 3  and END   (Task is 1% full.)
```

```
Z INITIALIZE LEGAL VARIABLES (a Task)

 1 Does BEGIN at page indexed by FIRST in folder LEGAL VARIABLES
 2  and START a loop
 3   Do .   GET the page indexed by NEXT in folder LEGAL VARIABLES
 4  and     IF the PAGE  IS  EMPTY then
 5   Do        LEAVE this loop
 6  and        END of test
 7  and     IF the USED  IS  NO then
 8   Do        LEAVE this loop
 9  and     OTHERWISE
10   Do        COPY from NO to USED
11  and        REPLACE this page in folder LEGAL VARIABLES
12  and        END of test
13  and     REPEAT this loop
14  and END    (Task is 6% full.)

Z INSURE ARM PULSE SUBROUTINE IS (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,400 then
 2   Do     COPY from STATEMENT NUMBER to HERE
 3  and     Z INCLUDE at 3,400 the statement "DELAY 1000:@#7807=#90" which mea
"SET STATUS BYTE"
 4  and     Z INCLUDE at 3,410 the statement "@#7806=#70" which means "ARM POWER
ON"
 5  and     Z INCLUDE at 3,420 the statement "@#7806=#F0" which means "SET ARM
R DRIVE"
 6  and     Z INCLUDE at 3,430 the statement "FOR I=1 TO P" which means "BEGIN A
LOOP"
 7  and     Z INCLUDE at 3,440 the statement "@D=M:@D=N" which means "PULSE THI
RM"
 8  and     Z INCLUDE at 3,450 the statement "NEXT I" which means "REPEAT THIS
OP"
 9  and     Z INCLUDE at 3,460 the statement "@#7807=#90:@#7806=#B0" which mea
"TURN OFF ARM POWER"
10  and     Z INCLUDE at 3,470 the statement "RETURN" which means "RETURN TO TH
MAIN PROGRAM"
11  and     COPY from HERE to STATEMENT NUMBER
12  and     END of test
13  and END    (Task is 36% full.)

Z INSURE BETA SUBROUTINE IS (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,160 then
 2   Do     COPY from STATEMENT NUMBER to ANSWER
 3  and     Z INCLUDE at 3,160 the statement "P=@#7800" which means "BUMPER PRESS
ED"
 4  and     Z INCLUDE at 3,170 the statement "IF P=255 THEN P=251" which means
ONAR TREATED AS BUMPER #1"
 5  and     Z INCLUDE at 3,180 the statement "V=@(N+P)" which means "CHECK EXPERI
ENCE BLOCK"
 6  and     Z INCLUDE at 3,190 the statement "IF V<>#FF GOTO 3220" which means
RY ACTION"
 7  and     Z INCLUDE at 3,200 the statement "V=RND(1,14)" which means "PICK RA
OM ACTION"
 8  and     Z INCLUDE at 3,210 the statement "IF (V=3) OR (V=12) OR (V=(O+P)) GO
O 3200" which means "PICK ANOTHER"
 9  and     Z INCLUDE at 3,220 the statement "@#7802=V" which means "TRY ACTION
10  and     WAIT this many seconds 2
11  and     IF the SONAR NEEDED  IS  YES then
```

```
Z INSURE BETA SUBROUTINE IS (a Task)

12   Do       Z COMPILE the BASIC statement "D=0" which means "DISTANCE EQUAL
 ZERO"
13  and       Z COMPILE the BASIC statement "LINK #1800" which means "CALL SOI
"
14  and       Z COMPILE the BASIC statement "@#7802=V" which means "CONTINUE /
ION"
15  and    OTHERWISE
16   Do       Z COMPILE the BASIC statement "D=200" which means "CANCEL SONAR
ST"
17  and       END of test
18  and    Z COMPILE the BASIC statement "IF (@#7800<>#FF) OR (D<95) GOTO 336(
which means "NOT SUCCESSFUL"
19  and    Z COMPILE the BASIC statement "@(N+P)=V" which means "SUCCESSFUL"
20  and    Z COMPILE the BASIC statement "CLEAR" which means "CLEAR ALL ITEMS'
21  and    PASTE the "GOTO " in front of RETURN STATEMENT NUMBER
22  and    Z COMPILE the BASIC statement FRONT which means "BACK TO MAIN PROGF
"
23  and    Z INCLUDE at 3,360 the statement "@(0+P)=V" which means "NOT SUCCES
UL"
24  and    Z INCLUDE at 3,370 the statement "GOTO 3200" which means ""
25  and    COPY from ANSWER to STATEMENT NUMBER
26  and    END of test
27  and END   (Task is 71% full.)

Z INSURE END OF PROGRAM IS THERE (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  2,999 then
 2   Do    END OF PROGRAM.
 3  and    END of test
 4  and END   (Task is 1% full.)

Z INSURE PORT 7801 SUBROUTINE IS (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,100 then
 2   Do    COPY from STATEMENT NUMBER to HERE
 3  and    Z INCLUDE at 3,100 the statement "U=@#7801" which means "TURN ON A
T"
 4  and    Z INCLUDE at 3,105 the statement "U=U OR X" which means ""
 5  and    Z INCLUDE at 3,110 the statement "@#7801=U" which means ""
 6  and    Z INCLUDE at 3,115 the statement "RETURN" which means ""
 7  and    Z INCLUDE at 3,120 the statement "U=@#7801" which means "TURN OFF A
IT @#7801"
 8  and    Z INCLUDE at 3,125 the statement "U=U AND X" which means ""
 9  and    Z INCLUDE at 3,130 the statement "@#7801=U" which means ""
10  and    Z INCLUDE at 3,135 the statement "RETURN" which means ""
11  and    COPY from HERE to STATEMENT NUMBER
12  and    END of test
13  and END   (Task is 24% full.)
```

```
Z INSURE RANDOM TURN SUBROUTINE (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,040 then
 2   Do    COPY from STATEMENT NUMBER to HERE
 3  and    Z INCLUDE at 3,040 the statement "V=RND(1,14)" which means "RANDOM TU
RN SUBROUTINE"
 4  and    Z INCLUDE at 3,045 the statement "IF (V=3) OR (V=12) GOTO 3040" whi
means ""
 5  and    Z INCLUDE at 3,050 the statement "RETURN" which means ""
 6  and    COPY from HERE to STATEMENT NUMBER
 7  and    END of test
 8  and END   (Task is 13% full.)

Z INSURE RANDOM WAIT SUBROUTINE (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,070 then
 2   Do    COPY from STATEMENT NUMBER to HERE
 3  and    Z INCLUDE at 3,070 the statement "T=RND(1,W)" which means "RANDOM W
T SUBROUTINE"
 4  and    Z INCLUDE at 3,075 the statement "GOSUB 3000" which means ""
 5  and    Z INCLUDE at 3,080 the statement "RETURN" which means ""
 6  and    COPY from HERE to STATEMENT NUMBER
 7  and    Z INSURE WAIT SUBROUTINE IS
 8  and    END of test
 9  and END   (Task is 12% full.)

Z INSURE VOICE SUBROUTINE IS (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,500 then
 2   Do    COPY from STATEMENT NUMBER to HERE
 3  and    Z INCLUDE at 3,500 the statement "V=TOP+500+A:@V=P" which means "STOR
E THE PHONEME"
 4  and    Z INCLUDE at 3,510 the statement "A=A+1" which means ""
 5  and    Z INCLUDE at 3,520 the statement "RETURN" which means ""
 6  and    COPY from HERE to STATEMENT NUMBER
 7  and    END of test
 8  and END   (Task is 11% full.)

Z INSURE WAIT SUBROUTINE IS (a Task)

 1 Does WHEN folder SOURCE  DOES NOT CONTAIN  3,000 then
 2   Do    COPY from STATEMENT NUMBER to HERE
 3  and    Z INCLUDE at 3,000 the statement "FOR S=1 TO T" which means "WAIT T S
ECONDS SUBROUTINE"
 4  and    Z INCLUDE at 3,005 the statement "DELAY 1000" which means ""
 5  and    Z INCLUDE at 3,010 the statement "NEXT S" which means ""
 6  and    Z INCLUDE at 3,015 the statement "RETURN" which means ""
 7  and    COPY from HERE to STATEMENT NUMBER
 8  and    END of test
 9  and END   (Task is 14% full.)
```

```
Z LOAD SONAR (a Task)

 1 Does DISPLAY the "Now loading the sonar subroutine..."
 2  and PRINT the ""
 3  and BEGIN at page indexed by FIRST in folder SONAR
 4  and START a loop
 5   Do    GET the page indexed by NEXT in folder SONAR
 6  and    IF the PAGE  IS  EMPTY then
 7   Do       LEAVE this loop
 8  and       END of test
 9  and    PASTE the SONAR ID in front of " "
10  and    PASTE the FRONT in front of BASIC TEXT
11  and    OUTPUT from FRONT using format DEFAULT
12  and    CARRIAGE return
13  and    DISPLAY the "$"
14  and    PRINT the ""
15  and    Z DELAY for this many loops:  5
16  and    REPEAT this loop
17  and Z DELAY for this many loops:  10
18  and PRINT the ""
19  and OUTPUT from "RUN" using format DEFAULT
20  and CARRIAGE return
21  and Z DELAY for this many loops:  50
22  and OUTPUT from "NEW#1841" using format DEFAULT
23  and CARRIAGE return
24  and Z DELAY for this many loops:  60
25  and OUTPUT from "NEW" using format DEFAULT
26  and CARRIAGE return
27  and DISPLAY the ""
28  and CARRIAGE return
29  and PRINT the ""
30  and END   (Task is 17% full.)

Z LOAD VOICE SUBROUTINE (a Task)

 1 Does CARRIAGE return
 2  and DISPLAY the "Now loading the voice subroutine..."
 3  and CARRIAGE return
 4  and PRINT the ""
 5  and BEGIN at page indexed by FIRST in folder VOICE SUBROUTINE
 6  and START a loop
 7   Do    GET the page indexed by NEXT in folder VOICE SUBROUTINE
 8  and    IF the PAGE  IS  EMPTY then
 9   Do       LEAVE this loop
10  and       END of test
11  and    OUTPUT from VOICE ID using format DEFAULT
12  and    CARRIAGE return
13  and    DISPLAY the "*"
14  and    PRINT the ""
15  and    Z DELAY for this many loops:  10
16  and    REPEAT this loop
17  and Z DELAY for this many loops:  10
18  and Z INSURE VOICE SUBROUTINE IS
19  and PRINT the ""
20  and OUTPUT from "@#2000=#03:@#2001=#0F:@#2002=#35:@#2003=#23:@#2004=#09" us
ng format DEFAULT
21  and CARRIAGE return
22  and Z DELAY for this many loops:  10
23  and OUTPUT from "@#2005=#21:@#2006=#1F:@#2007=#18:@#2008=#35:@#2009=#37:@#2(
0A=#1E" using format DEFAULT
```

```
Z LOAD VOICE SUBROUTINE (a Task)

24  and CARRIAGE return
25  and Z DELAY for this many loops:  10
26  and OUTPUT from "@#200B=#09:@#200C=#1E:@#200D=#3F:@#780B=#A7:S=#2000:LINK #
F50" using format DEFAULT
27  and CARRIAGE return
28  and Z DELAY for this many loops:  10
29  and DISPLAY the ""
30  and CARRIAGE return
31  and PRINT the ""
32  and END   (Task is 33% full.)

Z LOOK FOR BEGIN (a Task)

 1 Does COPY from STATEMENT NUMBER to HERE
 2  and START a loop
 3   Do    GET the page indexed by PREVIOUS in folder STRUCTURES
 4  and    IF the PAGE  IS  EMPTY then
 5   Do       GET the page indexed by HERE in folder STRUCTURES
 6  and       Z SAVE STRUCTURE ERROR missing the statement "BEGIN" to go with th
e statment "EXIT" at statement number HERE
 7  and       LEAVE this loop
 8  and       END of test
 9  and    IF the STRUCTURE TYPE  IS  "BEGIN" then
10   Do       GET the page indexed by HERE in folder STRUCTURES
11  and       COPY from YES to CHECKED
12  and       REPLACE this page in folder STRUCTURES
13  and       LEAVE this loop
14  and       END of test
15  and    REPEAT this loop
16  and END   (Task is 9% full.)

Z LOOK FOR MATCHING STATEMENT (a Task)

 1 Does COPY from STATEMENT NUMBER to HERE
 2  and START a loop
 3   Do    GET the page indexed by NEXT in folder STRUCTURES
 4  and    IF the PAGE  IS  EMPTY then
 5   Do       GET the page indexed by HERE in folder STRUCTURES
 6  and       Z SAVE STRUCTURE ERROR missing the statement MATCHING STATEMENT
go with the statment STATEMENT TYPES ID at statement number STATEMENT NUMBER
 7  and       LEAVE this loop
 8  and       END of test
 9  and    IF the CHECKED  IS NOT  YES then
10   Do       IF the STRUCTURE TYPE  IS  MATCHING STATEMENT then
11   Do          COPY from YES to CHECKED
12  and          REPLACE this page in folder STRUCTURES
13  and          GET the page indexed by HERE in folder STRUCTURES
14  and          COPY from YES to CHECKED
15  and          REPLACE this page in folder STRUCTURES
16  and          LEAVE this loop
17  and          END of test
18  and       END of test
19  and    IF the STATEMENT TYPES ID  IS  "IF" then
20   Do       IF the STRUCTURE TYPE  IS  "ELSE" then
21   Do          GET the page indexed by HERE in folder STRUCTURES
22  and          Z SAVE STRUCTURE ERROR missing the statement MATCHING STATEME
 to go with the statment STATEMENT TYPES ID at statement number STATEMENT NUMB
```

```
Z LOOK FOR MATCHING STATEMENT (a Task)

23  and           LEAVE this loop
24  and           END of test
25  and        END of test
26  and     REPEAT this loop
27  and END    (Task is 16% full.)

Z OUTPUT PAGE NUMBER (a Task)

 1 Does PASTE the "page " in front of PAGE COUNT
 2  and OUTPUT from SPACES using format 36
 3  and OUTPUT from FRONT using format DEFAULT
 4  and CARRIAGE return
 5  and CARRIAGE return
 6  and END    (Task is 2% full.)

Z OUTPUT PROGRAM HEADING (a Task)

 1 Does CARRIAGE return
 2  and OUTPUT from "     Stmt #" using format 10
 3  and OUTPUT from "   Tiny BASIC Text          " using format 30
 4  and OUTPUT from "        Meaning of Text" using format 23
 5  and CARRIAGE return
 6  and CARRIAGE return
 7  and END    (Task is 8% full.)

Z PATCH UP END STATEMENTS (a Task)

 1 Does COPY from STATEMENT NUMBER to END STATEMENT NUMBER
 2  and Z GET PREVIOUS IF OR ELSE
 3  and DELETE the page indexed by STATEMENT NUMBER from folder LOOPS
 4  and IF the STRUCTURE TYPE  IS  "IF" then
 5   Do     GET the page indexed by STATEMENT NUMBER in folder SOURCE
 6  and     CUT the BASIC TEXT after "GOTO "
 7  and     PASTE the FRONT in front of EXIT STATEMENT NUMBER
 8  and     COPY from FRONT to BASIC TEXT
 9  and     REPLACE this page in folder SOURCE
10  and OTHERWISE
11   Do     IF the STRUCTURE TYPE  IS  "ELSE" then
12   Do        SUBTRACT the 25 from STATEMENT NUMBER
13  and        COPY from DIFFERENCE to STATEMENT NUMBER
14  and        PASTE the "GOTO " in front of END STATEMENT NUMBER
15  and        Z COMPILE the BASIC statement FRONT which means "SKIP THE ELSE"
16  and        END of test
17  and     END of test
18  and COPY from END STATEMENT NUMBER to STATEMENT NUMBER
19  and END    (Task is 13% full.)
```

```
Z PATCH UP EXIT STATEMENTS (a Task)

 1 Does COPY from STATEMENT NUMBER to EXIT STATEMENT NUMBER
 2  and START a loop
 3   Do    GET the page indexed by NEXT in folder LOOPS
 4  and    IF the PAGE  IS  EMPTY then
 5   Do       LEAVE this loop
 6  and    OTHERWISE
 7   Do       IF the STRUCTURE TYPE  IS  "EXIT" then
 8   Do          GET the page indexed by STATEMENT NUMBER in folder SOURCE
 9  and          CUT the BASIC TEXT after "GOTO "
10  and          PASTE the FRONT in front of EXIT STATEMENT NUMBER
11  and          COPY from FRONT to BASIC TEXT
12  and          REPLACE this page in folder SOURCE
13  and          DELETE the page indexed by STATEMENT NUMBER from folder LOOPS
14  and       OTHERWISE
15   Do          LEAVE this loop
16  and          END of test
17  and       END of test
18  and    REPEAT this loop
19  and COPY from EXIT STATEMENT NUMBER to STATEMENT NUMBER
20  and END   (Task is 10% full.)

Z PATCH UP JUMP STATEMENTS (a Task)

 1 Does BEGIN at page indexed by FIRST in folder LOOPS
 2  and START a loop
 3   Do    GET the page indexed by NEXT in folder LOOPS
 4  and    IF the PAGE  IS  EMPTY then
 5   Do       LEAVE this loop
 6  and       END of test
 7  and    TAKE this many characters 1 from STRUCTURE TYPE
 8  and    IF the FRONT  IS  "J" then
 9   Do       COPY from STATEMENT NUMBER to TEMP STATEMENT NUMBER
10  and       COPY from BACK to LABEL NAME
11  and       Z FIND the label called LABEL NAME
12  and       GET the page indexed by TEMP STATEMENT NUMBER in folder LOOPS
13  and       DELETE the page indexed by THIS from folder LOOPS
14  and       END of test
15  and    REPEAT this loop
16  and END   (Task is 8% full.)

Z SAVE FOLDER ERROR (a Task)

 1 Does IF the ERROR CHECK  IS  YES then
 2   Do    ADD the STATEMENT NUMBER and 10
 3  and    COPY from SUM to LINE NUMBER
 4  and    PASTE the "Folder " in front of FOLDER NAME
 5  and    PASTE the FRONT in front of " doesn't contain the item "
 6  and    PASTE the FRONT in front of ITEM NAME
 7  and    COPY from FRONT to ERROR MESSAGE
 8  and    Z INCREMENT the ERROR NUMBER
 9  and    SAVE new page in ERRORS
10  and    END of test
11  and END   (Task is 9% full.)
```

```
Z SAVE NUMBER OR FOLDER ERROR (a Task)

 1 Does IF the ERROR CHECK  IS  YES then
 2   Do    ADD the STATEMENT NUMBER and 10
 3  and    COPY from SUM to LINE NUMBER
 4  and    PASTE the ITEM NAME in front of " should be a number or be containe
in the folder "
 5  and    PASTE the FRONT in front of FOLDER NAME
 6  and    COPY from FRONT to ERROR MESSAGE
 7  and    Z INCREMENT the ERROR NUMBER
 8  and    SAVE new page in ERRORS
 9  and    END of test
10  and END   (Task is 10% full.)

Z SAVE STRUCTURE ERROR missing the statement <1> to go with the statment <2> a
statement number <3>  (a Function)

 1 Does PASTE the "The " in front of <2>
 2  and PASTE the FRONT in front of " statement is missing the "
 3  and PASTE the FRONT in front of <1>
 4  and PASTE the FRONT in front of " statement"
 5  and COPY from FRONT to ERROR MESSAGE
 6  and COPY from <3> to LINE NUMBER
 7  and Z INCREMENT the ERROR NUMBER
 8  and SAVE new page in ERRORS
 9  and IF the <1>  IS NOT  "CALL" then
10   Do    COPY from YES to CHECKED
11  and    REPLACE this page in folder STRUCTURES
12  and    END of test
13  and END   (Task is 11% full.)

ZZ AVAIL TASK (a Task)

 1 Does END   (Task is 1% full.)

ZZ AVAIL TASK #1 (a Task)

 1 Does END   (Task is 1% full.)

ZZ AVAIL TASK #10 (a Task)

 1 Does END   (Task is 1% full.)
```

```
ZZ AVAIL TASK #11 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #12 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #13 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #14 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #2 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #3 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #4 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #5 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #6 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #7 (a Task)
 1 Does END   (Task is 1% full.)
ZZ AVAIL TASK #8 (a Task)
 1 Does END   (Task is 1% full.)
```