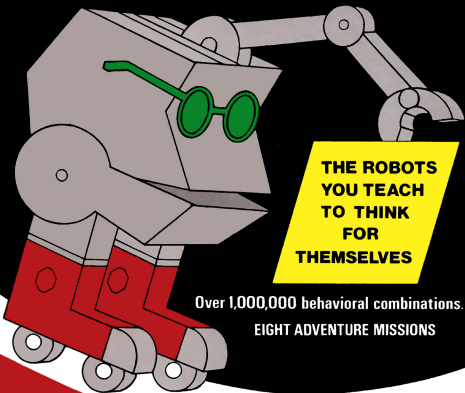


A challenge for
Ages 12 through adult

CHIPWITS™



**THE ROBOTS
YOU TEACH
TO THINK
FOR
THEMSELVES**

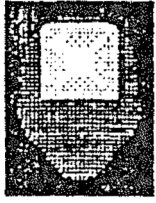
Over 1,000,000 behavioral combinations.

EIGHT ADVENTURE MISSIONS

DISKETTE and instructive user's manual included
No programming knowledge required

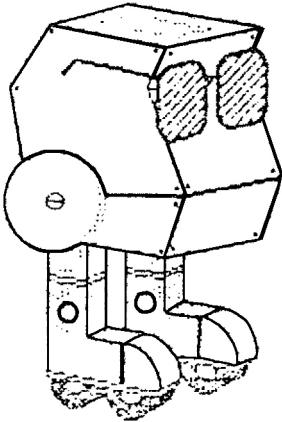
BRAINWORKS, INC.

24009 VENTURA BLVD. CALABASAS, CA 91302



BRAINWORKS INC.

24009 Ventura Boulevard
Suite 250
Calebasas, California 91302



Ready when you are.

CHAWITS™

by Doug Sharp and Mike Johnston

©1984 Discourse, Inc.

scan silicium.org 2015

ChipWits Table of Contents

Section 1: Using ChipWits	
Introduction to ChipWits	1
Using the Manual	1
Divisions	1
Topic Directory	1
Reference Card	2
Glossary	2
Using ChipWits	2
Start-Up Checklist	2
Start-Up	3
Running a Trial Mission	4
Inside an Environment	5
What's In a Chip	5
Environments	7
Putting It Together	7
Status and Memory Panel	11
The Debug Panel	13
The Workshop	14
Introduction	14
Getting Into the Workshop	14
A Tour of the Workshop	15
Control Panels	15
The Operator Menu	15
The Arguments Menu	16
The Wastebasket	16
Programming a ChipWit	16
Section Objectives	16
Getting Around the Workshop	16
The Chip Selection Window	17
Programming Procedures	18
Introduction	18
Communicating with Your ChipWit	19
Building a ChipWit	25

Using the Workshop	25
Modifying Existing Chips	26
Repeating the Program	26
SubPanel Programming	26
Clearing a Panel	27
Cutting, Copying, and Pasting Panels	28
Save a ChipWit	29
Section II: Advanced Programming	30
Objectives	30
The Key Press	30
Uses	30
Cycles	31
The CHIPWIT's Lifespan	31
On-Screen Cycle Count	33
Cycles and Scoring	33
Fuel	33
Damage	34
Conservation of CHIPWIT Resources	34
Memory Stacks	35
The Eight Environments	39

Section I: Using ChipWits

INTRODUCTION TO CHIPWITS

Robots are becoming a part of our everyday lives. Their presence has moved from science fiction to reality in only a few short years. Robots first made their appearance in steel mills and automobile factories. As they became more precise and intelligent in their operation, they made their way into new areas such as law enforcement and the exploration of dangerous environments. Now they are becoming available for the home in small but increasing numbers.

The CHIPWITS system gives you an opportunity to experiment with programming a robot for a variety of missions - and also lets you experiment with some new tools for solving thinking problems which you find in real life.

In summary, CHIPWITS is a robot-programming simulation and a tool for learning new problem solving skills.

USING THE MANUAL

This manual is designed to make it easy to use the CHIPWITS system. It will teach you techniques for solving new kinds of problems using the skills that you gain in programming **your** CHIPWITS.

Divisions

- Section I of the manual describes the CHIPWITS system in detail, the **Environments** in which CHIPWITS carry out **Mission Assignments**, and the Workshop in which the CHIPWITS are programmed to achieve their missions.

Section I: Using ChipWits

- Section II examines strategies which you may employ when programming your CHIPWITS to achieve the highest scores possible.

Reference Card

To help you program your CHIPWITS, there is a Quick Reference Card included which is designed to help you select the individual chips you use as you program each CHIPWIT.

Glossary and Index

You may find a number of terms unfamiliar to you. Use the Glossary to understand how each of these terms is used in the manual. To find an alphabetical listing of all Topic locations, use the Index at the back of this Manual.

USING CHIPWITS

After you read this section you will be able to:

- Use the **Workshop** to program up to sixteen separate CHIPWITS and prepare them for their Missions
- Enter a CHIPWIT of your choice into a Mission Environment and observe its performance
- Modify a CHIPWIT's programming to improve its performance

Startup Checklist

Before using the system you will need

- A 128k or 512k Macintosh with the mouse and keyboard connected. The Macintosh may either be turned on or turned off.

Section I: Using ChipWits

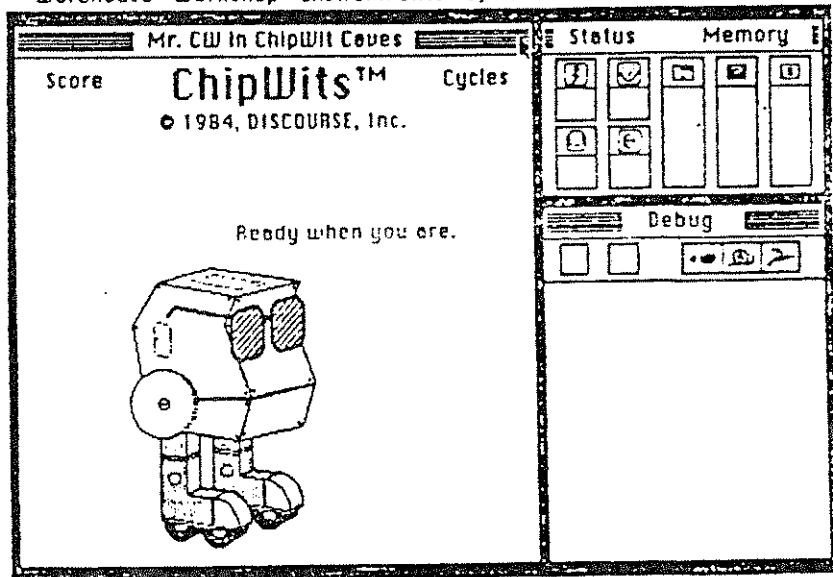
- An Imagewriter printer connected and turned on if you wish to print the **Workshop** and **Environments** screens. However, a printer is not required for using this system

Startup

Follow these steps to begin

1. Insert the disk in either the internal or external drive of the Macintosh.
2. If the power switch is off, turn it on.
3. The BRAINWORKS identification screen will come up. After a few moments, the CHIPWITS opening screen will appear.
4. It will take approximately twenty seconds of disk activity before the program is ready for you to begin.
5. When the CHIPWITS Startup screen appears, it looks like this:

Warehouse Workshop Environments Options



Section I: Using ChipWits

As you examine the screen you will see the **Menu Bar** across the top with the legends **Options**, **Environments**, **Workshop**, and **Warehouse**. Click on and pull down the **Warehouse** menu. Note the following:

- There is space for sixteen **CHIPWITS**. Some are already named. The rest are numbers.
- **CHIPWITS** which are named have been already built and tailored for a specific **Environment**. For each **Environment**, there is a **CHIPWIT** which has been constructed in the **Workshop** specifically for that **Environment**. Those which are numbered are currently unprogrammed and represent brand new **CHIPWITS**.
- The **CHIPWIT** name or number with the check mark is the **default** which will be used in the **Workshop** and the **Environment**, unless you select another.
- Named **CHIPWITS** are either in holdface or outline. The outlined names represent those **CHIPWITS** which you have designed specifically for the current **Environment**. To see which **Environment** is the current one, pull down the **Environments** menu and see which **Environment** is checked.
- **SAVE CHIPWITS** is used when you have finished designing, and testing your **CHIPWIT**, and you wish to **SAVE** it or store it away. Saving your **CHIPWIT** allows you to use it repeatedly.

Running a Trial Mission

Before trying to design your own **CHIPWIT**, you will want to see how a **CHIPWIT** behaves in an **Environment**. To run a trial mission, follow these steps:

1. Turn On the Macintosh.
2. Insert the **CHIPWITS DISKETTE** in either the internal or the external drive.
3. Wait approximately 20 seconds as the **BRAINWORKS** Screen comes up, and the **CHIPWITS** system loads.

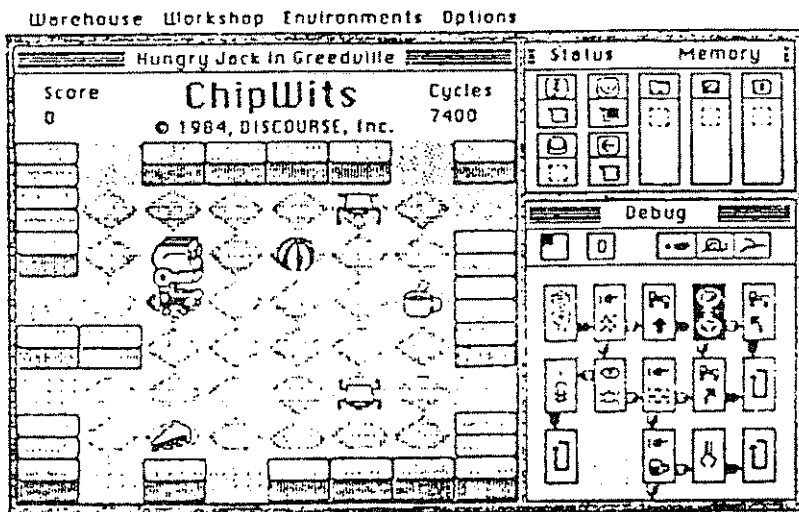
Section I: Using ChipWits

4. Pull down the Environments menu
5. Select Start Mission.
6. Observe the results of the mission.

INSIDE AN ENVIRONMENT

When you get your disk, the default Environment is called **Greedville**. The default CHIPWIT is named **Hungry Jack**. When the CHIPWITS screen first appears you will find the title is **Hungry Jack in Greedville**.

When you observe the mission, here are things that you should be looking for:



What's in a Chip

All CHIPWITS have these characteristics:

- Each chip contains an **Operator**. In computer terminology, this

Section I: Using ChipWits

is simply an **Action Command** that tells the CHIPWIT what to do. These operators include directions such as **LOOK [FOR]**, **SING [A NOTE]**, **SMELL [FOR]**, and **MOVE [TO]**. Each operator is represented by a small picture at the top of the chip. These pictures are called **ICONS**.



- With a few exceptions, **Operators need objects of their actions**. These objects, called **Arguments**, are shown in a box on the screen. Arguments include **THINGS** (OIL CANS, PIE, COFFEE, WALLS, FLOOR, ELECTRO-CRABS, DOORS, BOUNCERS, BOMBS, and DISKS), **MOVES** (FORWARD, REVERSE, LEFT 45 degrees, RIGHT 45 degrees), **NUMBERS** (0-7), **REGISTERS**, and **STACKS**. (more about these in Section II). Each Argument is represented by a small picture at the bottom of the **chip**.



- The operator and the argument make up the **Instruction Set** for each chip. When the CHIPWIT finishes carrying out an instruction, it needs to know what to do next. **Passing Control** to the next chip is handled by the **True/False Tabs** on the edges of the chips. This **chip**, for example, **instructs** a CHIPWIT to **look** directly ahead for a door:



In this case, the **True Tab** will pass control to the chip on the right if a door is seen. If there is no door ahead, or any other object is seen, the **False Tab** will pass control to the chip located directly below.

Section I: Using ChipWits

Environments

- Each Environment consists of a series of rooms, bounded by walls, and a floor. Walls are also constructed in the middle of the rooms.
- Openings in the outer walls are doorways. When a CHIPWIT reaches a doorway and moves through it, a new room is automatically displayed.
- Contained within the rooms are a variety of Things. These include: OIL CANS, PIE, COFFEE, WALLS, FLOOR, ELECTRO-CRABS, DOORS, BOUNCERS, BOMBS, and DISKS.

Putting it All Together

Each Environment has a distinct goal for the CHIPWIT to achieve (the **Mission**). In the earlier Environments, the CHIPWIT is faced with minor dangers like running out of energy or being damaged by running into things. In later Environments, CHIPWITS face hostile Things: ELECTRO-CRABS, BOUNCERS, BOMBS, etc. Here are the goals for each






Environment:

Name: Greedville

Objective: Acquire as many good Things as possible

Description: Four confusingly connected rooms tiled randomly with objects having only positive point values (good Things):

Point Assignment:

• 	60				
• 	30				

Cycles: 6,000






Section I: Using ChipWits

Name: ChipWit Caves

Objective: Acquire as many good Things as possible and zap Electro-Crabs

Description: Eight connected rooms (walks arranged to spell out the words "Chip Wits").

Point Assignment:

●		100	●		50	●		
●		40						






Cycles: 10,000

Name: Doom Rooms

Objective: Survive as long as possible

Description: Twelve interconnected rooms populated only with bad Things (i.e., Electro-Crabs, Bouncers, and Bombs).

Point Assignment:

			●		100	●		
			●		250			

Cycles: 10,000






Section I: Using ChipWits

Name: Peace Path

Objective: Acquire as many good Things as possible without harming Electric Crabs and Bouncers.

Description: Long connected hallway with isolated rooms on either side containing randomly distributed good and bad Things.

Point Assignment:

●		60	●		-30	●		
●		30	●		-60			






Cycles: 160,000

Name: Mystery Matrix

Objective: Discover the optimal pathway through the matrix by deciphering objects arranged as signposts.

Description: One hundred connected rooms, each with four identically located doors.

Point Assignment:

●		250	●		20	●		
●		150	●		40			

Cycles: 30,000






Section I: Using ChipWits

Name: Memory Lane

Objective: Move from one end to the other in a maze as many times as possible

Description: Ten room horseshoe maze populated with a variety of bad Things, and one high-point score at either end

Point Assignment:

●		250	●		30	●		
●		30	●		60			






Cycles: 20,000

Name: Octopus' Garden

Objective: Retrieve the good Thing located at the end of each arm of the maze

Description: Forty-four rooms divided in eight hallways which connect in only one central location

Point Assignment:

●		250	●		20	●		
●		50	●		40			

Cycles: 200,000






Section I: Using ChipWits

Name: Boom Town

Objective: Collect disks while avoiding the risk of sudden destruction.

Description: Bombs are clustered around disk. Pick up disks without triggering the bomb, which can instantly destroy the CHIPWIT.

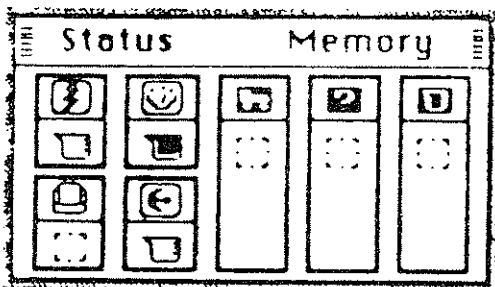
Point Assignment:

●		250	●		40	●		
								

Cycles: 12,000

Status and Memory Panels

While watching your CHIPWIT perform, you will need to have some information as to its health. This is provided through the Status Panel found at the top of the ENVIRONMENT screen.



Section I: Using ChipWits

This icon shows the damage your CHIPWIT has taken during its time in the environment. When the beaker under the icon is full, it indicates that the CHIPWIT has been damaged too severely to continue. The mission will end.



The CHIPWIT refuels itself when it eats a piece of pie or drinks a cup of coffee. If it does not find these Things, it will run out of fuel and come to rest in an **Out of Fuel** condition. The mission ends at that point. You can determine the fuel level by checking the beaker below the fuel icon.



After any **LOOK** command, if the CHIPWIT has located a Thing, the object's distance from the CHIPWIT will be indicated in the beaker below the Range Finder Icon.



The Keypress Icon shows the last key which has been pressed on the keyboard. This allows you to keep track of keys that you have previously used. Uses of the Keypress will be discussed in Section II.

The three columns that make up the **Memory Panel** represent **Stacks of Things, moves, and numbers** that have been stored for the CHIPWIT to remember under certain conditions. Their use will be fully discussed in Section II of this manual.

Section I: Using ChipWits

The Debug Panel

This panel is shown on the bottom right of the ENVIRONMENTS screen. There are several things to look at on the **Debug Panel**.

- First, notice that all the chips making up the CHIPWIT program for the currently loaded CHIPWIT are shown while the CHIPWIT moves through its mission. As each chip is used, it reverses color (black on white to white on black) and stays reversed until the next chip is executed. This lets you see if your plan is working as you intended.
- Since the action is fast, you may wish to slow it down. To slow down, point to the **Snail** icon and click the mouse once. This will cut the speed of the action significantly. To single step through the mission (each mouse click causes one command to be executed), point to the **Footprint** icon and click once. To resume full speed, point to the **Humming Bird** icon and click once.
- Because there is not enough room in the **Debug Panel** to display the entire active panel, the square in the upper left corner of the **Debug Panel** displays the quarter of the panel which is currently active. The adjacent box identifies the active panel. If there is a 0 in the box, then the **Main Panel** is being executed. If a letter is shown, it refers to one of the **Sub-Panels**.
- At the end of the mission, the **Debug Panel** is replaced with a **Status Panel**. Displayed on the **Status Panel** is the highest score made in a series of Missions and the average score over all Missions. This panel continues to add a complete record until you make a change in the **Workshop**. Any change made to the active CHIPWIT will zero out the scores in the **Mission Record** file.

Section I: Using ChipWits

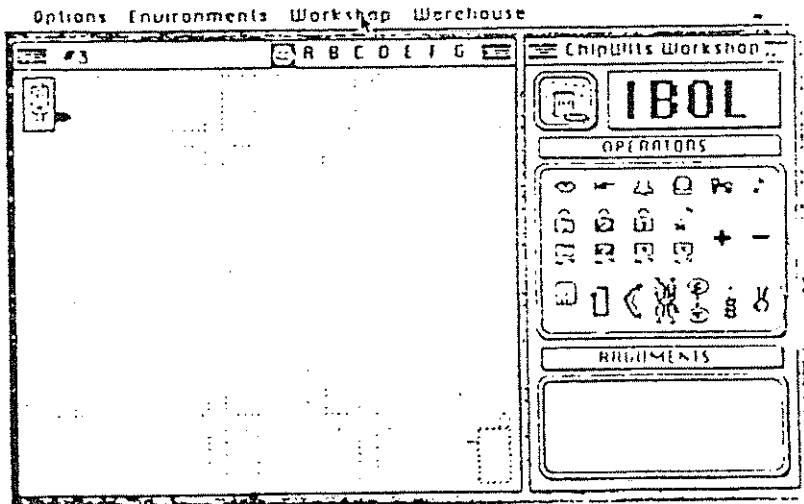
THE WORKSHOP

Introduction

The **WORKSHOP** is the heart of the **CHIPWITS System**. Here you program your **CHIPWIT** to prepare it for the specific **ENVIRONMENT** it is to master. You also return to the **WORKSHOP** when you want to modify or improve the performance of a **CHIPWIT**.

Getting into the Workshop

You may enter the Workshop at any time by simply pulling down the **WORKSHOP** menu and selecting the name of the **CHIPWIT** you wish to program. A previously unprogrammed **CHIPWIT** will have a number next to it. (You will name it later.) As soon as you have selected a **CHIPWIT**, the disk will spin for a few moments and you will see this display:



Section I: Using ChipWits

A Tour of the Workshop

To use the WORKSHOP, you need to know what each of its parts is designed to do.

Control Panels

A CHIPWIT is controlled by the IBOL program, which is "burned in" to the circuit chips on its **Main Panel** (or **Brain**) and seven **SubPanels** (labeled A through G). The chips contain instructions which occur in a sequence determined by the robot builder. This sequence is set by the position of the chips in relation to one another. The control of this sequence passes from one chip to the next through the True and False Tabs.

Main Panel/Sub-Panel

Main Panel/Sub-Panel Identification lets you know which of the eight available chip panels you are working with. The panel identification is found to the immediate right of the CHIPWIT name. The Main Panel is identified by a small face. Sub-Panels are identified by the letters A through G. The active panel is shown with a black box and a white icon. Inactive panels are black icons on a white background.

The Chip Panels

The Chip Panels are all identical except for identification. There is one Main Panel, and there are seven Sub-Panels. Each has space for sixty chips. In row 1, column 1 of each panel is a traffic light chip. You must begin programming your chips either in the space immediately to the right of, or immediately below, the traffic light

The Operator Menu

The Operator Menu allows you to choose which operator you are going to use for each chip. It is found directly below its title on the right side of the WORKSHOP screen.

Section I: Using ChipWits

The Arguments Menu

The **Arguments Menu** appears only after an Operator is selected. The Arguments which appear are the ones available for that Operator. In some cases, Operators do not take an Argument. In those instances, the Argument menu remains blank.

The Wastebasket

The **Wastebasket** is available to delete chips you wish to remove. An existing chip which is just going to be altered does not have to be sent to the Wastebasket. Simply click on the edge of the chip, and select the new Operator and Argument for it.

PROGRAMMING A CHIPWIT

Section Objectives:

After reading this section you will be able to

- Identify each of the parts of the Workshop
- Program a new CHIPWIT
- Modify the programming of an existing CHIPWIT

Getting Around in the Works

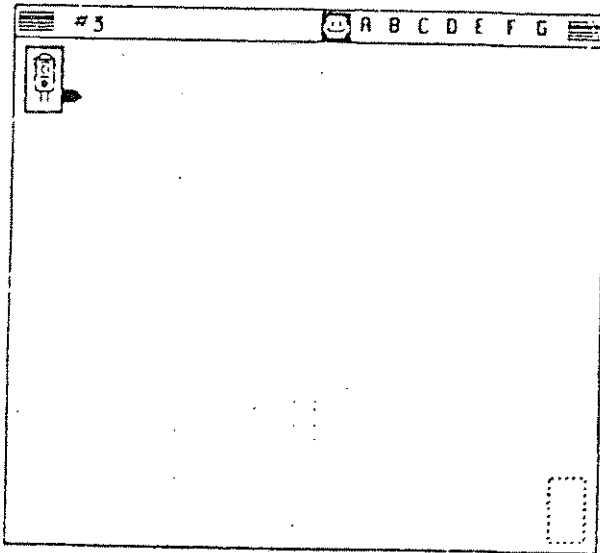
CHIPWIT Identification

The CHIPWIT being programmed is either a new unit or a previously programmed one which is now being modified. If the active CHIPWIT is new, its designation is a number which appears in the upper left corner of the WORKSHOP window. If the CHIPWIT was previously programmed, its name is found in the same location on the screen.

Section J: Using ChipWits

Panel Sub-Panel Designation

- CHIPWIT chips are "plugged in" to control panels in a sequence which you determine. Each panel contains room for sixty chips. There are eight panels available for your use. Sub-Panel chips on the Main Panel "call" the **Sub-Panels** as needed. (Sub-Panels can only be called from the Main Panel - not from another Sub-Panel.)



The Main Panel is represented in the identification strip as a CHIPWIT face icon. The Sub-Panels are named by individual letters A through G. The icon on the active panel is shown as white on a black background. These icons are shown at the top right of the panel window.

To change the active panel, point to the panel letter (or the face for the Main Panel), and click once. The default is the Main Panel.

The Chip Selection Window

Operators and Arguments

All of the **Operators** available to you are shown in the display under

Section I: Using ChipWits

the Operators label. Most, but not all, of the operators require an **Argument** to complete the programming of the chip. When an Operator is selected, the available arguments appear in the window below. If no Argument can be used with a given Operator, the Argument window remains blank.

The Wastebasket

There will be times when you wish to get rid of a chip. Click on the chip and then click on the Wastebasket icon found in the upper left corner of the Chip selection window.

PROGRAMMING PROCEDURE

Introduction

The fun and the challenge of the CHIPWIT system rests in your ability to program your CHIPWITS effectively to master the ENVIRONMENTS in which they are placed. After you read the Mission Description, you will discover several goals to achieve.

The Mission Objective

Each mission has a specific objective listed in its description. You will gain the highest scores when you program a CHIPWIT to be effective in a particular mission.

Hostile Things

BOMBS, **BOUNCERS**, and **ELECTRO-CRABS** are all hostile and dangerous to your CHIPWIT. Contact with them can either deplete the CHIPWIT's energy (fuel) or severely damage it. Therefore, you must look for ways to either avoid contact, or protect yourself from these bad Things.

Energy Conservation

Executing any operation requires fuel. Conservation of fuel requires that you give careful consideration to the best & most economical way of carrying out a complicated operation. There are many ways of doing the same thing, although some ways use more

Section I: Using ChipWits

energy than others. If you waste fuel (energy), or fail to refuel (eat pie or drink coffee), the mission will be aborted (ended) early. This will result in fewer points than you might otherwise have been able to achieve. On the other hand, you might be very cautious about using fuel but either run out of time (cycles) or fail to use your CHIPWIT effectively to gather points. A close look at the gains and losses of various alternatives is extremely important.

COMMUNICATING WITH YOUR CHIPWIT

Before you begin programming your CHIPWIT, you may want a better understanding of exactly what is happening when you provide it with its instructions. In many ways, you are giving instructions to the CHIPWIT just as you might to a small child. Suppose you were teaching a child to set the table for dinner. The instructions might sound something like this:

"Look in the cupboard for the plates."

The child finds the plates.

"Bring them to the table."

After the plates are deposited on the table, you say:

"Now look for the glasses."

This time he can't find them. You say:

"No, look in the cupboard on the left."

The instructions continue until the child has learned a routine which works for this particular situation.

By now you will have noticed that "programming" is broken into three distinct components: What CHIPWIT does (The Instruction Set), when it does it (The Sequence of Instructions), and did it accomplish its task? (Conditional Testing).

Section I: Using ChipWits

The Instruction Set

Each Operator represents an **action** to be taken by a CHIPWIT. It is the same as the action verb in an imperative sentence. As an example, take the sentence **[YOU are to] LOOK [for] PIE**.

When a sentence is spoke, or written in English we add the bracketed parts to make it clearer. In an imperative sentence, one that gives a command, the subject, **YOU**, is often left unsaid. Thus, we are used to hearing the sentence as: **LOOK [for] PIE**. The preposition **for** makes listening to the sentence easier and comfortable for people. CHIPWITS, being computer robots, do not require prepositions for communication. Thus, we put together the final sentence as **LOOK PIE**.

It is often said that "a picture is worth a thousand words." We substituted symbols, or pictures, to represent words. **THE SYMBOL** takes the place, or stands for the Operator **LOOK**. It is easy to understand, and it takes up much little room on the chip. The same thing is true of the Argument **PIE**.

The chart below lists each of the Operators, the icons they use, a description of their use, and the arguments each may employ.

OPERATOR	ICON	DESCRIPTION
ARGUMENTS		
LOOK	Look for a specific Thing directly ahead. If seen, branch True . If not seen, branch False .	Things
SMELL	Smell for a specific Thing within a room. If smelled, branch True . If not smelled, branch False .	Things
FEEL	Feel for a specific Thing within a room. If felt, branch True . If not felt, branch False .	Things
COMPARE KEY	Compare a keypress with a specific key (Argument). Branch True if the keypress and the key are the same. Branch False if they are not.	Keys

Section I: Using ChipWits

MOVE	Move forward, backward or turn 45°.	Moves
PICK UP	Pick up the Thing directly ahead	NONE
ZAP	Discharge plasma beam directly ahead	NONE
SING	Sing a Note.	Notes/Registers
COIN FLIP	Performs a random choice for a True or False branch.	None
SUBPANEL	Acts as connecting Operator the Main Panel and the SubPanels.	SubPanels A-G
LOOP	Return to beginning of the current SubPanel.	None
BOOMERANG	Return to Main Panel and continue execution.	None
SAVE THING	Push a Thing [put] onto the top of the Thing stack	Things
COMPARE THING	Compare with Thing on top of Thing stack and branch.	Things
SAVE MOVE	Push a Move [put] onto the top of the Move stack.	Moves
COMPARE MOVE	Compare with Move on top of Moves stack and branch	Moves
SAVE NUMBER	Push a Number [put] onto the top of the Number stack.	Numbers
COMPARE NUMBER =	Compare an Argument to a number on top of the Number stack. If they are equal, branch True. If not branch False.	Numbers Registers

Section I: Using ChipWits

COMPARE NUMBER	Compare an Argument to a number on top of the Number stack. If the argument is less than the number branch True . If not less than the number, branch False .	Numbers Registers
INCREMENT	Increase the Number stack by 1. (Numbers increase to 7 then go back to 0. This is called WRAP AROUND .)	None
DECREMENT	Decrease the Number stack by 1. (Numbers decrease to 0 then go back to 7. This is called WRAP AROUND .)	None
POP	Remove the top element from a stack	Any Stack
JUNCTION	Continue execution in direction of output wire	None

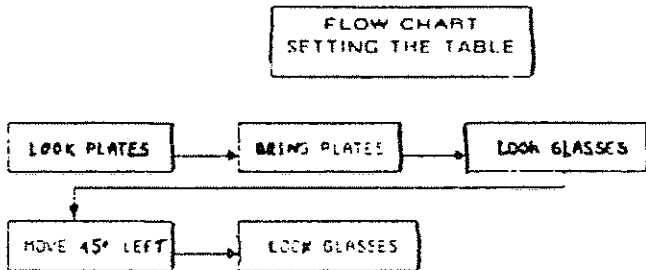
Sequence of Instructions

Looking at the set of instructions we gave the child earlier, you see that if our commands were programmed in chips, they would look like this:

1. LOOK PLATES
2. BRING PLATES
3. LOOK GLASSES
4. MOVE 45° LEFT
5. LOOK GLASSES

Section I: Using ChipWits

You will note that the chips are programmed to perform in a very specific order. CHIPWIT programming, like all other computer programming, requires that the instructions be placed in a sequence to allow the program to achieve its objectives for you. Programmers often use **Flow Charts** to help them figure out their best moves. The example we have been using is charted below.



Each rectangle represents an Operator and an Argument (one instruction set). The arrows and positions of the rectangles point out the sequence in which the instructions are carried out. As you develop your CHIPWIT programs, you may want to sketch out simple flow charts for yourself. Later we will tell you how to move your chips around on the panel to rearrange sequences of instructions.

Conditional Testing

Review the instruction set that we used before. You will note that the child ran into a problem:

1. LOOK PLATES
2. BRING PLATES
3. LOOK GLASSES
4. MOVE 45° LEFT
5. LOOK GLASSES

Section I: Using ChipWits

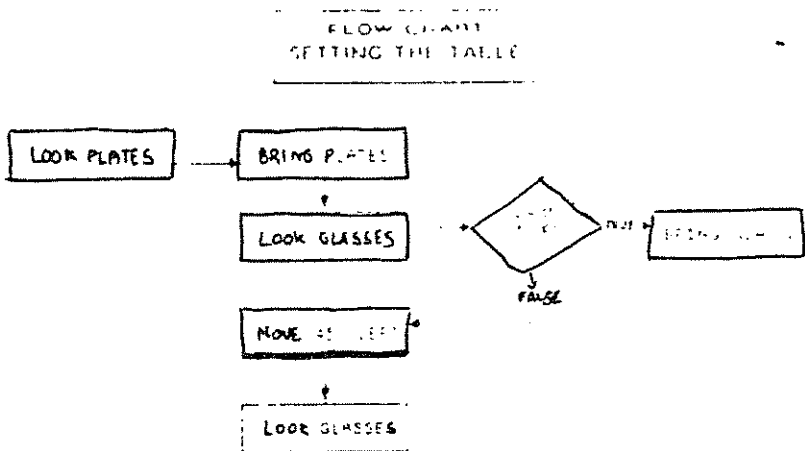
After the child carried out Instruction 3, we asked the child to change direction, and look again. This was because the child was unsuccessful in finding the glasses. What we were saying was, "[IF you] LOOK [and find the] GLASSES, [Then] BRING [the] GLASSES [to the table]. In this case, the child did not find the glasses, so we provided an alternative. This alternative was to move 45° Left, and look again. We can set up a general rule for this kind of Conditional Testing. That rule looks like this:

IF [something is true, or meets some condition you have set,] THEN DO [this thing]. IF [it is] FALSE, THEN DO [something different]

Programmers, who often hate to use more words than they absolutely have to, have a much shorter way of expressing the same rule:

IF...(x)...THEN DO...(y)...ELSE DO...(z)

The Flow Chart for this testing of conditions, looks like this:



The Diamond shaped figure which you see is the **Decision Box**. It represents the test determining whether the condition asked for is true or false. It then determines what the next instruction set should be. In this case, since the test to **FIND GLASSES** is false, the next instruction is to **MOVE 45° LEFT**, followed by another instruction to look for the glasses. If the answer had been true, the next instruction would be to **bring the glasses**.

Section 1: Using ChipWits

Most of the chips which have an active Operator/Argument combination provide for this testing of whether conditions are being met. You will notice the **COIN FLIP** operator. This allows you to branch true or false randomly.

Building a CHIPWIT

You are now ready to begin the construction and operation of your first CHIPWIT! The following instructions take you all the way through startup to the completion of the first mission. Startup

- > 1. Turn On the Macintosh.
- > 2. Insert the CHIPWITS DISKETTE in either the internal or external drive.
- > 3. Wait approximately 20 seconds as the BRAINWORKS screen comes up and the CHIPWITS system loads.
- > 4. Pull down the **Warehouse** menu.
- > 5. Select the first number on the menu as the CHIPWIT you will be building.

You are now ready to enter the Workshop

Using the Workshop

1. Pull down the Workshop menu.
2. Select **Enter**. The Workshop work area will appear on the screen. In the upper left chip socket is an icon which looks like a traffic light. In the lower right hand socket is a blinking (cursor) chip. The first chip you program must either be in the socket **immediately to the right** or the one **immediately below** the traffic light icon.
- > 3. Click on the socket position where you wish to place the chip. You will see the blinking outline of the chip in the socket.

Section I: Using ChipWits

4. Select the Operator you want, and click on it.
5. If the Operator requires an Argument, the Argument menu will appear. Select the Argument you want.
6. The **True/False** tabs will appear on the chip. If you wish to change the position of either one, click on the tab, then click on the side of the chip where you want the tab to point from.

Modifying Existing Chips

If you decide you wish to change an existing chip, simply click on the chip, and repeat steps 4, 5, and 6 above, making the changes you want.

Deleting a Chip

To delete or remove a chip completely, click on the chip, then click on the wastebasket. The chip operators and arguments will disappear. You can then click on a new socket to construct a chip in another place, or you can go through the chip building process using the same socket as before.

To move a chip, click and drag the chip to the desired socket. Release mouse button when chip is in desired location.

Repeating the Program

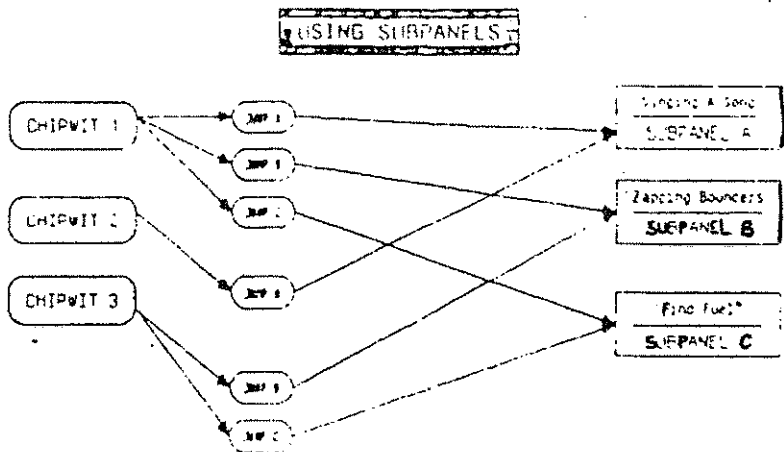
As you program your chips, you will have **one** or more final chips in your Main Panel routines. Final chips are the chips at the end of a sequence. Make certain you provide for a repeat of the main routine by ending every program branch on the Main Panel with a **LOOP CHIP**. This returns program execution to the traffic light chip at the beginning of this panel.

SubPanel Programming

When you have a complicated routine which you wish to use many times through a CHIPWIT program, the SubPanel is the place to program this routine. You may choose to use the same routine with more than one CHIPWIT. Suppose, for example, you want a routine in which the CHIPWIT is going to sing a song you have composed

Section I: Using ChipWits

You develop a SubPanel for the song. You can then transfer this SubPanel to the main program of any of your CHIPWITS. The figure below diagrams how this might be done.



In this example, CHIPWIT 1 needs to be able to use all three SubPanels. In the Main Panel (the primary program), a **JUMP** chip is used to "call" each SubPanel when needed. The Argument is A, B, or C. CHIPWIT 2 only needs the use of the "Singing a Song" SubPanel, and so only the **JUMP A** chip is required. Thus, the same SubPanels can be used by many CHIPWITS, or by the same CHIPWIT many times. Yet you will need to program that SubPanel only once. You must use the **BOOMERANG** as the last chip in a SubPanel program. When you use this chip, the program will return to the Main Panel and begin with the chip which comes immediately after the **JUMP** chip which took the program to the SubPanel. To select a SubPanel to program, simply click on the letter which identifies that panel.

Clearing a Panel

To clear any Main, or SubPanel of all chips, select the panel, then pull down the **Workshop Menu**. Select **Clear Panel**. The entire panel will be cleared. Be careful of this since you could lose some valuable programming time should you clear a panel you really meant to keep.

Section I: Using ChipWits

Copying, Cutting, and Pasting Panels

To remove a Panel and move it to another CHIPWIT

- > 1. Select the panel you want to move. This may be either a SubPanel, or the Main Panel.
- > 2. Pull down the **Workshop Menu**.
- > 3. Select **CUT Panel**.
- > 4. Pull down the **Warehouse Menu**.
- > 5. Select the CHIPWIT to which you want to transfer the panel.
- > 6. Pull down the **Workshop Menu**.
- > 7. Select **ENTER Workshop**.
- > 8. Select the panel you want to paste into. This may be either a SubPanel, or the Main Panel.
- > 9. Pull down the **Workshop Menu**.
- > 10. Select **PASTE Panel**

To copy a panel and move it to another CHIPWIT

- > 1. Select the panel you want to copy. This may be either a sub panel, or the main panel.
- > 2. Pull down the **Workshop Menu**
- > 3. Select **COPY Panel**.
- > 4. Pull down the **Warehouse Menu**
- > 5. Select the CHIPWIT to which you want to transfer the panel
- > 6. Pull down the **Workshop Menu**
- > 7. Select **ENTER Workshop**

Section I: Using ChipWits

Copying, Cutting, and Pasting Panels

To remove a Panel and move it to another CHIPWIT

- > 1. Select the panel you want to move. This may be either a SubPanel, or the Main Panel.
- > 2. Pull down the **Workshop Menu**.
- > 3. Select **CUT Panel**.
- > 4. Pull down the **Warehouse Menu**.
- > 5. Select the CHIPWIT to which you want to transfer the panel.
- > 6. Pull down the **Workshop Menu**.
- > 7. Select **ENTER Workshop**.
- > 8. Select the panel you want to paste it to. This may be either a SubPanel, or the Main Panel.
- > 9. Pull down the **Workshop Menu**.
- > 10. Select **PASTE Panel**.

To copy a panel and move it to another CHIPWIT

- > 1. Select the panel you want to copy. This may be either a SubPanel, or the main panel.
- > 2. Pull down the **Workshop Menu**.
- > 3. Select **COPY Panel**.
- > 4. Pull down the **Warehouse Menu**.
- > 5. Select the CHIPWIT to which you want to transfer the panel.
- > 6. Pull down the **Workshop Menu**.
- > 7. Select **ENTER Workshop**.

Section I: Using ChipWits

- > 8. Select the panel you want to paste into. This may be either a SubPanel, or the Main Panel.
- > 9. Pull down the **Workshop Menu**.
- > 10. Select **PASTE Panel**.

Save a CHIPWIT

After you have completed the programming of your CHIPWIT you can either:

- a. Try it in a mission, and modify it to your liking;
- b. **SAVE** the CHIPWIT and use it in a mission, or
- c. **SAVE** the CHIPWIT and end the session with your program on file. Since you will be going through a good deal of work you will certainly want to save the CHIPWIT before ending your time with the program. Here's how:
 - > 1. Pull down the **Workshop menu**
 - > 2. Select **Save ChipWit**. This will cause a dialog box to appear on the screen.
 - > 3. Drag across the name window until you highlight the current name. Keyboard the name you want for your CHIPWIT.
 - > 4. If you wish to assign your CHIPWIT as appropriate to a particular Environment, click in the box next to the Environment's name. (When that Environment is selected, the CHIPWIT name will be in outline when shown in the Warehouse Menu)
 - > 5. You may complete the **SAVE** by pressing either **[RETURN]** or **[ENTER]** on the keyboard after typing the name, or by clicking in the **OK box**. Either action will clear the dialog box from the screen.

Section II: Advanced Programming

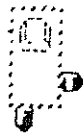
OBJECTIVES

In section I you learned how to operate your CHIPWITS system, and you learned some elements of beginning programming with the workshop. After completing this section you will be able to:

- Use the Keypress for testing program routines within an environment.
- Use Stacks and Registers to improve the efficiency of your CHIPWIT programs.
- Make judgments about tradeoffs between cycles, fuel consumption, damage risk, and scores to increase the effectiveness of your CHIPWIT.
- Increase the speed and efficiency of your CHIPWIT and your programming.

THE KEYPRESS

The Keypress Operator is represented by an icon of a computer key with no letter on it. (See Figure 2.1) By selecting that icon and then selecting a letter from the Arguments Menu, you install a manually controlled command in your CHIPWIT. This command runs whatever routine you link to the Keypress chip. The Keypress Operator gives your CHIPWIT the benefit of your human intelligence.



You can have as different Keypress commands: one for each letter of the alphabet.

Uses

As you watch your CHIPWIT in a Mission, you might notice it is unable to accomplish a specific task because it cannot tell when it is

Section II: Advanced Programming

appropriate to run the routine necessary to accomplish that task. Program the routine and link it to your program with a Keypress for the letter A. As you watch your CHIPWIT approach a situation in question, activate the Keypress routine by typing the letter A just *before the program passes over the Keypress chip*. This will divert the Flow of Control into the routine you have programmed to accomplish the task.

The Keypress can be used as a tool to help you begin programming CHIPWITS. Once you are comfortable getting your CHIPWIT to accomplish tasks (that is, giving it the right instructions in the right order), you will want to begin programming it to sense *when* it is appropriate to perform them. By removing the Keypress commands and substituting more complicated sensing routines, you will remove your guiding hand and let the CHIPWIT think for itself!

Each routine that you controlled with the Keypress can easily be constructed in a SubPanel. Use the Main Panel to determine when that routine is appropriate and substitute for the Keypress a jump chip to that SubPanel.

SubPanels are particularly useful for storing those routines which will be repeatedly required within an Environment.

You now can practice making the CHIPWIT perform tasks, and you can add manual controls in case your CHIPWIT is not sure when to perform them. The next section offers tips for *building a better* CHIPWIT. To do this, you need to know more about what makes your CHIPWIT run.

CYCLES

The CHIPWIT's Life Span

As we all know, every living thing has a life span — the period of time that it stays alive and active. Each CHIPWIT also has a life span, and that span is measured in **Cycles**. The number of Cycles given a CHIPWIT at the beginning of a Mission is determined by the Environment in which the Mission takes place. Cycles available range from 6,000 in **Greedville** to 39,000 in **Octopus Garden**.

Section II: Advanced Programming

Cycles cannot be regained once they are spent, and any action of the CHIPWIT will use up Cycles to some degree. Cycles are also consumed by hv commands on the Main and SubPanels (like Boomerangs and Loops). The second column of the chart below lists the cost in Cycles of each CHIPWIT activity.

Activity	Cycles	Fuel
Loop	1	1
SubPanel	1	1
Boomerang	1	1
Junction	0	0
Move	3	5
Pickup	1	5
Zap	1	7
Sing	2	2
Feel	4	4
Look	4	2
Smell	4	2
Flip Coin	3	1
Compare Key	3	1
Number =	2	1
Number ·	2	1
Object =	2	1
Move =	2	1

Section II: Advanced Programming

Save Number	1	1
Save Object	1	1
Save Move	1	1
Push/Pop Stack	1	1
Plus (Increment)	1	1
Minus (Decrement)	1	1

On-Screen Cycle Count

During a Mission, the remaining Cycle count is shown in the upper right hand corner of the Environment screen. When this count reaches zero, the Mission will come to an end regardless of the number of points accumulated or the CHIPWIT's fuel supply.

Cycles and Scoring

As a general rule, the more you conserve Cycles, the more functions your CHIPWIT can perform, and the more points it will accrue. Remember, point totals are increased by the amount of time your CHIPWIT has in its Environment. Conserving Cycles is discussed in the section titled *Conservation of CHIPWIT Resources*.

FUEL

CHIPWITs need fuel to operate. Like Cycles, fuel is automatically consumed as the CHIPWIT explores an Environment. The third column of the chart above shows the fuel requirements for each of the CHIPWIT's actions. Unlike Cycles, however, fuel can be regained in the Environment. Coffee and Pie are the Things which restore the CHIPWIT's fuel level when acquired (eaten). Conserving fuel is discussed in the section titled *Conservation of CHIPWIT Resources*.

During a Mission, the CHIPWIT's fuel supply is illustrated by a pair of icons in the Status Panel. (The Status Panel is to the right of the

Section II: Advanced Programming

Environment panel. The fuel icons are the second set in the top row of icons in the Status Panel.) The fuel meter icon on top labels the pair. Below it is a beaker which fills up and drains out indicating the CHIPWIT's fuel level.

DAMAGE

Like fuel and Cycles, Damage is a factor which must be considered when programming CHIPWITs. A CHIPWIT can suffer damage in many ways. In the more benign Environments, the only way for a CHIPWIT to sustain damage is by running into the walls that define the rooms. The more difficult Environments pose multiple damage risks including bombs, electro-crabs, and bouncers.

Damage, like Cycle consumption, is irreversible and will terminate the Mission automatically once it reaches a critical level. Avoiding damage is discussed in the section titled *Conservation of CHIPWIT Resources*.

During a Mission, the Damage your CHIPWIT has sustained is illustrated by a pair of icons in the Status Panel. (The damage icons are the first set in the top row of icons in the Status Panel.) The label icon on the top is an electrical bolt (a damaging Zap from an electro-crab!). Below it is a beaker which fills up as the CHIPWIT sustains damage. When the beaker is full, the Mission is over.

CONSERVATION OF CHIPWIT RESOURCES

The three resources the CHIPWIT possesses are Cycles, Fuel, and itself. The resources have different characteristics. Cycles are predetermined, automatically expended, and non-regenerating. Fuel is automatically expended as well but is acquired fairly easily; and damage can be avoided completely but is irreversible. Each Resource deficiency terminates the Mission when critical levels are reached. Therefore, the conservation of all three resources influences the ultimate goal of a CHIPWIT: to gain points. Memory Stacks provide one method to husband Resources.

Memory Stacks

Memory Stacks are storage areas in which you can put lists (up to 256 items long) of:

1. Moves your CHIPWIT makes.
2. Things your CHIPWIT encounters
3. Numbers.

Your chip programming controls what is pushed into each of the three memory stacks and what is popped (taken) out of them. Items are stored in Memory Stacks in the order they are entered. They can only be popped out in the reverse of this order.

Use of Icons

Figures 2.2, 2.3, and 2.4 are the icons for pushing data into the Thing, movement, and number stacks respectively. Figures 2.5, 2.6, 2.7, and 2.8 are the icons for comparing the top item in a Stack to the criteria you have set (the Argument of the chip calling for the comparison).

Look at Figures 2.2 and 2.5 (pushing and comparing data using the Thing stack). The Operator in Figure 2.2 will reveal only Arguments which are Things. You probably will not be using the comparison capability of the stack (Figure 2.5) until pushing to the stack is completed (you might put 7 different Things into the stack in a room). When you select the comparison Operator (Figure 2.5), it will cause only Things to appear in the Arguments window.



Figure 2.2



Figure 2.3



Figure 2.4



Figure 2.5

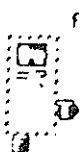


Figure 2.6



Figure 2.7

Figure 2.8



Section II: Advanced Programming

Figures 2.3 and 2.6 work together in the move Stack exactly as the Things pair does. Both Operators require Arguments, and both Operators will cause only Arguments pertaining to the stack (direction of motion) to appear.

Figures 2.4, 2.7, and 2.8 are used with the number Stack. In the other two stacks, the comparison of data with criteria was simple. Either the items in the stack were equivalent to the Argument in the chip that is calling for comparison or they were not. When comparing numbers, however, you have the additional capability of determining whether the value in the Stack is less than the Argument in the chip that is calling for comparison. This is why the number Stack can utilize two Operators. One (Figure 2.7) will determine if the value in the Stack is equal to the Argument in the chip calling for comparison and the other (Figure 2.8) will determine if the value in the stack is greater than the Argument. The icon indicating the value of the Argument is a beaker graded into eighths (this allows comparison with the numbers 0-7).

The Arguments for the number stack include three icons to instruct the CHIPWIT where to get the value for comparison. The electric bolt calls the damage level value, the fuel gauge calls the fuel meter value, while the Range Finder calls the current range value (the distance registered on the last look instruction).

Using the Stacks

Once you have pushed an item into a Stack, you can use the Stack to compare the numeric value of the item with a value you define (either less than X or equal to X). You can also compare it to Things and moves. Is the Thing or move the same as the one you are testing for? These three types of comparisons are done in the three different stacks (the number, move and Thing stacks). Once you have compared the item pushed into the stack with the condition you have set up, you can branch through the True or the False Tab depending on the results of the comparison. Let's see how this might work in a Mission.

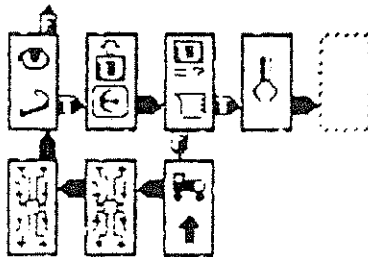
The Number Stack

Goal. Acquire a Piece of Pie Using Number Stack

Section II: Advanced Programming

1. Have CHIPWIT look for Pie.
2. If CHIPWIT sees Pie, use range finder to measure distance between CHIPWIT and Pie. Push value of distance into Number Stack where it will be represented by the range finder icon.
3. Compare the number generated by the range finder against the zero to determine whether the CHIPWIT should reach for the Pie or move towards it.
4. If the value generated by the range finder is equal to zero, instruct the CHIPWIT to reach for the Pie.
5. If the value generated by the range finder is greater than zero, instruct the CHIPWIT to move one space forward and repeat the sequence from the beginning.

Figure 2.9 shows how these steps appear when programmed.



Using this routine, you have conserved CHIPWIT's fuel. Without using the Number stack to compare the value the range finder determined with zero, CHIPWIT would have had to reach for the Pie if he saw it. If the pie was three squares away, CHIPWIT would have to reach out, get no pie, and move forward three times before getting the Pie. Reaching uses fuel!

Using the routine above, the maximum fuel consumption would be 15 units and the maximum Cycles used would be 16. If you had to program without use of Stacks to compare values, the maximum fuel consumption is 40 units and Cycle consumption is 40 units.

The Move Stack

The Move Stack allows you to record up to 256 CHIPWIT moves. This stack is useful for bringing your CHIPWIT back from extended

Section II: Advanced Programming

travels away from the center of Environments. You will need to let your CHIPWIT explore the Environment while you take note of a return path. As you fine tune the CHIPWIT for its Environment have the return moves pushed into the move Stack as the CHIPWIT heads to the edges of the Environment. When the CHIPWIT needs to return through the maze, begin popping the moves out of the Stack. (You might want a Keypress to control this.) They will come out in reverse order — just the way the CHIPWIT needs them to return!

Figure 2.10 illustrates the icon for the Operator to pop items out of stacks. The Arguments for this Operator are icons representing the three Stacks.



Figure 2.10

The Thing Stack

The Thing Stack is useful when you are building general purpose search routines. As the CHIPWIT enters a new room have it smell for the good Things in that room. Push the Things detected into the Thing stack. Now you can pop the first one out and conduct a search of that room for the Thing. After CHIPWIT has found the Thing, pop the next Thing out of the memory stack and reuse the search routine, this time with the new Argument.

Not only can the Thing Stack be used in conjunction with the smell Operator to detect good things, but it can be used to detect and store bad Things as well. Upon entering a room, the CHIPWIT might smell for the evils of the Environment. Have him smell for the most dangerous evils last (the bombs). Each evil detected could be pushed into the Thing Memory Stack. As the Things are popped out of the stack, the most dangerous one will be the first to come out.

Registers

Registers display the top three items in each of the three Stacks. They are displayed during Missions above the Debug Panel and to

Section II: Advanced Programming

the right of the Status Panel. If nothing has been pushed into a Stack, the register for that stack will be empty.

Avoiding Damage

A balance has to be struck between efficiency and precaution when planning strategies to avoid excess damage. Touch and look instructions which keep the CHIPWIT away from walls are Cycle and fuel intensive if used thoroughly.

The smell instruction will help your CHIPWIT determine the level of danger which exists within a room. This information, as mentioned above, can be stored in the Thing stacks for later use.

THE EIGHT ENVIRONMENTS

Each of the eight Environments has its own challenges for the CHIPWIT.

Greedville. This is a very friendly Environment. You need only develop routines for finding and consuming food. Remember that disks have the highest point value. Walls are your only danger here.

ChipWit Caves. Routines are necessary here to locate Electro-Crabs. Use the range finder to determine how far away they are.

Doom Rooms. Try to destroy the bad things using your Zapper. Be aware that since there are no good things in the rooms, your fuel will need to be carefully conserved.

Peace Path. Construct search routines to distinguish the good from the bad. You may find it necessary to kill a few bad Things.

Mystery Matrix. The key to deciphering the signposts to the Mystery Matrix rests with the CHIPWIT's ability to discover patterns in them.

Memory Lane. Try to use the move Stack to remember optimal pathways to follow as your CHIPWIT goes back and forth in the maze.

Section II: Advanced Programming

Octopus Garden: Keep the right hand wall in sight as a guide to moving in and out the legs of the garden. The move Stack can be useful here

Boom Town: Travel between bombs. There is no point in conserving fuel or cycles here as this is one of the most dangerous of Environments. Your most important task is to survive, not to husband resources

Glossary

Argument-The object of an Operator.

Boomerang-The return instruction from a SubPanel to the Main Panel.

Chip-The container of an individual Instruction Set

ChipWits-The robots programmed

Coin Flip-An Operator providing for a random choice between a True or a False branch

Compare Key-Operator which checks whether a specific key has been pressed by the person using the program and branches True or False accordingly.

Compare (Thing, Move, Number)-An Operator which checks a Stack to determine if a condition (Things, Moves, Numbers) is present.

Conditional Testing-A test for a condition being present leading to True or False branching depending on the answer.

Debug Panel-An on-screen display providing information allowing the user to trace program execution in three modes (step-through, slow pace, and full speed).

Decrement-An Operator used to decrease a value in the number Stack by one

Default-A preset condition changeable by the user

Environments-The rooms in which the ChipWit carries out its mission

Flow Charts-A graphic format used by programmers to plan their programs.

Flow of Control-The sequence of Instructions within a program.

IBOL (Icon Based Operating Language)-The set of commands which tells the ChipWit how to behave.

Icon-A stylized picture which represents an instruction, action, Thing, or condition.

Increment-An instruction to increase a value in the number Stack by one.

Instruction Set-The combination of an Operator and an Argument

Junction-A chip whose sole purpose is to connect two other chips

Loop-The return-to-beginning instruction in a Panel

Main Panel-The Panel on which the primary flow of Control is programmed

Menu Bar-The strip across the top of the screen showing the selections available to the user.

Mission Assignments-The objectives to be achieved within an environment by the ChipWit.

Operator-The action command telling the ChipWit what it is to do

Options-A utility menu found on the Menu Bar.

Paste Panel-An editing command allowing the user to transfer a previously Cut or Copied Panel

Pop-Removes an item from the top of a Stack

Registers-A storage location for Things, moves, and numbers which have been pushed into the stacks.

Save (Move, Number)-An Operator which places a move, Thing, or number into a Stack (push to stack)

Stacks-Memory storage provided for up to 256 Things, numbers, or moves.

Stats Panel-An on screen display showing the score, number of missions, the average score, and the highest score

SubPanel-A Panel which is called from the Main Panel and which can be repeatedly used in the program.

Tabs-(True and False) Directs Flow of Control based upon the results of a Conditional Test.

Things-Arguments which are the objects of the Operators.

Warehouse-The menu which allows you to load ChipWits into an Environment.

Wastebasket-A disposal instruction which allows you to delete unwanted chips

Workshop-The place where you build your ChipWits

Zap-The command for destroying Things