

Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website <https://blunham.com/Radar>, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: <https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin>

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (<https://blunham.com/Radar>). Please do not point them at the file itself as it may move or the site may be updated.

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

In order to generate this file, I need to scan the pages, split the double pages and remove any edge marks such as punch holes, clean up the pages, set the relevant pages to be all the same size and alignment. I then run Omnipage (OCR) to generate the searchable text and then generate the pdf file.

Hopefully after all that, I end up with a presentable file. If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you personally – I know that I would have liked to have found some of these files years ago – they would have saved me a lot of time !

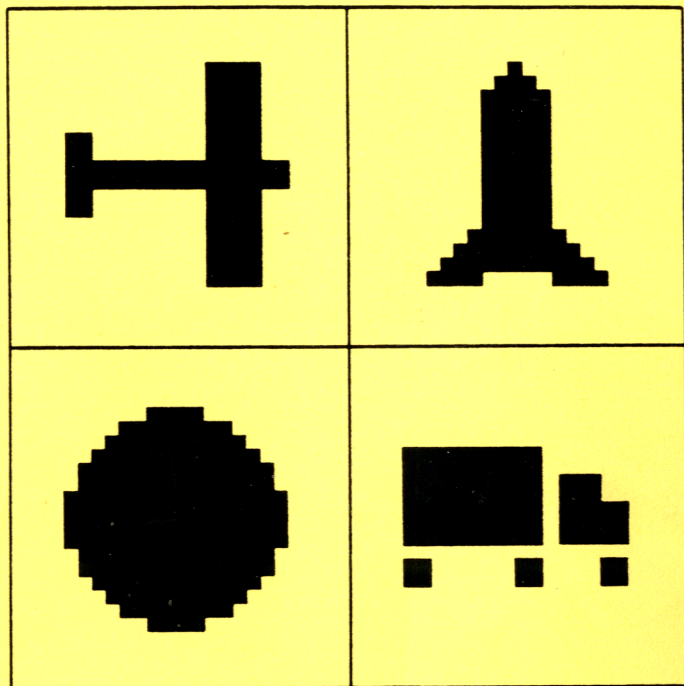
Colin Hinson

In the village of Blunham, Bedfordshire.



TI LOGO CURRICULUM GUIDE

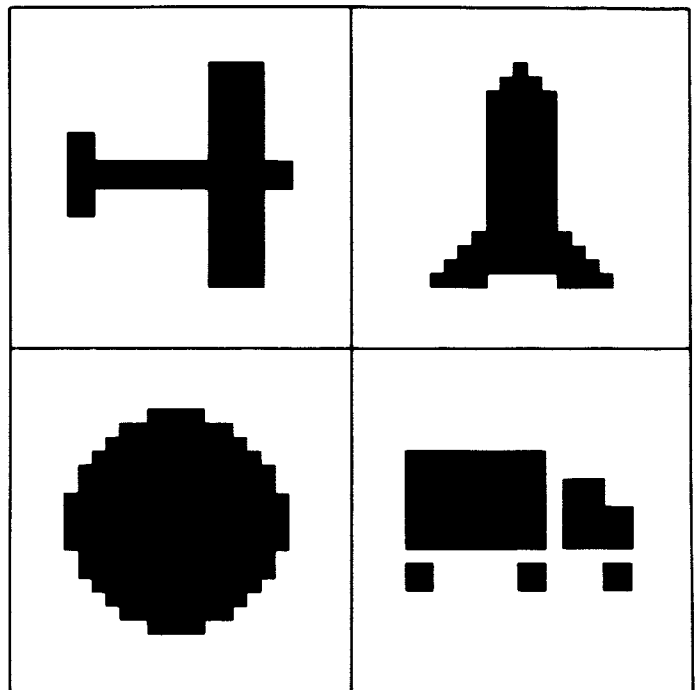
TI LOGO is a computer language designed to encourage a student's creativity by exploring the areas of mathematics, logic, and communication skills. This *Curriculum Guide* is written for you, the classroom teacher and problem solver. Use it to help you teach TI LOGO to students ranging in age 3 through 12. The *TI LOGO Reference Manual*, included in this package, provides a complete explanation of all the commands, operations, and variables that are part of the language.





TI LOGO CURRICULUM GUIDE

TI LOGO is a computer language designed to encourage a student's creativity by exploring the areas of mathematics, logic, and communication skills. This *Curriculum Guide* is written for you, the classroom teacher and problem solver. Use it to help you teach TI LOGO to students ranging in age 3 through 12. The *TI LOGO Reference Manual*, included in this package, provides a complete explanation of all the commands, operations, and variables that are part of the language.



This book was developed and written by:

Diane R. Musha, staff member of the Texas Instruments Learning Center

Coleta L. Lewis, staff member of The Lamplighter School, Inc.

Theresa Overall, staff member of The Lamplighter School, Inc.

With contributions by:

Sonia Duke

Marjie Price

Jacquelyn F. Quiram

Jan E. Stevens

and other staff members of the Texas Instruments Learning Center

The TI LOGO language was developed in cooperation with the Artificial Intelligence Laboratory at Massachusetts Institute of Technology. Appreciation is expressed to Professor Seymour Papert, Gary Drescher, Margaret Minsky, and Edward Hardebeck for their activities in its development.

Design and artwork by:

Don Kubecka

Diane Musha

KNR & Co.

Typesetting by:

Judy Lipsett

TABLE OF CONTENTS

YOUR STUDENTS AND THE COMPUTER

INTRODUCTION TO TI LOGO	1
Piaget's Influence	1
"Turtle Graphics"	2
The Classroom Computer	2
Teaching the Computer	3
LAMPLIGHTER SCHOOL, INC.	5
Lamplighter Computer Project	6
USING THIS MANUAL	7
DEVELOPING A CLASSROOM STRATEGY	9
USING THE TI LOGO ACTIVITIES	11
Loading the Activities	14
Loading Cassettes	15
CUE CARDS	17
Designing a Cue Card	17
Choices for Turtle Activities	17
Choices for Sprite Activities	17
Introducing an Activity with a Cue Card	18
PREOPERATIONAL LEVEL OF UNDERSTANDING	19
Activities for Preoperational Level Students	19
LINE Activity	20
PARK Activity	21
PEOPLE Activity	22
DRAW Activity	23
DALLAS Activity	24
BUILD Activity	25
MOVE and HALT Activities	26
ROAD Activity	27
EARLY CONCRETE LEVEL OF UNDERSTANDING	29
Adapting Cue Cards	29
Activities for Early Concrete Level Students	30
AIM Activity	31
PAINT Activity	33
GRID __ Activity	34
LISTEN Activity	36
MOVE __ and HALT Activities	37
TARGET Activity	38
Adapting Cue Cards	39
Concept: Giving the Computer Commands	40
Concept: Primitives with the Turtle	41
Concept: Talking to a Sprite	42
Concept: CLEARSCREEN	43
Concept: State of the Pen	44
Concept: Changing the Color of the Turtle's Ink	45
Concept: REPEAT	46
Concept: Talking to More Than One Sprite	47
Concept: Talking to :ALL	49
Concept: Dots and Quotes	50
Concept: SETHEADING and SETSPEED	51

Concept: No More Than Four Sprites in a Row	53
Concept: Which Four are Visible?	54
Concept: Even Pieces of a Sprite Count	55
Concept: MAKESHAPE and CALL	57
CONCRETE LEVEL OF UNDERSTANDING	59
Concept: Format for Short Forms of Commands	60
Concept: COLORBACKGROUND	61
Concept: FREEZE and THAW	62
Concept: HIDE TURTLE and SHOW TURTLE	63
Concept: State of the Sprite	64
Concept: Teaching the Computer	66
Concept: Making Big Shapes	69
Concept: Teams	70
Concept: Debugging Skills	72
Concept: Subprocedures	73
Concept: WAIT	74
Concept: BEEP and NOBEEP	75
Concept: Making Movies	76
Concept: Recursion	79
Concept: EACH	81
Concept: YOURNUMBER	82
Concept: EACH and YOURNUMBER Together	83
Concept: Multicolored Designs with Sprites and Shapes	85
Concept: Variables in a Procedure	87
Concept: More with Variables	89
Concept: PRINTCHAR, CHARNUM, and MAKECHAR	91
Concept: PUTTILE	94
Concept: Giving One Color to Characters	95
Concept: Giving Two Colors to Characters	97
Concept: Words and Lists	99
Concept: PRINT and TYPE	100
Concept: READLINE	102
Concept: Conditionals	103
Concept: READCHAR	106
Concept: More with SHAPE, COLOR, SPEED, and HEADING	108
Concept: X- and Y-Coordinates	109
Concept: Modulo	111
Concept: RANDOM	113
Concept: PENREVERSE	114
Concept: FIRST, LAST, BUTFIRST, and BUTLAST	115
Concept: Commands, Operations, and Variables	117
Concept: Output	119
Concept: Global and Local Variables	120
Concept: Velocity	121
Concept: GO and Labels	123
TI LOGO SIMULATION GAMES	125
Games the Turtle Plays	126
Introducing the Turtle Mode	126
The Human Turtle	127
Debugging with the Turtle	128
The Paper Turtle	129

Games Sprites Play	130
Introducing the Sprite Mode	130
Introducing More Than One Sprite	132
Introducing SETHEADING	134
Introducing EACH	136
Introducing YOURNUMBER	138
EACH and YOURNUMBER Together	139
Priority Sprites	140
Four Sprites All in a Row	141
Introducing Velocity	142
Games the Turtle and Sprites Play	143
Teaching the Computer	143
Dots and Quotes	144
Introducing Variables	145
Recursive Procedures	147
CLASSROOM HINTS	149
Student Journal Writing	149
Teacher Journal Writing	150
Computer Vocabulary and Its Importance	151
Computer Etiquette	153
Turning the Computer On and Off	154
Placement of the Computer	155
BIBLIOGRAPHY	156
APPENDIX A—CUE CARD SAMPLES	157
APPENDIX B—SHAPE AND COLOR CHART	182
APPENDIX C—STANDARD CHARACTER CODES	183
APPENDIX D—QUICK REFERENCE GUIDE	184
IN CASE OF DIFFICULTY	185
WARRANTY	186
TI LOGO REFERENCE GUIDE	

TI LOGO CURRICULUM GUIDE

YOUR STUDENTS AND THE COMPUTER

The Texas Instruments Home Computer is a rugged, durable device designed for easy use and care. Teach your students to give the computer the same good care and respect he or she would give a television set, record player, radio, or any other piece of electronic equipment:

1. Keep all food away from the console.
2. Don't hammer on the keyboard or place heavy objects on it.
3. Don't touch the module contacts. These are recessed in the module to help prevent accidental soiling and/or damage.

The letters and numbers of the keyboard are arranged in the same order found on standard typewriter keyboards. To assist the young student or the student unfamiliar with the keyboard, small squares of different colored tape may be placed to the left of each row of keys. This helps a student focus on the correct row and then locate the correct key.

When introducing your students to the computer, point out the row of numbers at the top and the rows of letter keys below. Demonstrate the special function keys, according to the TI Home Computer console you have. If you want your students to be responsible for inserting the module, show them how to insert it and how to select TI LOGO.

For more information concerning the computer in your classroom, see the "Classroom Hints" section.



INTRODUCTION TO TI LOGO

TI LOGO is an innovative approach to computer languages which not only develops computer awareness but also enriches a child's mathematical, logical, and communication skills. The language and the microcomputer system on which it is run can be incorporated into the curriculum as a valuable learning tool, to enhance individualized programs for each student. The classroom computer reinforces what the teacher is teaching; it does not displace current teaching methods.

TI LOGO is a derivation of LOGO, a computer language based on a philosophy of education developed over a 12-year period by Professor Seymour Papert and the staff of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Many of LOGO's premises are based on Jean Piaget's theory of intellectual development, which describes a child's development as taking place in a series of stages.

Piaget's Influence

Piaget viewed mental development as a process of adaptation to the environment and an extension of biological development. He grouped the stages of development into four categories:

- sensori-motor, when a child's behavior seems to be motivated by stimulated reflexes;
- preoperational, when behavior begins to be internalized and characterized by early stages of reflective thinking;
- concrete, when a child's behavior combines old and new experiences in a more systematic approach to situations; and
- formal, when behavior reflects the ability to use symbols and distinguish between the actual and the possible.

During these stages, cognitive acts are the result of the organization of and adaptation to a child's environment as he or she perceives it. At each stage, the learner's understanding of the world incorporates new experiences into previous understandings, resulting in cognitive change.

Piaget believed that every child progresses through these stages to reach the formal level of development. The rate of cognitive development, of course, differs for each child. According to Piaget, the important factor is that a child must achieve a certain amount of experience at each level of understanding before he or she is capable of advancing to a higher developmental stage.

TI LOGO offers a simulation of a real-world environment which a child can explore safely as he or she progresses through these developmental stages. The environment is rich in potential experiences, and TI LOGO encourages learning by discovery, the natural way that a child finds out about the world. Problem solving, logical sequencing, decision making, self-direction — these are skills developed by exploring the TI LOGO environment, as well as skills required in the classroom and in life.

Learning appears to be more successful when a person is able to perceive that the knowledge serves a personal need. For example, one of the most dramatic images of successful learning is the way children learn to talk. Because it is important and vital for a child to communicate with someone that can be of assistance, learning to talk engrosses the child from the first years of life.

TI LOGO CURRICULUM GUIDE

In contrast, learning to write is usually done to gain approval of adults. For this reason, children often do not find the development of writing skills to be a real or personal need. As a result, a child may not develop these skills to the level of his or her potential. With the computer, children can discover that the alphabetic language actually serves a real and personal need. They must use correct language structure and spelling in order to get the computer to do what they want.

TI LOGO lets the student, at all levels of understanding and experience, communicate with the computer using an easy-to-understand language.

“Turtle Graphics”

Turtle geometry, in the Turtle mode of TI LOGO, allows children to solve drawing problems by using a very powerful heuristic principle: internalizing problem solving by playing the role of the Turtle, walking through what you want the Turtle to do, and describing what you did in Turtle language. When concerned with the *state of the Turtle*, children can practice many powerful concepts. They learn to think of formal mathematics as it relates to body-mathematics, thus using mathematics as a language. In this way, children can think of mathematics as an instrument to be used for personal ends. See the “Concept: Primitives with the Turtle” section for an introduction to the Turtle mode.

The Classroom Computer

The computer is not intended to replace the teacher or traditional teaching methods in the classroom. The computer is a tool to assist, support, and facilitate classroom learning. You, the teacher, select the teaching style that best suits the learning style of your students. Then you observe and use this insight in integrating TI LOGO into the students’ classroom activities.

For best results, it is important that you, like your students, become a self-directed computer learner. During the process of thinking about the computer and “debugging” a TI LOGO procedure, the communication interaction between you and a student can be very articulate and effective. This communication presents an opportunity for the “helper” to become a fellow learner. Although the work at the computer may be private, it tends to increase the students’ desire for interaction. You ask and answer questions, provide help if asked, and sometimes provide models for the students. The flow of ideas between you and a student, and between one student and another, fosters a richer and deeper interaction between the participants.

TI LOGO is not intended to raise a student’s IQ or increase performance on standardized tests. The computer encourages the student to explore and extend existing concepts, build on current intellectual structures, and give concrete form to areas of knowledge that were previously abstract, theoretical, or unobservable. And the TI LOGO computer language fosters “computer literacy” while enriching the student’s logical thinking skills and developing a healthy, positive attitude toward computers.

There is no ceiling to learning with TI LOGO. The computer allows the student to pace his or her own personal level of achievement. All of this can contribute to the improvement of the student’s concept of self-worth.

The Lamplighter School, Inc. in Dallas, Texas, is a school where the teachers and students work together with the computers to create an individualized learning environment for each student. (See the “Lamplighter School, Inc.” section for further information.)



Teaching the Computer

To accomplish a specific task, a student must “teach” the computer what to do and how to do it. The individual determines the level of challenge he or she wants to explore in the areas of problem solving and communication skills. The student controls the computer, which in turn provides immediate feedback.

“Debugging” is a term that implies something is not working the way the computer user wishes. The user employs a trial-and-error method to discover why the computer is not performing as desired. There are no mistakes in this atmosphere of learning. There are “bugs.” Bugs occur and a learner uses the discovery method to debug.

TI LOGO can be integrated into any curriculum area. It is capable of reinforcing the classroom curriculum for young children as well as high school students. The TI LOGO activities included in this package are designed for students at the preoperational and early concrete levels of understanding (approximately ages 3 through 8). The activities provide experiences and computer readiness that eventually allow a student to teach the computer to accomplish the things he or she desires. All of the activities and procedures in this Guide and the procedures in the *TI LOGO User's Manual* are currently being used in the classroom.

You and your students should be comfortable with TI LOGO and the computer. Everyone should understand its capabilities and its limits and thoroughly enjoy the environment it offers.

TI LOGO CURRICULUM GUIDE



LAMPLIGHTER SCHOOL, INC.

The Lamplighter School, Inc., located in Dallas, Texas, is a private, non-profit school consisting of a preschool and grades one through four. The plaque above the front door displays the quote from Alexandrov, "A child is not a vessel to be filled but a lamp to be lighted." This is the thought that underlies the philosophy of educating young children at Lamplighter.

The school is designed in an L-shape, giving the classrooms at Lamplighter many windows and angles. Each classroom has a step-down "conversation well" for group and individual activities. Each group of three or four rooms clusters around an open, shared space. Teachers use this space as an extension of their classrooms. Usually, the computers are placed in this shared space for easy access by all classes.

As a playground, Lamplighter has a big barn with a loft for hay-jumping, a corral full of sheep, goats, horses, and other barnyard animals, fields for playing ball, and a "mountain" with a lookout and tunnel mine. These play areas were designed to give a child places to daydream, as well as places to release energy. The areas give a child opportunities to experience his or her environment at many levels: feel it, smell it, touch it, hear it, and even taste it.

While many of the outward signs of a school are nonexistent (report cards, bells signalling the end of a class, and four-walled classrooms with doors), the children learn in an environment individually tailored to meet their needs. This is not to imply that the children work separately. Children work in groups, some large and some small, sometimes with a teacher or perhaps sharing with another child. But the attitude that pervades the school is that each child is unique and individual.

Each grade level is taught by a team of teachers. While the team teaches mathematics, science, and grammar skills, several fine arts teachers assist in the weekly instruction of drama, music, library skills, motor development, French, and art. Recognizing that students learn in a variety of ways and at different paces, each team of teachers fashions the curriculum of each child to meet the diverse needs of that child. To accomplish this, Lamplighter is a leader in the use of new technology for teaching and learning. Electronic aids such as calculators, the *Speak & Spell*[™] learning aid, the *Little Professor*[™] learning aid, and the TI Home Computer are readily available and used daily by students and teachers.

TI LOGO CURRICULUM GUIDE

Lamplighter Computer Project

Erik Jonsson, a long-time friend of Lamplighter School, funded the development of the computer project that introduced Lamplighter to microcomputers. The 50 TI Home Computers placed throughout the school feature the TI LOGO computer language exclusively.

The teachers at Lamplighter find that with the TI LOGO language a different kind of individual learning environment can be created for and by the students. For the preschool learners, TI LOGO procedures or activities (included with the Guide) were designed that involve and challenge a child's development in a variety of areas — fine motor skills; letter, number, and color recognition; retention and recall capabilities; and task focusing. For the more developed students, writing their own procedures strengthens mathematical skills, logical thinking, and the ability to communicate precisely. The use of TI LOGO as part of the curriculum at Lamplighter supports the school's philosophy that learning should be purposeful and self-directed, and that each child develops skills to help him or her cope successfully with the world in which they live — a world that cannot be predicted.

Rather than replacing them, the teachers find that the computers in the classroom or shared areas assist, support, and facilitate learning. The teachers, along with the students, become computer learners. This step toward a situation in which the helper is also a learner increases the interaction and excitement of exploration between a teacher and a student and between fellow students.


The learners and users at Lamplighter find that learning to communicate with the computer through TI LOGO is a natural, enjoyable process. Many of the teachers think that it also enhances the way other types of learning take place. But they all believe that the use of TI LOGO contributes to the learner's positive self-image.

The faculty at the Lamplighter School strives to make teaching relate to the whole child, to look at learning through his or her eyes. Most of all they want all of their children to feel good about themselves.



USING THIS MANUAL

Before reading the *TI LOGO Curriculum Guide*, you should read the *TI LOGO User's Manual*. The *User's Manual* is written either for students to use as they work independently or as reinforcement material for your instruction. The first four chapters are written on a fourth-grade readability level. Due to the technical terms, the readability level of the last five chapters is higher. Reading this owner's manual also acquaints you with the few technical terms that are used in the *TI LOGO Curriculum Guide* and assumed to be familiar to you.

The *Curriculum Guide* is designed to help you understand how to teach TI LOGO to students from the preoperational through concrete levels of development. This encompasses an age range of approximately three through 12 years. In the explanation of each activity, the  symbol indicates what you say or ask your students. You can use the exact wording or paraphrase it to suit the vocabulary of your students. If an action or demonstration accompanies an explanation, the instructions appear in brackets.

The section "Developing a Classroom Strategy" provides information for determining the beginning level activity for students. The next section, "Using TI LOGO Activities," describes each file of activities and how to load the activities from diskette and cassette tape to the computer's memory.

The "Cue Cards" section explains what a cue card is, the design of one, and how to use one with a specific activity. The sections, "Preoperational Level of Understanding," "Early Concrete Level of Understanding," and "Concrete Level of Understanding" help you determine the level of understanding of your students. Each section contains information about appropriate activities and/or concepts of the TI LOGO language.

The *TI LOGO Reference Manual*, which is the last section in this book, provides a complete description of the TI LOGO computer language. This manual is designed to help you develop procedures for your students. As your students advance, they may want to review this material in order to develop more complex procedures.

This manual is for *you*, the classroom teacher and problem solver. Use it as the needs of the students indicate. Remember, the computer is a machine that you and students can manipulate to help solve problems that are important to the learner and are within the realm of the learner's skills at a given time.

TI LOGO CURRICULUM GUIDE

Developing A Classroom Strategy






DEVELOPING A CLASSROOM STRATEGY

Determine the correct level at which a student can begin to experience TI LOGO. This step helps create and maintain a comfortable, secure and unthreatening environment where learning is a "fun thing." By understanding the general tendencies and characteristics of students at the preoperational, early concrete, and concrete levels of development, you can identify an individual student's level of development and determine which activities support and extend that student's capabilities and development.

The following chart lists the appropriate activities for each developmental level. As the chart shows, an activity may be used for enrichment by students at an early developmental stage or for reinforcement by students with comprehension difficulties at a later stage of development.

ACTIVITIES BY DEVELOPMENTAL LEVEL

Developmental Level:	Preoperational		Early Concrete		Concrete		
	Nursery	Kindergarten	1	2	3	4	5
Activity							
<i>Turtle Mode</i>							
LINE							
DRAW							
AIM							
TARGET							
<i>Tile Mode</i>							
ROAD							
PAINT							
<i>Sprite Mode</i>							
PARK*							
PEOPLE*							
DALLAS*							
BUILD*							
MOVE							
HALT							
GRID _							
LISTEN							
MOVE _							
HALT							

enrichment 
 practice 
 reinforcement 

*Note that the PARK, PEOPLE, DALLAS, and BUILD activities are available in more than one file, depending on the number of options included. See the chart in the "Using the TI LOGO Activities" section for a detailed list of the activities.

TI LOGO CURRICULUM GUIDE

The age and level of development of your students also determine the amount of time you spend on each activity or concept. Some suggestions are to spend one short class time on a concept, and then, in the next class time, move on to a different one. Or spend one class time introducing the concept, another one sharing projects, and another class time playing games that reinforce the concept or help correct any misconceptions.

Playing games that help to internalize knowledge about TI LOGO is very important, especially to the younger child. Several TI LOGO games are included in this book, and a detailed description of each appears in the section about games. Where concepts are discussed in the Guide, any related games are referenced in the discussion.

Activities, concepts of the TI LOGO language, and experiences with the language can be shared with an individual student, a small group, or an entire class. The sequence of activities and concepts that appears in this *Curriculum Guide* is not mandatory — it's a suggestion. The *TI LOGO User's Manual* gives another possible sequence, allowing students with a greater level of cognitive and motor skill development to work at their own pace.

In a one-teacher, one-class situation, begin with an activity or a concept that the majority of your students can understand. Students who have already worked with this area can still benefit — they can offer suggestions, review concepts, pick up some new ideas, or dispell any misconceptions.

Each concept has suggested activities that are open enough so that every student can accomplish them at his or her current level of cognitive development and computer awareness. Yet the activities are structured enough to ensure that each student has some direction and can venture into a somewhat controlled experience as a starting point.

The vocabulary and terminology in the book are descriptive, yet appropriate for classroom use. It is important that computer learners (you and your students) develop a means of communication that is consistent at all levels of understanding. This gives them and you ways to think about the computer that relate to everyday experiences. Remember, have fun and help your students be successful.



USING THE TI LOGO ACTIVITIES

The activities on the diskette and cassette tape are designed to provide TI LOGO and computer readiness for young students at the preoperational and early concrete levels. They can also be used by students at any level of development as an introduction to TI LOGO and computers. Students with a greater degree of cognitive and motor skill development should progress through the activities more rapidly than students with a lesser degree of development.

The following chart lists the activities on the diskette and cassette tape included with the *Curriculum Guide*. It also indicates the file in which the activity is stored and the type and number of attributes a student has available to him or her. The letters that appear in bold type indicate the single-letter keystrokes a learner presses.

CONTENTS OF TI LOGO ACTIVITIES

File	Activity	Turtle/Sprites	Shape	Color	Direction	Speed
PREOP/1	LINE	Turtle			Forward Right Left	
	PARK	10 sprites	Car Garage	Red Blue Yellow White	↑ ← → ↓	
	PEOPLE	20 sprites	head torso arm arm legs	Red Blue Yellow White	↑ ← → ↓	
PREOP/2	DRAW	Turtle			Forward Back Right Left	
	DALLAS	10 sprites	Plane Truck	Red Blue Yellow White	→ ← → ↓	0 1 2 3 4 5
	BUILD	10 sprites	block	Red Blue Yellow White	↑ ← → ↓	

TI LOGO CURRICULUM GUIDE

File	Activity	Turtle/Sprites	Shape	Color	Direction	Speed
PREOP/3	PARK	31 sprites	Car Garage	Red Blue Yellow White Green Orange Purple	↑ ← → ↓	
	PEOPLE	24 sprites	head torso arm arm legs	Red Blue Yellow White Green Orange Purple	↑ ← → ↓	
	DALLAS	31 sprites	Plane Truck	Red Blue Yellow White Green Orange Purple	↑ ← → ↓	0
						1
						2
						3
						4
						5
						6
						7
						8
						9
	BUILD	31 sprites	block	Red Blue Yellow White Green Orange Purple	↑ ← → ↓	
	MOVE	any number of sprites				
	HALT	any number of sprites				
	ROAD	Turtle		red Nodraw Draw	Forward Back Right Left	

Using the TI LOGO Activities



File	Activity	Turtle/Sprites	Shape	Color	Direction	Speed
EARLYCON	AIM	Turtle			F__(0-29) B__(0-29) Right Left	
	PAIN_T	Turtle		Setcolor Turtle's ink	Forward Back Right Left	
				Color- background Paint Nopaint		
	GRID __	11 sprites numbered 15-25	any			
	LISTEN	sprite	any		↑ ← → ↓	
	MOVE __	sprite	any			0-49
HALT	sprite	any				
TARGET	Turtle				F__(0-100) B__(0-100) R__(0-360) L__(0-360)	

TI LOGO CURRICULUM GUIDE

Loading the Activities

STEP 1: Be sure that the Memory Expansion unit is connected to the computer and that the TI LOGO Command Module is inserted into the console. Also, be sure that the Disk Memory System or cassette recorder is attached to the computer and turned on. (See the owner's manuals or the *User's Reference Guide* for product details.)

STEP 2: Press any key to pass the master title screen. Then press the number corresponding to TI LOGO. When the question mark and cursor (a flashing black underline) appear, load the program by typing

RECALL

and pressing **ENTER**.

When the Recall selection list is displayed, press **3** for BOTH 1 AND 2 to load PREOP/1, PREOP/2, PREOP/3, or EARLYCON.

Next, the Device selection list appears on the display. To load the program from a cassette tape, insert the tape into the recorder. Next, refer to the "Loading Cassettes" section in this manual for instructions on determining the program's position on the cassette tape. When you have properly positioned the tape counter on your recorder, press **1**. The computer then displays directions for loading the program.

To load the program from diskette, insert the diskette into Disk Drive 1 and press **2**. Then press the **SPACE BAR** to review the file names of the programs.

When you reach the file you want, press **ENTER**. (See the *TI LOGO User's Manual* for additional information.)

The following files and activities are available.

<i>PREOP/1</i>	<i>PREOP/2</i>	<i>PREOP/3</i>	<i>EARLYCON</i>
LINE	DRAW	PARK	AIM
PARK	DALLAS	PEOPLE	PAINT
PEOPLE	BUILD	DALLAS	GRID _
		BUILD	LISTEN
		MOVE	MOVE _
		HALT	HALT
		ROAD	TARGET

STEP 3: When the question mark and cursor reappear, the computer is ready for a student to type the name of the activity and press **ENTER**.



Loading Cassettes

Copies of all programs listed on the cassette tape label are located on both sides of the tape. If for any reason you experience trouble loading or accidentally erase a program, another copy is available on the other side of the tape.

To attach and operate your cassette recorder, refer to the instructions found in the *User's Reference Guide*. Follow the instructions carefully, and the programs should load easily.

However, if your recorder does not respond when you press **ENTER** while loading the package, your cassette recorder's drive motor may not be compatible with the computer's circuitry. Although the computer may not be able to operate the recorder automatically, you may be able to operate your recorder manually. Connect the red and white plugs to the recorder unit as described in the *User's Reference Guide*, but do not connect the black plug. Follow the procedure for loading data as described. When the message "PRESS CASSETTE PLAY" is displayed, press the **ENTER** key immediately after pressing the recorder's PLAY key. If the data loads successfully, you may continue to operate the cassette manually.

Follow these steps to determine the exact location of all programs:

STEP 1: Rewind your tape and reset the counter to zero.

STEP 2: Disconnect the computer-to-cassette cable from the cassette player. You now can hear what is on the tape as it plays.

STEP 3: Press PLAY.

STEP 4: The programs in the *TI LOGO Curriculum Guide* package are listed on the cassette tape in the following order:

PREOP/1
PREOP/2
PREOP/3
EARLYCON

A blank section of tape precedes each program. When you hear program data, write the tape counter position beside the program name listed above. You may wish to subtract 1 or 2 from the number on the counter reading to ensure that, when you load the program, the beginning of your program loads properly.

STEP 5: Use these counter settings in the future to quickly load cassette tape programs.

Note: This process can be speeded by alternating between PLAY and FAST FORWARD as you listen.

TI LOGO CURRICULUM GUIDE



CUE CARDS

A "cue card" is a simplified set of instructions indicating what keys the learner presses and the results of pressing those keys. It can be used to introduce an activity to a student and to assist a student in working independently.

A cue card should reflect your students' current levels of understanding. For example, the style of the letters should be consistent with the development of your students. Use capital letters for children at the preoperational level and lower-case letters for the child at the early concrete level of development. Since the cue card is designed to help the young learner or a student with difficulties, the key presses should be single-letter designations that are consistent with the associated command in TI LOGO.

A copy of a cue card for each activity appears in *Appendix A*. At the bottom of each card is a note indicating which activity and file are appropriate for use with the card and giving permission for the card to be reproduced. (*Note: To give you a visual concept of a cue card, we suggest you look at the cards in Appendix A before continuing.*)

Designing a Cue Card

The general design of a cue card is the same for all activities and all levels of students. A cue card can be made on 9"x12" posterboard and covered with plastic wrap for protection. Write the activity's name on the top of the card followed by the word **ENTER**. (*Note: The word ENTER can be colored orange to represent the ENTER key on the TI-99/4 console or with a yellow dot drawn next to the word to represent the ENTER key on the TI-99/4A console.*) Draw a solid black line below the title line of the activity.

The information appearing below the solid line indicates the attribute choices available for the Turtle or sprite.

Choices for Turtle Activities

The first letter of the commands for moving and turning the Turtle should be the first options you place under the title and the solid line. Write the options available for giving the Turtle color or changing the *state of the pen* below the direction choices. Draw a solid black line under the options in the activity.

Write "Q = Quit" under the solid line, indicating that, once this option is selected by typing the letter **Q**, the activity stops. To restart a Turtle activity, the child looks at the top of the cue card and repeats the steps of typing the name of the activity and pressing **ENTER**.

Choices for Sprite Activities

If the activity offers a choice of shapes for a sprite to carry, draw the corresponding letter and the graphic of the shapes on the card. Then, draw a dotted black line under the shape choices, indicating that the options that follow apply to the sprite that carries the selected shape.

If color is an attribute option, use the first letter of the color as the designated keystroke. This reinforces the association of the initial sound of the word for that color. To the right of each letter representing a color is a circle. Color the circle the indicated color. This helps a child recognize the meaning of the letter.

TI LOGO CURRICULUM GUIDE

If direction is an option, draw up, down, right, and left arrows in a column to the right of the colored circles. In an activity offering the option of speed, write the numbers representing the available speeds in a column to the right of the arrows.

Putting the choices in this left-to-right order of color, direction, and speed encourages a child to give the sprite the attributes in that order. This sequence is consistent with the sequence a child uses when giving a sprite attributes using the TI LOGO commands.

If more than one sprite is available in an activity, draw a dotted line to separate the attribute choices from the "A = Another" (sprite) choice. The dotted line indicates that the options appearing below alter the current *state of the sprite*, but do not stop the activity. Draw a solid black line under the options in the activity.

Write the prompt for stopping an activity, "Q = Quit", under the solid black line. To play the activity again, the student looks at the top of the cue card and repeats the steps of typing the name of the activity and pressing **ENTER**.

Introducing an Activity with a Cue Card

To begin, introduce the cue card concept to each student individually or to a small group of students.

SAY: The computer can do activities but doesn't know which activity to do until you type the name of the activity. [Show a cue card.] This is a cue card. It helps you tell the computer what to do.

With a child seated in front of the computer, point to and say each letter in the activity's name. Encourage the child to say the letter while he or she locates and types it. Children may have difficulty in recognizing and then locating the correct key. Coding each row of keys with colored tape often helps to minimize this problem. You can help the child focus on the correct row by saying, for example, "The letter A is in the red row."

When children begin typing on the computer keyboard, they often make typing errors because of their limited physical capabilities. As you introduce the first TI LOGO activity, you may want to tell the student that he or she can correct a typing error by pressing **ERASE** before pressing **ENTER** and then retyping the name of the activity. If you choose not to explain how to correct a typing error, tell your student that, if a typing error is made and **ENTER** is pressed, a message appears on the display telling him or her that the computer did not understand the name that was typed. Then tell your student to retype the name correctly. After the name of the activity is typed correctly,

SAY: Now press the **ENTER** key; this tells the computer to start the activity.

Tell each student that he or she has choices to make with each activity. Review the choices related to the activity that is being introduced. For the sprite activities, placing your finger on the display assists the child in determining the position of a sprite, thus avoiding the placement of more than four sprites on the same horizontal row. When you think a student understands the concept of giving attributes to one sprite, introduce the ability to talk to another sprite by pressing **A** for "Another."

Explain to each student that he or she can stop an activity at any time by pressing **Q** for "Quit." To play the activity again,

SAY: All you need to do is type the name of the activity and press **ENTER**; then follow the instructions that are on the cue card.



PREOPERATIONAL LEVEL OF UNDERSTANDING

The TI LOGO procedures for students at the preoperational level of development should meet specific educational objectives by providing experiences that encourage a student's cognitive development. These experiences come from TI LOGO activities that are "prompters" and/or that support the classroom curriculum.

An activity that is a TI LOGO prompter allows the child to explore the graphics capabilities of the computer with a minimum of key presses. Some areas in which the activities can support the classroom curriculum are vocabulary, exploration of the senses, large and fine motor skill development, and time and space concepts.

The activities included with this Guide were designed to provide students with a beginning knowledge of TI LOGO and several different experiences with the language. This combination should, in turn, help students make a smooth transition into "teaching" the computer with TI LOGO commands.

Each activity was also designed to meet a series of general objectives. These objectives include:

- Recognizing letters, numbers, and colors.
- Associating a letter with a word and the initial sound of the word, such as "R" represents red and is the initial sound of the word red.
- Making judgments about how far to move the Turtle and sprites.
- Following directions by reading a cue card.
- Working independently.
- Encouraging the development of a longer attention span by focusing on a task.
- Attending to detail.
- Developing reading readiness.
- Developing decision-making skills.
- Encouraging divergent thinking: an infinite number of possible correct answers allow children to use variables creatively.

The next sections describe each activity in detail. In addition to the above-mentioned objectives, specific objectives for each activity are listed with its description.

Activities for Preoperational Level Students

LINE, PARK, and PEOPLE, the three activities in the PREOP/1 file, are designed to introduce students to the Turtle and the sprite modes. They also provide students with an environment for discovering directionality; letter, number, and color recognition; and creative decision making.

DRAW, DALLAS, and BUILD, the three activities in the PREOP/2 file, extend students' concepts of the Turtle and the sprites. The activities also provide students with an environment for exploring space, turns, and relationships.

PARK, PEOPLE, DALLAS, BUILD, MOVE, HALT, and ROAD, the seven activities in the PREOP/3 file, are designed for students who have had successful experiences with the Turtle and sprites. The PARK, PEOPLE, DALLAS, and BUILD activities in this file provide additional choices of color and/or speed. Combining MOVE and HALT with one of the four activities adds the dimension of movement to the activities.

With the ROAD activity, a student tells the Turtle to draw or not to draw a wide red line. The capability of moving the Turtle without drawing and then telling it to draw again helps a student create interesting designs.

TI LOGO CURRICULUM GUIDE

LINE Activity

Objectives

- Introduces the TI LOGO Turtle mode.
- Introduces directionality (up, down, right, left).
- Encourages recognition of upper-case letters.
- Encourages creative decision making.

Materials

- Program diskette or cassette tape with PREOP/1 loaded into the computer.
- LINE cue card (for a reproducible copy, see *Appendix A.*)

Introducing LINE

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After LINE is entered, the name of the activity disappears and the Turtle appears at HOME.

Playing LINE

SAY: The triangle is called a Turtle. It is in the middle of the display, a place called HOME. But you can move the Turtle by pressing three special keys. Look at the Turtle. Can you see which direction it is facing? [The Turtle always faces north when the LINE activity begins.] To move the Turtle forward, press the letter **F**. [The student should do this.] Look! The Turtle left a black line. Move the Turtle forward some more by pressing **F** again.

When the student understands this concept, introduce the ability "to make" the Turtle turn left and right by pressing **L** for left and **R** for right. (Note: The Turtle draws a line 10 Turtle steps long each time **F** is pressed and makes a 45° turn each time **R** or **L** is pressed. Some students may have difficulty seeing the turn the Turtle makes. Telling the Turtle to turn several times usually eliminates the visualization difficulty.)

After turning the Turtle,

SAY: Now, to move the Turtle forward, press **F**. To turn the Turtle to the right again press **R**, and then press **F** to move it forward again.

Continue giving instructions until you feel the student can move and turn the Turtle following the cue card.

Ending LINE

SAY: When you finish drawing with the Turtle, press **Q** for quit.

Replaying LINE

SAY: To play LINE again, look at the top of the cue card, type the letters L-I-N-E and then press **ENTER**. The Turtle appears at HOME and is ready to listen to your instructions.

Preoperational Level of Understanding



PARK Activity

Objectives

- Introduces the sprite mode.
- Introduces the concept of giving the attributes of color and direction to a sprite.
- Encourages decision-making skills.

Materials

- Program diskette or cassette tape with PREOP/1 loaded into the computer.
- PARK cue card (for a reproducible copy, see *Appendix A*).

Introducing PARK

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After PARK is entered, the name of the activity disappears and the display is empty.

Playing PARK

SAY: The activity PARK lets you pick a car or a garage by pressing **C** for car or **G** for garage. When you press **C**, a black car appears at HOME, the center of the display. You can move the car up, left, right, and down by pressing the arrow keys [↑, ←, →, and ↓].

Since all the sprites initially appear at HOME, encourage your students to move the sprites away from HOME. If the sprites are not moved, the students cannot see the new sprite when "A" for another is pressed and a selection of a garage or car shape is made.

SAY: You can also change the color of the car by pressing **R** for red, **B** for blue, **Y** for yellow, and **W** for white. Once the color of the shape is changed, it cannot be colored black again.

When the student has positioned and colored the car to his or her liking,

ASK: Are you ready to see another shape? Press **A** for another, and let's pick a garage. Press **G** and a black garage appears at HOME. You move the garage by pressing up, down, right, and left arrow keys. And you change the color of the garage by pressing **R** for red, **B** for blue, **Y** for yellow, and **W** for white.

SAY: You can put 10 cars or garages on the display by pressing **A** for another and then **C** for a car, or **G** for a garage. [Note: The number of sprites available is determined by the level of the activity selected. In PREOP/1 10 sprites are available.]

Ending PARK

SAY: You can stop playing PARK any time by pressing **Q** for quit. Or, after you put 10 cars and garages on the display, **OUT** appears in the upper left-hand corner.

Replaying PARK

SAY: To replay PARK, look at the top of the cue card, type the name of the activity and press **ENTER**.

TI LOGO CURRICULUM GUIDE

PEOPLE Activity

Objectives

- Expands the concept of giving attributes to sprites.
- Encourages making a small task part of a large one.

Materials

- Program diskette or cassette tape with the PREOP/1 file loaded into the computer.
- PEOPLE cue card (for a reproducible copy, see *Appendix A*).

Introducing PEOPLE

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After PEOPLE is entered, the name of the activity disappears and a black face appears at HOME.

Playing PEOPLE

SAY: The activity PEOPLE lets you build "people" on the display.

■ There are five parts to each body — the head, the torso, the left arm, the right arm, and the legs. [Demonstrate each part of the body as you say it.] You need to tell each part of the body what color to be by pressing **R** for red, **B** for blue, **Y** for yellow, or **W** for white. You can also move the shape up by pressing the up arrow key, down by pressing the down arrow key, right by pressing the right arrow key, and left by pressing the left arrow key. [Point your finger in the appropriate direction as you say it.] When you are ready for the next part of the body, press **A** for another, and the next part appears at HOME. [Note: Remind your students to move each part away from HOME. See the "CONCEPT: No More than Four Sprites in a Row" section. Also, remind them if the color of a part of the body is changed, that part cannot be made black again.]

SAY: You can build up to four people on the display.

Ending PEOPLE

SAY: After four people are built or 20 shapes are on the display, ■ **OUT** appears in the upper left-hand corner. You can stop the activity at any time by pressing **Q** for quit.

Replaying PEOPLE

SAY: To play PEOPLE again, look at the top of the cue card, type ■ the name of the activity and press **ENTER**. A black face appears at HOME.

Preoperational Level of Understanding



DRAW Activity

Objectives

- Provides readiness for the FORWARD, BACK, RIGHT, and LEFT commands.
- Provides readiness for the PENUP command.
- Encourages exploration of space, turns, and relationship of lines by creating designs.

Materials

- Program diskette or cassette tape with the PREOP/2 loaded into the computer.
- DRAW cue card (for a reproducible copy, see *Appendix A*).

Introducing DRAW

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After DRAW is entered, the name of the activity disappears and the Turtle appears at HOME.

Playing DRAW

SAY: The DRAW activity is like the LINE activity except that you can tell the Turtle to do something new.

Ask your students to remember what keys were pressed in the LINE activity to move and turn the Turtle. After the keys have been named correctly,

SAY: The same keys, **F**, **R**, and **L**, move and turn the Turtle in this activity. But now you can also move the Turtle back by pressing **B**. [Note: You may ask your student to guess what letter would move the Turtle back by emphasizing the initial sound of the first letter of the other special keys and then emphasizing the "b" sound for "back."]

Now let the students experiment with moving and turning the Turtle. When the students are comfortable with these keys,

ASK: Wouldn't it be fun and interesting if you could move the Turtle without it drawing a line? You can. The special key that tells the Turtle not to draw is **N**, for nodraw. Press **N**. Can you tell that the Turtle isn't drawing? (ANSWER: No.) What can you do to the Turtle to see that it's not drawing? That's right, move it.

Encourage the student to move and turn the Turtle several times.

ASK: Are you ready for the Turtle to draw again? Just press **D** for draw and the Turtle is ready to draw. Try it. But remember, you won't see it drawing until you move it. [Note: Now is a good time to suggest that the Turtle draws much in the same way you write with a pen. This prepares the student for the *state of the pen* concept and the commands PENUP and PENDOWN. For more information, see the "CONCEPT: State of the Pen" section.]

Ending DRAW

SAY: When you finish drawing with the Turtle, press **Q** for quit.

Replaying DRAW

SAY: Look at the top of the cue card, type the letters D-R-A-W and then press **ENTER**. The Turtle appears at HOME, ready to listen to your instructions.

TI LOGO CURRICULUM GUIDE

DALLAS Activity

Objectives

- Introduces the attribute of speed for sprites.
- Demonstrates the concept of speed by relating a numerical value to the corresponding speed of a sprite.

Materials

- Program diskette or cassette tape with the PREOP/2 file loaded into the computer.
- DALLAS cue card (for a reproducible copy, see *Appendix A*).

Introducing DALLAS

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After DALLAS is entered, the name of the activity disappears and the display is empty.

Playing DALLAS

SAY: Just as you did in the PARK activity, you need to tell the computer what shape to show. You can choose a truck or a plane. To see a truck, press **T**; to see a plane, press **P**. [Help each student locate these new keys.] You change the color of the truck or the plane by pressing **R** for red, **B** for blue, **Y** for yellow, and **W** for white. But remember, once you change the color from black, you cannot change the object to black again.

An exciting attribute of a plane is that it flies, and an exciting attribute about a truck is that it moves. You can also make your plane fly and your truck move. You can make them go fast, or you can make them go slow. You type a number to tell the plane and the truck how fast to go. Five is the fastest speed.

ASK: What do you think a slow speed is? (ANSWER: 1, 2, 3, or 4.)

■ What speed would you give the truck or plane if you want it to stop? That's right, 0 speed, so type the number **0**. Your plane can also fly up, down, left, or right. Press an arrow key to tell the plane in which direction to fly. Use the same arrow keys to tell the truck in which direction to move.

SAY: There are many attributes you can give a truck or a plane.

■ Remember to give all the attributes to a shape before pressing **A** for another. You cannot go back and change an attribute of a truck or plane after you press **A**.

Ending DALLAS

SAY: You can put a total of 10 trucks and planes on the display.

■ You can stop the activity at any time by pressing **Q** for quit.

Replaying DALLAS

SAY: To replay DALLAS, look at the top of the cue card, type

■ D-A-L-L-A-S and press **ENTER**.

Preoperational Level of Understanding



BUILD Activity

Objective

- Encourages making a small task part of a larger one.

Materials

- Program diskette or cassette tape with the PREOP/2 loaded into the computer.
- BUILD cue card (for a reproducible copy, see *Appendix A*).

Introducing BUILD

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After BUILD is entered, the name disappears and a black block appears at HOME.

Playing BUILD

SAY: BUILD lets you build designs on the display like you build with building blocks on the floor.

ASK: What keys do you think you press to change the color of a block? That's right, **R** for red, **B** for blue, **Y** for yellow, and **W** for white. And what keys do you press to move a block? Correct, **↑**, **←**, **→**, and **↓**. Remember to give all the attributes to a block before getting another block. Do you know what key you press to see another block at HOME? Good, the **A** key.

Encourage your students to move the blocks away from HOME and to avoid putting more than four on the same "step." See the "CONCEPT: No More Than Four in a Row" section.

Ending BUILD

SAY: You can stop playing BUILD at any time by pressing **Q** for quit. After a total of 10 blocks are on the display, **OUT** appears in the upper left-hand corner.

Replaying BUILD

SAY: To play BUILD again, look at the top of the cue card, type the name of the activity and then press **ENTER**. A black block appears at HOME.

TI LOGO CURRICULUM GUIDE

MOVE and HALT Activities

Objective

- Provides readiness for the TELL :ALL command by giving the same attribute to several sprites simultaneously.

Materials

- Program diskette or cassette tape with the PREOP/3 file loaded into the computer.
- MOVE and HALT cue cards (for a reproducible copy, see *Appendix A*).

Introducing MOVE and HALT

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed.

ASK: Have you ever made a design that you'd like to see move?

- The MOVE activity tells the design on the display to move to the right. What if you want to stop it? Then you choose the HALT activity. After typing the letters H-A-L-T and pressing ENTER, the design stops immediately.

Playing MOVE and HALT

SAY: Make a design with the PARK, PEOPLE, DALLAS, or BUILD activity. [This may take several minutes for the students to complete. However, an activity does not have to be completed. Q can be pressed and then MOVE and HALT used with the design.]

- **SAY:** Look at the cue card and type MOVE. When it is typed correctly, press ENTER. Watch...your design moves to the right. Are you ready to stop it? Type HALT and press ENTER.

Ending MOVE and HALT

This activity ends when the student enters the name of another activity.

Replaying MOVE and HALT

Students can MOVE and HALT a design as frequently as they like.

Preoperational Level of Understanding



ROAD Activity

Objectives

- Extends the concept of the Turtle by drawing on the width of a tile.
- Provides readiness for the PENUP and PENDOWN commands.
- Introduces the capability of adding color to the tile on which the Turtle draws.
(Note: Young children need the large degree of turn to see the effects of the Turtle drawing on the width of a tile.)


Materials

- Program diskette or cassette tape with the PREOP/3 file loaded into the computer.
- ROAD cue card (for a reproducible copy, see *Appendix A*).

Introducing ROAD


Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. After ROAD is entered, the name of the activity disappears and the Turtle appears at HOME.

Playing ROAD


SAY: The ROAD activity is like the DRAW activity except you can  tell the Turtle to draw or not to draw a wide red line.

Review the special keys for moving and turning the Turtle (**F**, **B**, **R**, and **L**). Let each student experiment not drawing with the Turtle by pressing **N**, and then drawing again by pressing **D**. If a student is having difficulty moving the Turtle, you may want to suggest a simple design or project.

Ending ROAD

SAY: When you finish drawing with the Turtle, press **Q** for quit.


Replaying ROAD

SAY: Look at the top of the cue card, type the letters R-O-A-D, and  then press **ENTER**. The Turtle appears at HOME and is ready to listen to your instructions.

TI LOGO CURRICULUM GUIDE

Early Concrete Level of Understanding



EARLY CONCRETE LEVEL OF UNDERSTANDING

Cognitive development does not move from one discrete level to another. Instead, it encompasses many areas of intellectual growth. The student absorbs new information and integrates it into his or her existing schemation, but new understandings may or may not directly affect all areas of cognition. The early concrete level of understanding refers to the time when the child begins to make logical conclusions but is still influenced by perception.

The student at this level of understanding may be able to plan elaborate designs with the four activities PARK, PEOPLE, DALLAS, and BUILD. Encourage your students to build a huge box, or giant stairs, or a window. Three “people” may stand on top of other people’s heads in the PEOPLE activity in a totem pole effect. Four garages and cars may be built on four different streets in the PARK activity. One simple idea from you can usually stimulate several from a student.

The activities for the early concrete level student are designed to continue supporting the general objectives outlined for the preoperational level activities and provide additional readiness for a student “teaching” the computer.

Adapting Cue Cards

The cue cards for the early concrete level student should reflect the additional options in the color and speed categories. They should also support the majority of the students’ current level of alphabet recognition (lower-case or cursive letters).

The cue cards may have the initial letter of the shape or color, or a graphic of the shape with color minus the initial letter of the word.

The last two cue cards a student should require is one that demonstrates how to talk to one sprite and one that tells the student how to talk to more than one sprite. (*Note: For a reproducible copy of these cue cards, see Appendix A.*)

TI LOGO CURRICULUM GUIDE

Activities for the Early Concrete Level Student

The PARK, BUILD, PEOPLE, and DALLAS activities in the PREOP/3 file are appropriate for students at the early concrete level of understanding. The activities adapted for the early concrete learner give the student eight choices, rather than five, in the color category. The colors are black (the initial color of the shape), red, blue, yellow, white, green, orange, and purple. At this stage, all students should be selecting from eight colors. Some students may need an introduction to the three new colors. Rather than telling them what key to press, help them figure it out by verbalizing the initial sounds of the colors they already know. Then say the name of the new colors, helping your students deduce the initial sound and thus the correct key to press (**G** for green, **O** for orange, and **P** for purple). This process also helps prepare students for deducing the short form for TI LOGO commands.

There are ten options in the speed category of the DALLAS activity. If this is a student's first experience with the full range of speed options, help him or her discover the additional speed options and the appropriate keys to press (**6** equals a speed of 60, **7** equals a speed of 70, **8** equals a speed of 80, and **9** equals a speed of 90).

The activities in the EARLYCON file, AIM, PAINT, GRID __, LISTEN, MOVE __, HALT, and TARGET, reinforce earlier experiences with the Turtle and sprites and provide an avenue for new experiences. The AIM and TARGET activities introduce the concept of estimating the number of steps it takes the Turtle to reach a particular spot. Playing AIM before TARGET provides students with the skills necessary for a smooth transition from turning the Turtle with a single key press to giving the Turtle a variable: the degree of turn.

The MOVE and HALT activities are also available to the learner. They allow the student to move or stop the design created in one of the four preoperational activities. On the cue card, print the word MOVE in green ink and the word HALT in red ink. At this level, the MOVE activity requires a number. The student types the word MOVE, presses the **SPACE BAR**, and chooses a number from the cue card. Then he or she presses **ENTER**.

The GRID __ activity lets a student use a 16-by-16 square grid to design a shape for a sprite to carry. As a design is drawn on the grid using the arrow keys, a sprite shows the student the actual size of the design. The activity provides readiness for the MAKESHAPE mode. The LISTEN activity then lets a student move his or her design.

Early Concrete Level of Understanding



AIM Activity

Objectives

- Encourages an understanding of the value of the numbers 0 through 29.
- Encourages decision making by making judgments about how far to move the Turtle.
- Provides readiness for giving the Turtle FORWARD and BACK commands.
- Introduces the concept of using a variable with the FORWARD command.

Materials

- Program diskette or cassette tape with the EARLYCON file loaded into the computer.
- AIM cue card (for a reproducible copy, see *Appendix A*).

Introducing AIM

Be sure the concept of a cue card has been introduced and the procedure for "Introducing an Activity with a Cue Card" has been followed. Show the students where the **SPACE BAR** is and explain that pressing the **SPACE BAR** types a blank space.

ASK: If you were typing your name and wanted to type a blank space between your first and last name, what key would you press? That's right, the **SPACE BAR**. How many times do you think you'd press it? You press it one time for each blank space.

SAY: In this activity you are going to be telling the Turtle to move forward and back by pressing **F** and **B**. You also have to tell the Turtle the exact number of steps to move. A space has to be typed between **F** and the number of steps or **B** and the number of steps. For example, to move the Turtle FORWARD 5 steps, you type **F**, a space, and **5**. Then, just as you do when you finish typing the name of the activity, you press **ENTER**.

After AIM is entered, the name of the activity disappears and the Turtle draws a double ring and then positions itself randomly on the display.

Playing AIM

SAY: The object of the AIM activity is to move the Turtle into the double ring. Remember, you have to tell the Turtle the exact number of steps to take forward or back. For example, the Turtle takes a small step when you type **F**, a space, the number **2**, and then press **ENTER**. The largest step the Turtle can take is 29.

ASK: Can you tell the Turtle how to take a large step back? That's correct. Type **B**, a space, **25**, and then press **ENTER**. [*Note:* Remind your students that they can erase a typing mistake by pressing **ERASE** before pressing **ENTER**, and then retyping the command correctly.]

SAY: If you give the Turtle a number larger than 29, the Turtle spins around in a circle. This tells you that the Turtle doesn't understand that number. When the Turtle stops spinning, give the Turtle a new command using a number between 0 and 29.

TI LOGO CURRICULUM GUIDE

Ending AIM

SAY: To turn the Turtle to the right, press **R**. What key do you think you press to turn the Turtle to the left? That's correct. Press **L**. When you put the Turtle inside the double ring by moving and turning it with the correct keys, a colored circle replaces the center ring. To stop the activity at any time, press **Q** for quit.

Replaying AIM

SAY: Look at the top of the cue card, type the name of the activity and then press **ENTER**. The Turtle draws a double ring, places itself on the display, and is ready to listen to your instructions.

Early Concrete Level of Understanding



PAINT Activity

Objectives

- Provides readiness for the SETCOLOR and COLORBACKGROUND commands.
- Provides new experience with the Turtle by letting the student change the color of the painting bar and the display.
- Encourages creative "paintings."

Materials

- Program diskette or cassette tape with the EARLYCON file loaded into the computer.
- PAINT cue card (for a reproducible copy, see *Appendix A*).

Introducing PAINT

Be sure the concept of a cue card has been introduced and the procedure for "Introducing an Activity with a Cue Card" has been followed. To introduce the new options available in the PAINT activity,

SAY: In this activity, the Turtle draws a wide line, just like it did in the ROAD activity. Do you remember what color the Turtle drew with? (ANSWER: Red.) It would be fun and interesting if the Turtle could draw with other colors. It can. The computer knows 16 colors. The special key that changes the color the Turtle draws with is **S**. [Help your students find the **S** key.] Each time you press **S**, the color changes. If you press it more than 16 times, the colors repeat.

In all the activities, the background color, or color of the display, has been a blue color called cyan. But the computer knows 16 colors. Do you think you can change the background color? (ANSWER: Yes.) That's right. The special key for coloring the background is **C**. To see each color change, press **C** 16 times.

ASK: What happens when a bar the Turtle draws is the same color as the background? (ANSWER: The bar cannot be seen.) That's correct. To see the bar, press **C** to change the color of the background. That might make a different bar disappear.

SAY: Like with the ROAD activity, you can tell the Turtle when to paint and not to paint. The special key you press for telling the Turtle not to paint is **N** [nopaint]. When you move the Turtle after pressing **N**, it doesn't paint a bar. What key do you think you press to tell the Turtle to paint again? That's correct. Press **P**.

SAY: After PAINT is entered, the name of the activity disappears and the Turtle appears at HOME.

Playing PAINT

Encourage your students to move and turn the Turtle with **F**, **B**, **R**, and **L**. Also encourage them to change both the color with which the Turtle draws by pressing **S** and the background by pressing **C**.

Ending PAINT

SAY: To stop playing PAINT, press **Q** for quit.

Replaying PAINT

SAY: Look at the top of the cue card, type the name of the activity, and press **ENTER**. The Turtle appears at HOME.

TI LOGO CURRICULUM GUIDE

GRID __ Activity

Objectives

- Provides readiness for the MAKESHAPE mode.
- Provides readiness for using the special keys for designing a shape.

Materials

- Program diskette or cassette tape with the EARLYCON file loaded into the computer.
- GRID __ cue card (for a reproducible copy, see *Appendix A*).
- Grid paper on which students draw designs they intend to draw on the computer.
- Large grid paper for demonstrating how to make a design.

Introducing GRID

On large grid paper or a grid that has been drawn on a chalkboard, have your students help color in a number of squares to make a design.

SAY: You make a design on the computer the same way we colored the squares on this large grid.

Playing GRID __

SAY: There are 11 grids on which you can make designs. The numbers are 15 to 25. To help the computer remember what grid your design is on, give it a number.

While demonstrating on a computer,

SAY: Type GRID, a space, and then a number from 15 to 25. [Ask a student to give you an appropriate number and then type that number. Next, press **ENTER**.] Look! The display is green. That means that the computer is ready to be taught something.

SAY: The grid is just like the one we drew on except there's a cursor, the flashing square in the upper left-hand corner. That's what you move to color or not color the squares. You move the cursor with the arrow keys. To color a square, hold down the **SHIFT** key (on the TI-99/4 console) and press an arrow key, or hold down the **FCTN** key (on the TI-99/4A console) and press an arrow key. The square the cursor leaves stays black. [Demonstrate this by moving the cursor several squares.]

SAY: To move the cursor without coloring a square, press an arrow key without holding down **SHIFT** or **FCTN**. [Demonstrate this by moving the cursor without coloring in the squares.]

SAY: If you want to erase the entire grid and start a new design, press **CLEAR** (**SHIFT C** on the TI-99/4 console or **FCTN 3** on the TI-99/4A console). The grid erases and the cursor moves back to the upper left-hand corner. Now you can redesign the shape.

Notice that as you design the grid, the shape appears to the right of the grid. That shows you the real size of your design. Practice drawing designs on grid paper. Then teach the computer the designs you drew.

Early Concrete Level of Understanding



Ending GRID __

SAY: After you finish your design, press **BACK** to return to the cyan screen. You can now use the new design with the activities **MOVE** __, **HALT**, and **LISTEN**.

Replaying GRID __

SAY: To make another design, type **GRID**, a space, and a number from 15 to 25. Then press **ENTER**. If you select a grid that already has a design on it, do not erase that design. Press **BACK** and retype the name of the activity with a different number.

TI LOGO CURRICULUM GUIDE

LISTEN Activity

Objective

- Provides readiness for giving commands to a shape made in the MAKESHAPE mode.

Materials

- Program diskette or cassette tape with the EARLYCON file loaded into the computer.
- LISTEN cue card (for a reproducible copy, see *Appendix A*).

Introducing LISTEN

Be sure a design has been made using the GRID _ activity.

SAY: You can move the new shape you designed with the LISTEN activity. Type LISTEN and press ENTER. The last design you made with the GRID _ activity is ready to listen to forward, back, right, and left commands.

Press the arrow key that shows the direction in which you want the design to move eight steps. For example, to move the design up, which arrow do you press? That's right, the arrow that points up.

Ending LISTEN

SAY: To stop the activity at any time, press Q for quit.

Replaying LISTEN

SAY: To play LISTEN again, be sure that you have made a design with the GRID _ activity. Then type LISTEN and press ENTER.

Early Concrete Level of Understanding



MOVE __ and HALT Activities

- Objective** ■ Provides readiness for giving commands to a shape made in the MAKESHAPE mode.
- Materials** ■ Program diskette or cassette tape with the EARLYCON file loaded into the computer.
■ MOVE __ cue card with MOVE __ written in green ink and the HALT cue card with HALT written in red ink (for a reproducible copy, see *Appendix A.*)
- Introducing MOVE __ and HALT** **SAY:** These two activities work almost like the MOVE and HALT activities you used with PARK, PEOPLE, DALLAS, and BUILD. The difference is with the MOVE __ activity. Now you have to tell the computer how fast to move.
- Playing MOVE __ and HALT** **SAY:** With a design created with the GRID __ activity shown on the display, type MOVE, a space, and then a number from 0 to 49. If you type 45, do you think the design will move fast or slow? (ANSWER: Fast.) [Continue by asking similar questions that correlate the number entered to the resulting speed of the design.]
- Ending MOVE __ and HALT** **SAY:** When you are ready to stop your design, type HALT and press **ENTER**. There's another way you can stop a design. Can you think of it? (ANSWER: Type MOVE 0 and press **ENTER**.)
- Replaying MOVE __ and HALT** **SAY:** Be sure a design created with the GRID __ activity is shown on the display. Type MOVE, a space, and a number. Then press **ENTER**.

TI LOGO CURRICULUM GUIDE

TARGET Activity

Objectives

- Demonstrates an understanding of the value of making up to 360 degree turns with the Turtle.
- Provides readiness for giving the Turtle the RIGHT and LEFT commands.
- Introduces the concept of using a variable with the RIGHT and LEFT commands.

Materials

- Program diskette or cassette tape with the EARLYCON file loaded into the computer.
- TARGET cue card (for a reproducible copy, see *Appendix A*).

Introducing TARGET

Be sure the concept of a cue card is introduced and the procedure for "Introducing an Activity with a Cue Card" is followed. Repeat the concept of typing a space between the letter representing a command and the number and then pressing **ENTER**. Remind your students that they can erase a typing mistake by pressing **ERASE** before pressing **ENTER**. After TARGET is entered, the name of the activity disappears, and the Turtle draws a circle and then positions itself randomly on the display.

Playing TARGET

SAY: The object of the TARGET activity is to tell the Turtle to hit the target. In this activity you have to tell the Turtle the exact number of steps and the size of the turn to make.

SAY: If a number larger than 100 for Forward and Back or 360 for Right and Left is used, the Turtle does not move. [Tell your students these limitations. This information helps a student know exactly why something happens — thus avoiding the Turtle's having a "magical" effect.]

Ending TARGET

SAY: When you move the Turtle into the target, the background flashes colors and the activity is finished. To stop the activity at any time, press **Q** for quit.

Replaying TARGET

SAY: To replay TARGET, type the name of the activity and press **ENTER**. The Turtle draws a circle, positions itself on the display, and is ready to listen to your instructions.

Early Concrete Level of Understanding



Concept: Primitives with the Turtle

If your students have done the activities included with this package, they should require only a brief verbal introduction to the Turtle mode. If they haven't completed the activities, play the game "Introducing the Turtle Mode" in the "Games the Turtle Plays" section.

When your students start using TI LOGO commands, they need to give the Turtle the exact number of Turtle steps and Turtle turns. To get the Turtle's attention, type TELL TURTLE and press **ENTER**. Any writing on the display disappears, and the Turtle appears in the middle of the display, a place called HOME. A question mark and cursor appear in the lower left-hand portion of the display.

Concepts your students will encounter in the Turtle mode (either for the first time or after they have played LINE, DRAW, AIM, PAINT, and TARGET) are as follows.

- FORWARD and BACK are the only two commands that cause the Turtle to draw a line on the display. RIGHT and LEFT change the Turtle's heading or the direction it faces. The Turtle has to be told exactly how many steps or turns to take. These four commands have to be followed by a blank space and then a number.
- All commands in the Turtle mode are relative to the direction the Turtle faces. If the Turtle is facing the top of the display, your right and left are the same as the Turtle's. If the Turtle is facing the bottom of the display, the Turtle's right and left are the opposite of yours. When the Turtle is facing the bottom of the display a student can turn his or her body upside down to see which way the Turtle sees FORWARD, BACK, RIGHT, and LEFT.
- If the computer displays OUT OF INK, the Turtle's pen has run out of ink. To draw more lines, type CLEARSCREEN and press **ENTER**. All the Turtle lines and typing on the display are erased and the Turtle goes HOME.

Bugs your students might encounter

- ☞ A common bug is to use the word backward for the BACK command.
- ☞ Some students enjoy giving the Turtle very large numbers. If the number is too large, the Turtle won't move. At other times, the Turtle draws as much of the line as it can before it runs OUT OF INK. When it draws very long lines, it is hard to predict where the Turtle is and where it is going. Since the student seems to have no control over the Turtle, the computer appears "magical." One way to discourage this is to tell students to only give the Turtle numbers they can read. Hopefully the students will see the power of controlling the Turtle by giving it manageable numbers of steps and turns.

Things to do

- ✓ Create a very simple design on paper. Then tell the Turtle to draw a similar design on the display. See "Paper Turtle" in the "Games the Turtle Plays" section for more ideas.
- ✓ Give the computer some FORWARD, BACK, RIGHT, and LEFT commands using random numbers. Look at the random design. Complete it by giving the Turtle the commands that change the random design into a specific design.
- ✓ Design your initials or favorite letters of the alphabet with the Turtle.

TI LOGO CURRICULUM GUIDE

Concept: Talking to a Sprite

To introduce students to the concept of a sprite and the commands that cause a sprite to have color and shape, play the "Introduction to Sprites" game that appears in the "Games Sprites Play" section. The basic ideas to be introduced and some earlier concepts to be modified are as follows.

- A sprite is really an invisible character that does work.
- When a sprite has a shape and a color, you can tell where it is. Even though you can't see the sprite, you can see the colored shape it is carrying.
- If a sprite has a shape but no color or a color but no shape, you can't see it.
- Attributes are independent of each other with each requiring a specific command. For example, if the color of a sprite is changed, the shape remains the same; when the shape of a sprite is changed, the color stays the same.
- A sprite is known by its number — like players on a team.
- HOME is the location in the middle of the display.

Your students should be ready to give a sprite attributes using the TI LOGO commands.

Changing the shape (:TRUCK, :PLANE, :ROCKET, :BALL, and :BOX) and the color (:RED, :WHITE, :BLUE, :YELLOW, :BLACK, :PURPLE, :ORANGE, and :GREEN) of one sprite, plus the commands HOME and TELL SPRITE 1, and a reminder about the ERASE key, are plenty for a student's first experience at giving the computer instructions. Typing the full commands

```
TELL SPRITE 1
HOME
CARRY :TRUCK
SETCOLOR :BLUE
```

may seem like a lot of typing, but most students enjoy it. Typing the full commands help reinforce which command is associated with a shape and which is associated with a color.

A simple explanation that "dots" (:) always go in front of a name should be enough. If your students are familiar with the parts of speech, tell them that "dots" go in front of a noun. Computer words that are verbs don't need : (dots) in front of them. (TELL SPRITE 1 is an exception to the rule!)

A single sprite also can be made the listener by entering TELL *number*. The computer understands that you are talking to the sprite of that number and not the Turtle or a tile. However, typing TELL SPRITE *number* helps a student remember that he or she is talking to a sprite.

Bugs your students might encounter

- ☛ The **ERASE** key only works before **ENTER** has been pressed. Once **ENTER** is pressed, the computer does what it was told to do. If a student doesn't like what he or she has typed, tell him or her to erase it, retype it, and then press **ENTER**. If the student has played AIM or TARGET, you can remind them of this experience. This also encourages the development of proofreading skills.

Things to do

- ✓ Pick a favorite color and TELL SPRITE 1 to carry all five shapes, one at a time, in that color.
- ✓ Pick a favorite shape. Tell the sprite to change the color of the shape eight times to have all eight colors.
- ✓ To discover which is your favorite shape and color, look at all five shapes with different colors.

Early Concrete Level of Understanding



Adapting Cue Cards

Depending on your students' capabilities, the writing on the cue card can be all upper-case letters, cursive, or lower-case letters. An example of a cue card to be used for talking to a sprite can be found in *Appendix A*.

Notice that the design of the cue card for giving the computer TI LOGO commands correlates to the ones used with the preoperational level activities. Above the solid line are the commands that get the attention of a sprite. The commands below the line are the choices of attributes — in this case, shape and color. (*Note:* At this level, the colors are shown as splashes of paint, rather than circles, to avoid confusion with the ball shape.) The “splash” of the appropriate color to the left of the command helps those students who have difficulty spelling or reading.

If your students have played the AIM or TARGET activities, they are acquainted with the concepts of typing more than one letter, pressing the **SPACE BAR**, and pressing the **ENTER** key after each command. The sequence can be difficult to remember at first; for this reason, it is graphically represented on the cue card. Since students are no longer doing preprogrammed activities, **Q** for “quit” does not appear on the cue card.

If you decide to make a large classroom poster of this cue card, glue a real eraser over the word **ERASE**. Then, on the cue card, draw an eraser that resembles the one that you've used. Remind your students of the steps for erasing a character. Tell them, “To erase a character, hold down the **SHIFT** key with one hand, then press the letter **T** key” (for the TI-99/4 console), or “Hold down the **FCTN** key with one hand and press the number **3** key with the other hand” (for the TI-99/4A console). “These keys cause the cursor to back up one space and erase one character.”

One cue card at each computer is all that's needed, since the students probably won't need one very long. If your students' writing skills permit, encourage them to keep a computer journal. They can write down the information they want in the format that best suits their own needs. (See “Student Journal Writing” in the “Classroom Hints” section.)

TI LOGO CURRICULUM GUIDE

Concept: Giving the Computer Commands

At this stage of development, children are ready to give the computer direct commands of their own, using words and concepts with which they are familiar. If your students have done the activities included on the diskette and cassette tape, they understand that each letter they type appears on the display and that pressing the **ENTER** key after typing a series of letters causes the computer to do something. They should be comfortable with the concept of talking to the Turtle. They also understand the concept of talking to one sprite at a time and leaving it before talking to another sprite.



Concept: CLEARSCREEN

In the sprite mode, CLEARSCREEN clears all of the characters (typed letters and symbols) off the display. In the Turtle mode, the command clears the characters and designs and sends the Turtle HOME. Encourage your students not to use the CLEARSCREEN command excessively. Emphasize that it is valuable to be able to look at previously entered commands. It also helps young students remember how to spell the commands correctly. In the sprite mode, it is good for remembering which sprite you are talking to.

Because the CLEARSCREEN command contains many letters, you may want to introduce the short form of this command, CS. The important element is that the students recognize and understand the concept of the CLEARSCREEN command, not become frustrated by the amount of typing.

Bugs your students might encounter

- ♣ When the computer is in the Turtle mode, using the CLEARSCREEN command clears the display of *both* the type and designs. This is only a bug if your students don't want *both* the type and designs erased.
- ♣ When there are sprites on the display, entering the CLEARSCREEN command clears the display of only the type and not the sprites. To remove a sprite, all of the attributes it was given must be taken away.

TI LOGO CURRICULUM GUIDE

Concept: State of the Pen

As students start creating elaborate designs, they may want to move the Turtle without leaving a line. In the "Games the Turtle Plays" section, the examples with the "Paper Turtle" introduce the commands `PENUP`, `PENDOWN`, and `PENERASE`.

An image to convey to your students is that the Turtle is similar to a pen. When it moves, the Turtle draws a line like a pen leaves a line as someone draws with it. The command `PENUP` tells the Turtle to lift up its "pen" and then, when it is moved, to move without leaving a line. It does not need a value, just type `PENUP` and press **ENTER**. There is no obvious change that the Turtle has picked up its pen. You can only tell that the pen is up when the Turtle is moved with the `FORWARD` and `BACK` commands.

When you reposition the Turtle and are ready to draw again, type `PENDOWN` and press **ENTER**. Now, when the Turtle is moved, it draws a line. If you want to erase a portion of a design, tell the Turtle to exchange its pen for an eraser with the `PENERASE` command. This command also does not need a value. Again, you don't know that the Turtle has changed to its eraser until it is moved. If you move the Turtle over a line it has drawn before, the Turtle erases it. To stop erasing and start drawing, type `PENDOWN` and press **ENTER**. Or, to stop erasing and move without leaving a line, type `PENUP` and press **ENTER**.

It is appropriate to talk to your students about the *state of the pen* at this time. Three *states of the pen* are up, down, and erase. If you were to ask a student who was working with the Turtle, "What is the state of your Turtle's pen?", the student should reply, "It's up," or "It's down," or "It's an eraser." Later, he or she will learn that there are additional *states of the pen*, including color and `PENREVERSE`.



Concept: Changing the Color of the Turtle's Ink

The color of the ink with which the Turtle draws can be changed with the SETCOLOR command. Unless it is changed, it is always black. The Turtle's ink can be any one of the 16 colors the computer knows (see *Appendix B*). The Turtle's ink appears invisible if :CLEAR or :CYAN (the color of the background) is used with the SETCOLOR command. (*Note: Due to the resolution on some monitors and television sets, the color of the vertical and some diagonal lines that the Turtle draws is not always true. However, this does not affect the color of the horizontal lines.*)

The SETCOLOR command can be introduced to the students at any time. You might want to introduce it when a student asks about such a possibility or when students start creating designs that need color. Or, it can be saved and introduced to stimulate more interest in Turtle graphics.

Bugs your students might encounter

- ✎ SETCOLOR only affects the ink of the Turtle's pen. It does not change the color of any designs already on the display. After the ink is given a new color, the color change isn't seen until the Turtle is moved.
- ✎ If this is a student's first encounter with : (dots), the semantics of typing in SETCOLOR :color may be difficult. Patience and a cue card can help minimize this problem.

Things to do

- ✓ Think of a simple design. Tell the Turtle to draw it at several different places on the display using a different color of ink for each design.

TI LOGO CURRICULUM GUIDE

Concept: REPEAT

The REPEAT command lets you tell the computer to do the same thing or things a given number of times. REPEAT can be introduced when your students ask, "How can I make the computer do something a bunch of times and then stop?"

Using REPEAT with the Turtle makes it easier to tell the Turtle to draw specifically sized designs. It also helps your students realize how many sides there are to a specific design. For example, after your students discover the number of sides and the size of the angle in a pentagon, they can save typing time by using the REPEAT command.

REPEAT can take one command, a list of commands, one procedure, a list of procedures, or a list combining procedures and commands. The syntax for the REPEAT command (REPEAT *number* [action or actions to be repeated]) may seem difficult initially, but it's easy once the student understands the logic.

REPEAT requires two inputs — the first is the number of times something is to be repeated and the second input is what action is to be repeated. The "what" must be put in square brackets. The REPEAT command can be used in any mode and is especially effective in procedures.

Bugs your students might encounter

- ✘ Help your students avoid using a large number with the REPEAT command. This is the time to introduce recursion if they want something to happen "forever" (see the "Concept: Recursion" section). Encourage them to use manageable numbers with REPEAT to maintain control of the computer and what it is doing.

Things to do

- ✓ Make a sprite "dance" around the display by using REPEAT with FORWARD, BACK, RIGHT, and LEFT commands. The following commands are one way to accomplish this.
TELL SPRITE 1
HOME
CARRY :BALL
SETCOLOR :LEMON
REPEAT 32 [FORWARD 10 RIGHT 45]
- ✓ Play with creating geometric shapes by telling the Turtle to REPEAT a set of steps and turns. The following commands are one way to accomplish this.
TELL TURTLE
REPEAT 9 [FORWARD 20 LEFT 80]
- ✓ Discover some interesting angles, and then try to discover exactly how many REPEAT commands are needed to draw the design without tracing over any lines the Turtle has already drawn.
- ✓ After understanding how circles are made by playing the game "The Human Turtle" (see the "Games the Turtle Plays" section), use the REPEAT command to draw many different sized circles on the display.

Early Concrete Level of Understanding



Concept: Talking to More Than One Sprite

Your students are probably asking how to put more than one sprite on the display (as experienced in BUILD, DALLAS, and other preoperational level activities). Some students may have already figured out how to do it. As an introduction, play the "Introduction to More Than One Sprite" game in the "Games Sprites Play" section. The following are some important concepts to be introduced or reinforced.

- There are 32 sprites in TI LOGO, numbered from 0 to 31. Each sprite has its own number.
- HOME is always in the center of the display. If you put more than one sprite at HOME, the sprites stack on top of each other. Use FORWARD or BACK and a number of steps to move the sprites apart.
- Sprites can be stacked on top of each other. They can even be partially stacked which causes interesting effects.
- The BOX, ROCKET, BALL, and PLANE shapes are 16 sprite steps wide and high. The TRUCK shape is 16 sprite shapes wide and 10 sprite steps high. Some children will use these facts to determine the placement of the sprite carrying a shape on the display.
- When TELL SPRITE *number* is typed and entered, the sprite with that number listens to everything you say and does it (if it can). To get the attention of another sprite, enter the TELL SPRITE *number* command with a different number. The sprite with that number listens and the previous sprite stops listening.
- Unlike the sprites and shapes in the preoperational level activities, your students can now go back and talk to old sprites by typing TELL SPRITE and the number of the sprite to which they want to talk.
- Encourage the students not to clear the display (CS) frequently. The typing that remains on the display is a valuable reminder of commands and their correct spellings. It also provides information about the sprites to which you are talking.

Initially, you might elect to give your students a limited number of sprites to which to talk. If you are introducing five sprites, each one with a cue card, the activities could center around each sprite carrying a different shape. Eventually tell your students, or they may discover it on their own, that there are more than five sprites. There are 32 but they're numbered 0 through 31. After they are familiar with the concepts of more than one sprite and five shapes and eight colors, you can introduce all 16 color words (see *Appendix B*). Since some of the names of the colors may be unfamiliar to some students, demonstrate each color on the display. Then encourage the students to think of something that color. Be sure to tell the students that the color cyan is the color of the display. Any shape with that color looks like it's not there. Any shape that is colored clear allows the cyan background to be displayed through it. As a result, the shape appears to be invisible.

TI LOGO CURRICULUM GUIDE

A reproducible cue card for putting more than one sprite on the display can be found in *Appendix A*. The statement at the bottom of the cue card indicates the correct card.

Have as many different cue cards as you feel necessary. Having one card for each sprite number reinforces the need to type `TELL SPRITE number` each time a student wants to talk to another sprite. For your students who are still copying exactly from the cue card, `FORWARD number` and `BACK number` commands are suggested. Students should reach the stage when they are comfortable enough with the `FORWARD` and `BACK` commands to experiment with their own values.

Bugs your students might encounter

- ♣ If students start stacking sprites (making them appear one on top of the other), they need to know that sprites stack up with the smallest numbered sprite on top. (See the "Sprites and Shapes" chapter in the *TI LOGO User's Manual* for more details.)
- ♣ If the smallest numbered sprite in a stack of at least five sprites has no attribute of color, the rest of the sprites on the stack are not visible.
- ♣ No more than four sprites can be seen on the same horizontal line (see "Concept: No More Than Four Sprites in a Row").
- ♣ If a student gives a sprite a number that is larger than 31, the computer calculates the number of the appropriate sprite by subtracting 32 until the result is a number from 0 through 31. This capability is called *modulo* (see "Concept: Modulo"). For example, `TELL SPRITE 56` is the same as `TELL SPRITE 24`.

Things to do

- ✓ Place sprites up and down the display, each carrying a different colored shape.
- ✓ Can you make a thick line of boxes (any color or colors) up and down the middle of the display?
- ✓ Experiment with lots of different numbers for the `FORWARD` and `BACK` commands. Spread out a number of sprites, and then put them close together.
- ✓ Create designs by putting two shapes close together or on top of each other. (Make a tree using the `ROCKET` and the `BALL` shapes. A stoplight can be made by putting a green `BALL` above a yellow `BALL`, which is above a red `BALL`. A `PLANE` can be in front of a gray cloud, the `BALL`.)

Early Concrete Level of Understanding



Concept: Talking to :ALL

You can tell all of the sprites (0 - 31) to listen at the same time by typing TELL :ALL and pressing ENTER. Now all 32 sprites do the same thing at the same time.

Bugs your students might encounter

- ♣ TELL :ALL and SETCOLOR :CLEAR make all the sprites look as if they've disappeared. They are still on the display with all of the attributes they had, minus color.
- ♣ TELL :ALL, SETCOLOR :CLEAR, HOME, and FORWARD 97 move all of the invisible sprites off of the display (if the heading of the sprites is 0). Now that you have this information, try to help your students discover it for themselves.
- ♣ When talking to :ALL, remember that *all* the sprites do the same thing at the same time. Normally, they are all stacked together and look as if just one sprite is on the display (the exception would be any sprites that you've already talked to which may be separated from the group).
- ♣ There are many possible ways to make the sprites look like they have disappeared. Putting all the sprites at HOME has the potential for causing confusion later. When all of the sprites are at HOME, any sprite numbered larger than 3 is not visible. (See "Concept: Priority Sprites" and "Concept: More Than Four Sprites in a Row" for more details.)

Things to do

- ✓ Spread out several sprites, and then make them all the same color and the same shape.
- ✓ Tell several sprites to carry the same shape and have the same color. Then make one sprite different.
- ✓ Place several sprites up and down the display. Then make them invisible (:CLEAR) and then visible (:any color except the background color or :CLEAR).

TI LOGO CURRICULUM GUIDE

Concept: Dots and Quotes

Dots (:) goes in front of a word and tells the computer to figure out the value of the thing or things associated with that word. For example, PURPLE has the value of 13. When you enter SETCOLOR :PURPLE, dots tells the computer to figure out the value of the word PURPLE, and then to use that value.

Quotes (") tells the computer to use the string of characters that follow the quote. Remind your students to use two hands when typing : and ".

The game "Dots and Quotes" in the "Games the Turtle Plays" section helps explain the functions of these two punctuation marks.



Concept: SETHEADING and SETSPEED

Heading can be a difficult concept even if your students are familiar with the directions North, South, East, and West. Students can relate the RIGHT and LEFT commands to themselves, but the concept of direction is more abstract. When your students are comfortable with giving a sprite the heading of :NORTH, :SOUTH, :EAST, and :WEST, play the "Introducing SETHEADING" game in the "Games Sprites Play" section to introduce SETHEADING with numbers.

Explain that every possible direction is represented by a number from 0 through 360. When introducing numbers, encourage your students to discover that a heading of 0 is :NORTH, a heading of 270 is the same as :WEST, and so forth. Don't give your students the headings 0, 90, 180, and 270. There's plenty of motivation to encourage them to discover these for themselves. If they ask for help, ask them questions like, "Do you need a heading larger or smaller than your last one? What direction is 300? What number gives a southeast heading?" Continue asking them questions until you feel they understand the concept well enough to apply it to the Turtle or a sprite. You can also use a compass with numerical headings marked on it to introduce the concept of SETHEADING with numbers.

Some specific applications of heading are as follows.

- The Turtle and sprites always have a heading of 0 when they first appear on the display.
- Since a sprite is invisible, you never see it turn to a new heading until it is moved. Then the change can be seen. Shapes do not turn or change directions, sprites do.
- Even if you don't know which direction a sprite is currently heading, you can make sure it faces the way you want by using the SETHEADING command.
- Headings start at 0 and get larger in a clockwise rotation. Since the SETHEADING command is modulo (see "Concept: Modulo"), students can give it a number larger than 360. However, to help the students maintain control of a situation, encourage them to use the numbers 0 through 360.
- The SETHEADING command has a different effect on the Turtle and sprites than the RIGHT and LEFT commands do. Heading is a set direction, whereas RIGHT and LEFT commands are relative to the current position of the Turtle and sprites.

Two analogies for introducing SETSPEED are the cruise control on your car or telling someone how fast to drive their car. SETSPEED is a concept that's easy for students to understand and one they've probably been asking about (especially if they've played the DALLAS activity). There are a few points as it relates to the computer.

- Once the speed of a sprite has been set, that sprite keeps that same speed until it is changed. Even if you talk to another sprite, the first sprite keeps its given speed.
- The concept of "wrapping" needs to be introduced or reviewed and that terminology explained to your students. Physically wrapping a string around a box helps demonstrate the concept of what happens "behind the display." The computer, however, calculates where the sprite ends up on the other side of the display and sends the sprite there almost immediately (see "Sprites and Shapes" chapter in the *TI LOGO User's Manual* for more details).

TI LOGO CURRICULUM GUIDE

- If the heading of a sprite is straight up or down (0 or 180) or straight across (90 or 270), the sprite wraps around and around. Because the monitor is rectangular, corner to corner wrapping doesn't work evenly. To help students visualize this concept, wrap a string around a box.
- Speed can only be given a value from -127 through 127. If the speed of a sprite is set to a smaller or larger number, the computer prints the message SETSPEED DOESN'T LIKE . . . AS INPUT. The student needs to reenter the SETSPEED command using the parameters the computer understands.
- Since speed is generally considered a positive motion, giving the SETSPEED command a negative value does not mean that the sprite has the attribute of negative speed. Rather, the sprite has the reverse value of a positive speed.

Bugs your students might encounter

- 🔥 It makes a difference if the speed is set before the heading or if the heading is set before the speed. Experiment both ways. Have the students predict what happens as you experiment. Predicting helps your students develop logical sequencing skills.
- 🔥 Help your students figure out what each sprite has to do differently to separate from the stack. There is no one correct answer. They could each have a different speed or a different heading or move a different number of steps.

Things to do

- ✓ Can you discover the headings which will send a sprite to each of the four corners or the four sides?
- ✓ Make many sprites go in different directions carrying any shape you want. Give them the same speed, and then give each one a different speed.
- ✓ Can you create a scene with sprites carrying different shapes, going in different directions at different speeds, or standing still?
- ✓ Make many sprites go across the display in different directions using TELL :ALL. Then tell them all to go in the same direction at the same speed, carry the same shape, have the same color, and become invisible (remember they still have the shape, speed, and heading attributes).
- ✓ How can you make all of the sprites disappear and move off of the display?

```
ANSWER: TELL :ALL
        SETSPEED 0
        HOME
        SETCOLOR :CLEAR
        SETHEADING 0
        FORWARD 97
```

Help your students discover that SETHEADING 0, FORWARD 97, and BACK 95 move a sprite off the display if it is on the x-axis. It can be an interesting exercise that helps the students discover one of the parameters and features of TI LOGO.

Early Concrete Level of Understanding



Concept: No More Than Four Sprites in a Row

Students should know that computers have different characteristics.

A characteristic of the TI Home Computer is that no more than four sprites can be viewed on the same horizontal axis. To demonstrate this phenomenon, have your students watch as you type the following commands. Before starting this demonstration, be sure that all 32 sprites have no attributes and are not at HOME.

TELL SPRITE 1	TELL SPRITE 2	TELL SPRITE 3	TELL SPRITE 4
HOME	HOME	HOME	HOME
CARRY :BOX	CARRY :BOX	CARRY :BOX	CARRY :BOX
SETCOLOR :BLACK	SETCOLOR :RED	SETCOLOR :YELLOW	SETCOLOR :GREEN
SETHEADING 90	SETHEADING 90	SETHEADING 270	SETHEADING 270
FORWARD 40	FORWARD 20	FORWARD 20	FORWARD 40

Note: All four sprites are visible on a horizontal row. Now put Sprite 5 at HOME.

```
TELL SPRITE 5
HOME
CARRY :BOX
SETCOLOR :BLUE
```

Ask your students, "Can you see it? We've given it the attributes of shape, color, and location. Why can't you see it?"

Move it and see what happens.

```
SETHEADING 0
FORWARD 20
```

"There it is! It *really* was at HOME with all of the attributes, but the computer isn't capable of displaying it on the same row as four other sprites."

Now, enter SETSPEED 10. Sprite 5, the one carrying the blue box, goes up (because the heading is 0), wraps around, and appears at the bottom of the display. It continues moving and seems to disappear when it passes between the other sprites.

Ask your students, "Does this happen only when the four sprites are at HOME?" Move the sprites forward a number of steps and see.

```
TELL :ALL
SETHEADING 0
FORWARD 50
```

(ANSWER: No.) Now the blue box is visible at HOME but becomes temporarily invisible as it passes between the other sprites.

Remind your students that they might have encountered this situation in the PARK, PEOPLE, DALLAS, and BUILD activities.

TI LOGO CURRICULUM GUIDE

Concept: Which Four Are Visible?

How does the computer decide which four sprites are displayed? The computer uses the same priority system as it does to decide which sprite is on top of the stack. The four smallest numbered sprites are visible.

To demonstrate this, put the five sprites in the example in "Concept: No More Than Four Sprites in a Row" on the display.

Then enter the following commands.

```
TELL SPRITE 5  
SETSPEED 0  
HOME  
FORWARD 50
```

Although we can't see it, Sprite 5 is in the row.

Continue by entering the following commands.

```
TELL SPRITE 4  
FORWARD 20
```

Sprite 4 should still have a heading 0 from the commands TELL :ALL and SETHEADING 0. Now the blue box that Sprite 5 is carrying becomes visible. That's because Sprites 1, 2, 3, and 5 are the only sprites on that horizontal row. Ask your class to predict what happens if Sprite 4 is given a speed of 10. After they predict, enter the following.

```
TELL SPRITE 4  
SETSPEED 10
```

Sprite 4 moves towards the top of the display and wraps from the top to the bottom. What happens as it reaches the row of boxes? The blue box starts to disappear. The green box Sprite 4 is carrying is still visible. Ask your students why. (ANSWER: As Sprite 4 reaches the row of boxes, five sprites are already in a row and the computer has to decide which four should be visible. It chooses the four sprites with the smallest numbers — 1, 2, 3, and 4.) At the point where the Sprite 4 crosses the horizontal axis, Sprite 5 becomes temporarily invisible.

Early Concrete Level of Understanding



Concept: Even Pieces of a Sprite Count

Pieces of sprites also affect the visibility of four sprites on the same horizontal axis.

Continuing the demonstration, put Sprites 1, 2, 3, 4, and 5 across the display with no speed and a heading of 0. (Note: Sprite 5 seems to be invisible.)

```
TELL :ALL
SETSPEED 0
SETHEADING 0
BACK 50
TELL SPRITE 4
SETSPEED 0
HOME
SETHEADING 270
FORWARD 40
SETHEADING 0
```



Sprite 4



Sprite 3

Sprite 5



Sprite 2



Sprite 1

Next, enter the following commands.

```
TELL SPRITE 5
FORWARD 8
```

Sprite 5 moves forward eight steps, but only half of the box is visible.



Sprite 4



Sprite 3



Sprite 5



Sprite 2



Sprite 1

TI LOGO CURRICULUM GUIDE

Now enter the following commands.

```
TELL SPRITE 5  
HOME  
TELL SPRITE 2  
FORWARD 8
```

Only the lower half of Sprite 5's shape is visible.



Sprite 4



Sprite 3



Sprite 5



Sprite 2



Sprite 1

Make sure your students understand that the concept applies to all shapes, all sprites, and all horizontal axes. It is a concept that students need to understand.

Bugs your students might encounter

- ✘ Placing all 32 sprites at HOME with no color means that more than four sprites are on the same horizontal axis, even if they can't be seen.

Things to do

- ✓ Use the fact that no more than four sprites can be seen in a row to your advantage. With various shapes, have the sun slowly set behind an office building or a rocket blast off out of a box.

Early Concrete Level of Understanding



Concept: MAKESHAPE and CALL

Ask your class, "Have you ever wondered how the computer remembers the five different shapes?" To see how it remembers the rocket shape, type `MAKESHAPE :ROCKET` and press **ENTER**.

A 16-by-16 grid (16 rows and 16 columns) appears on the display. Note that the background turns green, signalling that the computer is learning. A large rocket shape appears on the grid. This is the pattern for the shape that a sprite carries. Press **BACK** to leave the `MAKESHAPE` mode and return to the mode the computer was in previously. Demonstrate all five shapes using the `MAKESHAPE` command.

There are 26 grids (numbered 0 through 25) available on which shapes can be made. Grids 1 through 5 are the shapes used by the computer (`:ROCKET`, `:PLANE`, `:BALL`, `:BOX`, and `:TRUCK`). These shapes can be redesigned using the arrow keys. When the computer is turned off, shapes 1 through 5 return to the original design. The other 21 grids are blank and can be used for creating new designs.

Since shapes can be identified by both a name and a number, the commands `CARRY` and `MAKESHAPE` can be followed by *:name* or by a *number*. The computer understands that `CARRY 3` and `CARRY :ROCKET` are the same. Because names are descriptive, it is easier for students to remember a shape by its name. It can be confusing to a student to say `TELL SPRITE 3` and `CARRY 3`.

Try creating your own shape. Decide what shape you'd like to make and the name of it. The `CALL` command assigns a name to a value. For example, assuming that there is no design on grid 6, type `CALL 6 "SMILE` and press **ENTER**. Now type `MAKESHAPE :SMILE` and press **ENTER**.

The display turns green and the blank grid number 6 appears. The flashing black square in the upper left-hand corner is the cursor. It serves the same function as the flashing underline () in other modes. Using the arrow keys, `↑` (E), `←` (S), `→` (D), and `↓` (X), move the cursor around the grid. To fill in a square, hold down the **SHIFT** key (for the TI-99/4 console) or the **FCTN** key (for the TI-99/4A console) and press an arrow key. The cursor moves in the direction of the arrow and fills in the square that it left. Try to make the grid look like a smiling face. When you finish designing your shape, press **BACK**.

Now the computer knows six shapes: `BALL`, `ROCKET`, `PLANE`, `BOX`, `TRUCK`, and `SMILE`. To see your shape on the display, tell any sprite to `CARRY :SMILE`. Put it at `HOME` and then give the sprite the attribute of color.

The short form for `MAKESHAPE` is `MS`. If your students don't discover this for themselves, introduce the format for figuring out the short forms of commands (see "Concept: Format for Short Forms") and help them discover the short form of the `MAKESHAPE` command.

Bugs your students might encounter

☛ Pressing **CLEAR** (**SHIFT C** on the TI-99/4 console) clears the grid. Because of C's proximity to X, the down-arrow key, shapes are sometimes inadvertently cleared. This situation does not occur on the TI-99/4A console since **FCTN 5** clears the grid. Moving the cursor across an already filled-in square without holding down the **SHIFT** or **FCTN** key erases it. At this point, discuss computer etiquette with your class. Tell them not to clear a shape someone else has designed; even if just one square is filled in, it may be a star, bullet, or something else that's essential to someone else's work.

☛ A shape can have more than one name.

`CALL 6 "SMILE`

`CALL 6 "TERRY`

If you type `MAKESHAPE :SMILE` and `MAKESHAPE :TERRY`, the same grid appears.

TI LOGO CURRICULUM GUIDE

- ☛ To blacken every square, move the flashing cursor into a square that is already filled in. Moving the cursor into a filled square does *not* cause the square to clear. The only way to clear a square is to move the cursor out of the square without holding down the **SHIFT** or **FCTN** key.
- ☛ Shapes are modulo (see "Concept: Modulo"). If you make a shape or tell a sprite to carry a shape with a number larger than 25, the computer calculates the appropriate shape number. The computer is actually making or carrying a shape with a number from 0 through 25.

Things to do

- ✓ Think of a design you'd like to make on the grid. Name it and then design it (CALL *number* "*name*, MAKESHAPE :*name*).
- ✓ Think of a name (your own for example), call a shape that name, and then design a shape on a grid that fits the name.
- ✓ After designing a shape, tell several sprites to carry the shape. Then give the sprite other attributes of color, heading, and speed.

Concrete Level of Understanding



CONCRETE LEVEL OF UNDERSTANDING

The course of a child's development is continuous — progressing gradually from one level to the next. At about the age of 7 or 8 years, children begin to enter what Piaget classified as the concrete level of development. A child at this stage can perform certain actions mentally, but is not yet able to internalize abstract concepts. The understanding of concepts of conservation — number, area, volume, mass, and weight — do not occur all at the same time. Having mastered one concept of conservation, such as numbers, the child is still not able to generalize to another area, such as weight. This lack of ability to generalize typifies the degree to which the thoughts of children between 7 and 11 years are concrete. While their mental ability is tied to specific situations and objects and cannot be applied to other situations and objects, children can profit from an environment that provides experiences and opportunities to test concepts.

The concepts of TI LOGO for this cognitive level of development are fun and easy, and can be introduced at any time after your students enter this level of development. As you introduce a new concept, give your students plenty of opportunity to experiment and discover the capabilities of the concept. While each concept is independent of the other concepts, it is important that students have a basic understanding of all prior concepts before progressing to a new one. A more complete understanding of each concept occurs as your students use it with previous and new concepts.

TI LOGO CURRICULUM GUIDE

Concept: Format for Short Forms

Most of the primitives in TI LOGO have a short form. The short form of one-word commands combines the first and last letters of the command, such as FORWARD = FD and RIGHT = RT. The short form of compound-word commands combines the first letter of each word in the compound command. Examples of these are CLEARSCREEN = CS, COLORBACKGROUND = CB, and SETCOLOR = SC. Some commands do not have a short form, for example, CARRY, TELL, and SPRITE. When your students start using short forms, help them remember what they mean by always verbalizing the long form. For example, say SETCOLOR dots BLUE for the typed version SC :BLUE. When demonstrating or introducing concepts, always type the long form of a command. This provides visual reinforcement while your saying the command provides verbal reinforcement.

Bugs your students might encounter

- ♣ Students may try to use short forms for names in TI LOGO, such as :PE for :PURPLE and :YW for :YELLOW. A primitive *name* does not have a short form. However, students can teach the computer short forms with the command CALL.
- ♣ Entering CALL 11 "YW teaches the computer that 11 has two names — "YW and "YELLOW. (*Note:* Since the new name is not a primitive, it is not in the computer's memory when the computer is turned on the next time.)



Concept: HIDE TURTLE and SHOW TURTLE

These two concepts are usually easy for students to understand. Students should be encouraged to hide the Turtle *only* when they've finished a design and *not* while they are drawing a design with the Turtle. When the Turtle is hidden using the HIDE TURTLE command, the \triangle disappears. The Turtle is still in the same location with the same heading. If you give the Turtle a FORWARD command, the Turtle moves FORWARD, drawing a line the specified number of steps (providing the *state of the pen* is PENDOWN). SHOW TURTLE is the command that makes the Turtle visible again.

The best use of the HIDE TURTLE command is when students complete a design in which they don't want the Turtle showing.

In a recursive procedure or commands with REPEAT, hiding the Turtle lets the Turtle draw the design faster. To demonstrate this, enter REPEAT 36 [FORWARD 5 RIGHT 10] with the Turtle showing. Then clear the display, hide the Turtle, and reenter REPEAT 36 [FORWARD 5 RIGHT 10]. The Turtle draws the circle much faster when the Turtle is hidden. (*Note: Although it is dramatic and exciting having the Turtle draw designs quickly, it is more valuable that your students watch the step-by-step action the Turtle uses when drawing a design. This use of HIDE TURTLE should only be introduced after extensive work has been done in the Turtle mode.*)

Bugs your students might encounter

- ☛ If the Turtle is at the edge of the drawing area (near the top, bottom, or sides), it may seem to disappear. SHOW TURTLE cannot make the Turtle visible in these cases. By using FORWARD, BACK, or HOME or setting the x- and y-coordinates of the Turtle, you can reposition the Turtle so that it is visible again.
- ☛ If four sprites (with or without the attribute of color) are on the same horizontal axis and the Turtle is moved to that axis, the Turtle disappears. The lines the Turtle draws are behind the sprites. If the sprite has color, the line can't be seen. If the sprite does not have the attribute of color, the Turtle lines can be seen through the sprite. A sprite with the attributes of color and speed makes the Turtle disappear briefly when the sprite passes in front of the Turtle. To debug any of these situations, use the following procedure to move all the sprites off the display.

```
TO OFF
TELL :ALL
SETSPEED 0
HOME
SETHEADING 0
FORWARD 97
END
```

When you finish typing the procedure, press **BACK** to return to the mode the computer was in. Then type **OFF** and press **ENTER**. To make the Turtle the listener again, reenter the TELL TURTLE command.

TI LOGO CURRICULUM GUIDE

Concept: State of the Sprite

Every sprite can have the five following attributes: shape, color, speed, location, and heading. Knowing all of the attributes is the same as knowing the *state of the sprite*. Being able to describe the *state of the sprite* is very helpful in debugging projects with sprites.

Knowing the *state of the sprite* helps you determine why a sprite can't be seen. For example, all of the following situations result in a sprite being invisible.

- A sprite carries a shape but has no color.
- A sprite has a color but carries no shape (either 0 or one of the other shapes that haven't been designed yet).
- A sprite carries a shape and has a color, but the color is the same as the background color.
- A sprite carries a shape and has a color but is located somewhere off the visible portion of the display.
- A smaller numbered sprite is on top of or on the same axis as four larger numbered sprites.

The heading of a sprite with no speed cannot be determined by looking at the shape it carries. You either have to remember what heading it has or give the sprite a speed and watch what direction it moves. The first time you talk to a sprite, the heading is 0.

You do not have to rely on your memory to know the *state of the sprite*. A sprite knows its state at all times. You can ask the computer to tell you a sprite's attributes. The computer can only tell one attribute at a time and can only determine attributes of one sprite at a time.

The PRINT command tells the computer to print the information on the display. It prints the information about an attribute of a sprite when used with commands that question the *state of the sprite*. Shapes and colors have both names and numbers assigned to them. The computer uses the numbers of the shapes and colors when describing these attributes. A sprite can only describe its location on the display according to its x- and y-coordinates.

Type PRINT HEADING and press **ENTER**. The computer prints the number of the heading of the current sprite. If you were talking to a group of sprites or :ALL, the computer prints a meaningless number. So, the first information required for discovering the *state of the sprite* is the number of the sprite. Type PRINT WHO and press **ENTER**. The number the computer prints is the number of the sprite to which you are currently talking. If it prints a number other than one between 0 and 31, you must be talking to a list of sprites. If you are talking to only one sprite, the computer can also print its SHAPE, COLOR, SPEED, XCOR, and YCOR.

Concrete Level of Understanding



Concept: COLORBACKGROUND

The background of the display is the color cyan. You can change it to any of the 15 other colors the computer knows by typing `COLORBACKGROUND :name`; for example, `COLORBACKGROUND :RUST`.

If the computer is in the sprite mode and there are sprites on the display, the sprites keep all of the attributes they had before the background color was changed. After changing the background with the `COLORBACKGROUND` command, the last sprite you talked to is still listening. You don't have to enter `TELL SPRITE number` unless you want to talk to a different sprite.

Using the `COLORBACKGROUND` command in the Turtle mode does not affect any designs on the display. You do not need to enter `TELL TURTLE` since the Turtle is still listening to all commands.

Another way to change the background color is with the `TELL BACKGROUND` command.

```
TELL BACKGROUND
SETCOLOR :name
```

The background color changes to the specified color. This command also makes the background the current "listener" of all commands. The sprite or Turtle is no longer listening to commands. To get the attention of a sprite again, enter `TELL SPRITE number`. To get the attention of the Turtle, enter `TELL TURTLE`.

Bugs your students might encounter

- ☛ If a sprite or a Turtle graphic design is the same color as the background, it appears to be invisible. Change the background color and it reappears.
- ☛ Students enjoy coloring the background black. Often it makes the sprite and Turtle designs very exciting. Since the computer type is also black, your students cannot see what they have typed. This makes it difficult to correct bugs and typing errors.
- ☛ `COLORBACKGROUND :CLEAR` looks black because the screen is black and it shows through the clear color.

Things to do

- ✓ Change the background to several different colors until you find your favorite.
- ✓ Put a number of sprites on the display with attributes of shape, color, and speed. Then, using `TELL :ALL` and `COLORBACKGROUND`, make the sprites look like they've disappeared (without using `SETCOLOR :CLEAR`).

TI LOGO CURRICULUM GUIDE

Concept: FREEZE and THAW

FREEZE and **THAW** are two fun and easy commands that should be introduced together. **FREEZE** causes *all* of the sprite motion to stop. When you type **FREEZE** and press **ENTER**, the sprites stop where they are. The sprites maintain all other attributes except speed, which they resume when given the **THAW** command. While the sprites are frozen, any attribute can be changed using the appropriate command. Your students can talk to a single sprite (**TELL SPRITE number**) or to all 32 sprites (**TELL :ALL**). The effects of changing the speed and heading are visible when the sprites are given the **THAW** command. **THAW** causes the sprites to start moving again, providing they had a speed previously. If the speed or heading is changed while the sprites are frozen, they start moving in the new direction or at the new speed. (Note: If the attributes of a list of sprites are changed after entering the **FREEZE** command, only the sprites in the list thaw when the **THAW** command is entered. If you continue making attribute changes to a sprite or a list of sprites, results that differ from the commands entered may appear on the display.)

Bugs your students might encounter

- ☛ The **FREEZE** and **THAW** commands apply to all of the sprites that appear on the display. If only one sprite is displayed, the **FREEZE** command stops that sprite. Likewise, if 32 sprites are displayed, **FREEZE** stops all 32. Use the **TELL SPRITE number**, **SETSPEED 0** commands to stop a specific sprite when many sprites are displayed.
- ☛ The **TELL SPRITE 2** and **FREEZE** commands cause all of the sprites to stop, with Sprite 2 being the listener. Although **FREEZE** affects all of the sprites on the display, any attribute changes affect only Sprite 2. To change the attributes of other sprites, you must make the appropriate sprite (or sprites) the new listener (or listeners).
- ☛ If the listener is changed while the sprites are frozen, only the new listener thaws when the **THAW** command is entered. To thaw all the sprites, enter **TELL :ALL** before the **THAW** command.

Concrete Level of Understanding



For an example, let's tell Sprite 4 to carry a red box and be at HOME. You could determine the *state of the sprite* by typing the following commands and reading the computer's answer.

```
PRINT WHO
4
PRINT SHAPE
5
PRINT COLOR
6
PRINT SPEED
0
PRINT HEADING
0
PRINT XCOR
0
PRINT YCOR
0
```

Being able to gather such information from the computer can be very helpful when writing procedures and debugging situations.

TI LOGO CURRICULUM GUIDE

Concept: Teaching the Computer

At this point, all of your students should have had several successful experiences at the computer. If so, they'll enjoy teaching the computer to do a series of commands or a *procedure*. Ask your students, "Have you made a typing mistake and had the message TELL ME HOW TO . . . appear on the display? That's because anything the computer doesn't understand, it assumes you do. The computer wants you to teach it something new." (*Note: To help demonstrate the concept of teaching the computer, play the "Teaching the Computer" game in the "Games the Turtle and Sprites Play" section.*)

The easiest introduction to teaching the computer is to have each student write down a short list of commands that tell the Turtle to create a design. Using the analogy of baking a cake in the "Teaching the Computer" game, ask each student to think of a name for what he or she wants to teach the computer.

Since many of your students have been typing a list of commands that take away the attributes of a sprite, this may be an appropriate procedure for the sprite mode. (*Note: In TI LOGO, always refer to what the computer is taught as a *procedure* and not a program.*)

Select a name for your procedure that represents what the procedure does. VANISH, CLEAN, SWEEP, or POOF are good names for a procedure that causes the sprites to become invisible. Involve your students by asking them for names, and then select one of the suggested names.

Type the selected name for the procedure and press **ENTER**. (*Note: The name VANISH was selected for this example.*) The computer gives the message TELL ME HOW TO VANISH. To tell the computer it's time to learn, use the word TO. At a computer set up for this demonstration, type TO, a space, and then VANISH. (*Note: The name must be a single word, and a space must be between TO and the name.*) Then press **ENTER**.

The screen turns green and TO VANISH appears in the upper left-hand corner of the display. END also appears. The red cursor is at the end of the title line. Press **ENTER** and the cursor moves down one line, creating a blank line for typing the first statement of the procedure.

Type the first statement and then press **ENTER**. For verbal reinforcement, say each command as it is typed. Also, say "enter" each time this key is pressed. Remember to press **ENTER** after each statement, *except the last one*.

END must be the last line in the procedure. That tells the computer that the procedure is finished. The statements in the VANISH procedure should be as follows.

```
TO VANISH
TELL :ALL
SETCOLOR :CLEAR
SETSPEED 0
SETHEADING 0
HOME
FORWARD 97
END
```

When you finish typing the procedure, press **BACK** to return to the mode the computer was in. This tells the computer that you are finished teaching it. (*Note: The cursor can be on any line in the procedure when **BACK** is pressed.*)

Concrete Level of Understanding



Now the computer knows the procedure, just like it knows HOME, SETCOLOR, and all the other primitives. Type only the name of the procedure and press **ENTER**. Because the computer has been taught the name of the procedure and the steps in it, it “runs” or does the procedure. (Note: If you type TO VANISH, the computer tries to learn the procedure again by turning the display green and listing the procedure.)

If you make a typing mistake while typing the procedure, press **ERASE** before pressing **ENTER** to backspace and erase the mistake. If the cursor is at the beginning of the line on which the mistake is located, press **CLEAR** to erase the entire line. Then retype the command correctly. The arrow keys, if used while holding down **SHIFT** (on the TI-99/4 console) or **FCTN** (on the TI-99/4A console), move the cursor, letting you make corrections or additions. (See the “Teaching Mode” chapter in the *TI LOGO User’s Manual* for further details.)

If you run a procedure and the computer doesn’t understand a line in the procedure, a message is displayed telling you where the error is. The procedure needs to be edited to correct the error. Type EDIT, a space, and the name of the procedure (for example, EDIT VANISH). Then use the arrow keys to move the cursor to the mistake. Correct the error using the appropriate keys. Then press **BACK** to tell the computer that you are finished editing the procedure. To run the procedure, type the name and press **ENTER**.

Bugs your students might encounter

- ☛ If, while writing a procedure, a space is accidentally left between quotes and the word following quotes (for example, CALL 5 “ FUN), the computer gives the message TELL ME MORE after **BACK** is pressed. This indicates that the computer did not learn the named procedure. The procedure also cannot be edited because the computer never learned it. To correct this bug, teach the computer the procedure again, omitting the space between quotes and the following word.
- ☛ If the word END has accidentally been erased and doesn’t appear as the last line of a procedure, it needs to be added. Pressing **BACK** tells the computer to stop learning. If END is omitted when the computer returns to the mode it was in, a > instead of the ? appears on the display. This means that the computer is still waiting to know when the procedure ends. Type END, then edit the procedure using the appropriate keys to erase any extra characters that may now appear in the procedure.
- ☛ If a two-word name is used for a procedure, the second name does not appear on the green display. If the second name is added to the title line while in the edit mode, the computer returns the message TELL ME MORE when the name of the procedure is entered. (Note: The computer understands the second word in the title line to be a variable. See the “Concept: Variables in a Procedure” section for more information.) Edit the procedure so that the name is only one word. Almost any character can be used for the name of a procedure. (Note: The following characters cannot be used because they are primitives — /, -, +, [,], (,), :, ., =, and *.) If you combine these symbols with letters or numbers, unpredictable messages occur when a procedure is run.
- ☛ A student might get the message TO DOESN’T LIKE . . .AS INPUT. This means that the computer already understands that word or series of letters. To correct this bug, rename the procedure.
- ☛ As more students write procedures, a situation may occur where two students select the same name. The second student can tell when this happens because when TO *name* is typed and **ENTER** is pressed, the green display appears with a procedure already on it. “Computer Etiquette” says that a student should then press **BACK** and pick another name for the procedure.

TI LOGO CURRICULUM GUIDE

Things to do

- ✓ Give your students some sample procedures to copy until they feel they can create their own.

Sample Procedures

```
TO TRAFFICLIGHT
TELL SPRITE 1
HOME
CARRY :BALL
SETCOLOR :RED
FORWARD 40
TELL SPRITE 2
HOME
CARRY :BALL
SETCOLOR :YELLOW
FORWARD 20
TELL SPRITE 3
HOME
CARRY :BALL
SETCOLOR :GREEN
END
```

```
TO TREE
TELL SPRITE 6
HOME
CARRY :ROCKET
SETCOLOR :BLACK
TELL SPRITE 7
HOME
CARRY :BALL
SETCOLOR :GREEN
FORWARD 14
END
```

```
TO ARROW
TELL TURTLE
FORWARD 40
RIGHT 120
FORWARD 20
BACK 20
RIGHT 120
FORWARD 20
END
```

- ✓ Teach the computer a procedure using the Turtle. Then run it two or three times and see what kind of design it creates.
- ✓ Write down a list of commands on paper that you think create a certain design. Then without entering each command separately to try it out, teach the computer that list of commands. Run the procedure and see how close you were to your original idea. If they weren't alike (what you thought of and what you taught the computer), which do you like better? How could you change either one?
- ✓ Write a procedure using your name as the title of the procedure.

Concrete Level of Understanding



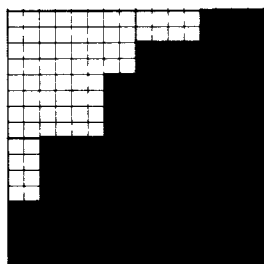
Concept: Making Big Shapes

If your students are not comfortable with teaching the computer, they can still go on to this concept and have success with it. However, they might experience frustration with long-term projects that require great amounts of typing.

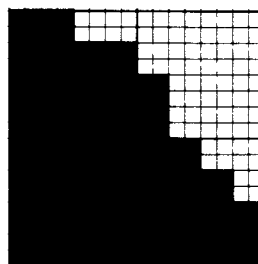
Shapes can be created that are larger than the 16-by-16 grid by putting pieces of shapes together like a puzzle. Use the examples below, or create your own, to demonstrate this concept. Remember to say each command as it is typed and **ENTER** as it is pressed.

```
CALL 6 "LEFTSIDE
CALL 7 "RIGHTSIDE
MAKESHape :LEFTSIDE
MAKESHape :RIGHTSIDE
```

Make shapes 6 and 7 look like the left and right sides of a mountain.



Shape 6



Shape 7

Now teach the computer how to make the big shape.

```
TO MOUNTAIN
TELL SPRITE 1
HOME
CARRY :RIGHTSIDE
SETCOLOR :OLIVE
SETHEADING 90
FORWARD 16
TELL SPRITE 2
HOME
CARRY :LEFTSIDE
SETCOLOR :OLIVE
END
```

Since MOUNTAIN is a procedure, press **BACK** to return to the mode the computer was in. Now, run the procedure by typing MOUNTAIN and pressing **ENTER**.

Bugs your students might encounter

- ☛ Only four sprites are visible on the same horizontal axis. You may need to introduce or remind the students about this concept (see "Concept: No More Than Four in a Row"). Big shapes can be as tall as a student wants but no more than four sprites wide.

Things to do

- ✓ Design a big shape by putting together two or more shapes that have been created with the MAKESHape and CALL commands. Examples might include a big airplane, a ship, an office building, or a snake.
- ✓ Make a multicolored big shape, for example, a blue house with a white roof or a green bottle with a red cap.

TI LOGO CURRICULUM GUIDE

Concept: Teams

Besides talking to one sprite at a time or :ALL, you can talk to a group of sprites by listing the numbers of the sprites inside square brackets. This group of sprites is called a list or team. You can give a team a name using the CALL command. The following example demonstrates two teams, one called COWBOYS and the other called BRONCOS.

```
CALL [1 2 3 4] "COWBOYS
CALL [5 6 7 8] "BRONCOS
TELL :COWBOYS
CARRY :BALL
SETCOLOR :BLUE
SETSPEED 10
TELL SPRITE 1
SETHEADING 0
TELL SPRITE 2
SETHEADING 90
TELL 3
SETHEADING 180
TELL 4
SETHEADING 270
TELL :ALL
HOME
TELL :BRONCOS
CARRY :BALL
SETCOLOR :WHITE
SETSPEED 10
TELL 5
SETHEADING 45
TELL 6
SETHEADING 135
TELL 7
SETHEADING 225
TELL 8
SETHEADING 315
TELL :BRONCOS
HOME
TELL :COWBOYS
HOME
TELL :ALL
HOME
TELL :COWBOYS
SETCOLOR :RED
HOME
TELL :BRONCOS
CARRY :BOX
HOME
TELL :ALL
HOME
```

Concrete Level of Understanding



The statement CALL [1 2 3 4] "COWBOYS assigns the name COWBOYS to the list of sprites. :COWBOYS tells the computer to figure out what list of sprites has been assigned the name COWBOYS and then to use that list.

Putting sprites in a team or list and giving the list attributes, rather than giving each sprite attributes individually, encourages logical thinking and decreases the amount of typing required. With the TRAFFICLIGHT example (in "Concept: Teaching the Computer"), the sprites used to set up the back of the light could be listed as follows.

```
CALL [4 5 6] "BACK
TELL :BACK
HOME
CARRY :BOX
SETCOLOR :BLACK
```

Then tell each sprite to move FORWARD the appropriate number of steps.

Bugs your students might encounter

- ☛ A sprite can be on more than one team. For example, Sprites 1, 2, 3, and 4 are on the COWBOYS, but they are also on the team called :ALL.
- ☛ You can talk to a team and then talk to an individual sprite in that team. A student may forget that the last sprite talked to was an individual and give commands intended for the team.

Things to do

- ✓ Get two teams of sprites moving in opposite directions, each team carrying its own shape and having its own color. (For example, you can have two squadrons of airplanes flying at each other and raindrops falling on rockets flying north.)
- ✓ Use teams as a shorter way to set up a multicolored scene. (For example, design four trees with black trunks and green tops in a row, and small boxes inside big boxes floating on the display).
- ✓ After creating a "big shape," tell it to move using the team concept.

TI LOGO CURRICULUM GUIDE

Concept: Debugging Skills

By now, some of your students might be writing longer and more complicated procedures. If you are following the sequence of concepts that appear in this guide, now is an appropriate point to talk about debugging skills.

A “bug” occurs when a procedure doesn’t do what you thought or wanted it to do. A bug is often a wonderful surprise — when it’s not doing what you like, but it ends up doing something exciting and unexpected. Debugging that kind of procedure just means analyzing each line and discovering which statement line or lines cause the surprise.

Some bugs are typing mistakes. These are easily corrected by editing the procedure, locating the typing error, and then correcting it. Students need to learn how to read the messages from the computer. For example, the message TELL ME HOW TO FDD IN LEVEL 2 LINE 3 OF BOX, tells the student to EDIT BOX, count down to the third line, and change FDD to FD. (*Note:* The title line of a procedure is not considered line 1. The first statement is line 1.) If a statement extends across the display and continues below that, the computer does not consider it a new line. If a blank line appears in a procedure, the computer counts it as a statement line.

A common typing mistake is to interchange the letter O for the number zero. In our language, we use the same shape and we can tell whether it’s a zero or an O by how it’s used. The computer can’t do that, so it uses different characters. The letter O is square looking and the number zero has rounded corners. Have students type several of each. Younger students who can’t yet understand the above explanation are usually quite satisfied with the explanation that the number zero is on the top row of keys with the other numbers and the letter O is with the letter keys.

Because zero and O are closely located on the keyboard, even “expert” typists make this mistake. Because the computer doesn’t know the wrong character has been typed, it cannot give a message explaining that. But, being aware of the situation helps locate this bug.

Reading messages from the computer is essential to developing debugging skills. Debugging is a skill that needs to be developed by everyone (especially teachers since they’re usually called on to help)!

Another common bug that occurs is when a procedure is written to do one thing and it does something else. The student needs to read over the procedure carefully and figure out what happens at each line and, if possible, locate the line or lines causing the bug. Encourage your students to think through each procedure step-by-step and calculate, “How do I tell the computer to do that?”

The following are some questions to consider when debugging a procedure.

To whom are you talking? Were you talking to one sprite when you wanted to talk to :ALL? Were you still talking to a list of sprites when you wanted to talk to one? Were you talking to the correct sprite? Were you talking to the Turtle when you wanted to be talking to a sprite?

Does your procedure assume anything? The first time you ran the procedure, was your sprite at HOME or did it have a different speed or heading? Answering this question involves discussing the *state of the sprite*. What was the *state of the sprite* before the procedure? What is the *state of the sprite* after the procedure is run? Likewise, when debugging a procedure involving the Turtle, consider the *state of the pen* and the *state of the Turtle*.

Concrete Level of Understanding



Concept: Subprocedures

A subprocedure is any procedure used within another procedure. It's still an entire procedure, but it is a part of the procedure that uses it. The idea of subprocedures is a very powerful one. They can simplify a very long procedure into easier-to-handle shorter procedures. Likewise, they can help simplify a seemingly difficult or involved project into smaller projects. You can write a procedure for each small project, and then put these subprocedures into a procedure that accomplishes the original, big project.

The easiest introduction to subprocedures is to tell students that once you finish writing a procedure, the computer knows it just like the primitives and you can use it in another procedure just like the primitives. After that, experience is the best teacher.

Bugs your students might encounter

- ☛ Sometimes when you are combining two procedures into a superprocedure, a need arises for a few intermittent steps. It's fine to combine primitives and procedures. This is a good opportunity to review the *state of the Turtle*, *state of the sprite*, and debugging skills.
- ☛ If your students are already writing recursive procedures, there are several potential bugs in combining recursion and subprocedures (see "Concept: Recursion").

Things to do

- ✓ *One-student exercise:* Write a Turtle procedure to draw a design. Then run that procedure over and over again (counting how many times you run it), until you create a new design. Now write a new procedure that draws the larger design. Two examples appear below.

```
TO CORNER  
TELL TURTLE  
FORWARD 25  
RIGHT 90  
END
```

```
TO SQUARE  
CORNER  
CORNER  
CORNER  
CORNER  
END
```

```
TO MARK  
TELL TURTLE  
FORWARD 20  
RIGHT 135  
FORWARD 10  
LEFT 45  
END
```

```
TO NEW  
MARK  
MARK  
MARK  
MARK  
END
```

- ✓ *Two-student exercise:* Let each student write a short Turtle procedure. Combine the two procedures to create a new design. Then, write a new procedure using the two procedures as subprocedures. (*Note:* These exercises may seem nonsensical at first, but they are fun and easy ways to see the power and potential of subprocedures.)
- ✓ Think of a Turtle or sprite project. Break it down into several subprojects and write a procedure for each subproject. Then write a superprocedure that includes the subprocedures.
- ✓ Think of a project that uses both the Turtle and sprites, for example, trucks moving on a road drawn by the Turtle, or a football sailing over goalposts drawn by the Turtle. Write a procedure for the Turtle part and another procedure for the sprite part. Then write a superprocedure to accomplish the big project.

TI LOGO CURRICULUM GUIDE

Concept: WAIT

The computer can operate very quickly. Sometimes it does things so fast that you can't see them. Since you are working with graphics and you want to be able to see them, it is important to be able to manipulate the rate at which the computer does some things. Where the operating process cannot be slowed down, the WAIT command can be placed between other commands, enabling you to see the effect of the first command before the computer goes on to perform the next one.

An example of this is a procedure that uses FREEZE and THAW. When typing the commands FREEZE and THAW, there is a time delay due to the time it takes to type and enter the commands. But using the commands in a procedure has a different effect. Assuming that there are several sprites on the display, enter this sample procedure.

```
TO JUMPY
FREEZE
THAW
END
```

Now type JUMPY and press **ENTER**. The sprites seem to falter for just a moment and then resume their attributes. They froze, but they thawed almost immediately. Now, edit JUMPY to include the WAIT command. Notice that the WAIT command needs an input telling the computer how long to wait.

```
TO JUMPY
FREEZE
WAIT 60
THAW
END
```

Enter JUMPY and see the effect of combining the FREEZE and WAIT commands.

The unit of time for WAIT is 1/60th of a second. Both the Turtle and sprites can be given the WAIT command. Since 1/60th of a second is a difficult fraction with which to work, simply tell the students that WAIT 60 causes the computer to wait one second. This gives them an identifiable reference.

Bugs your students might encounter

- ✘ If a procedure contains a long WAIT command, it may appear that the computer has stopped working. To check the situation, look at the cursor. If the question mark does not appear but the underline (the cursor) is flashing, the computer is still running the procedure.

Things to do

- ✓ Write a procedure that causes a sprite, a group of sprites, the background, or a group of characters to flash different colors.

Concrete Level of Understanding



Concept: BEEP and NOBEEP

BEEP causes the computer to emit a tone. The sound stops when NOBEEP is entered. (Be sure to establish classroom policies on volume control and the use of earphones or headphones.)

Putting the WAIT command between BEEP and NOBEEP controls the duration of the tone.

For example, HONK is a subprocedure of PEEP that causes a short sound to be repeated 10 times.

```
TO PEEP
REPEAT 10 [HONK]
END
```

```
TO HONK
BEEP
WAIT 15
NOBEEP
WAIT 15
END
```

Bugs your students might encounter

- ☛ If a recursive procedure is stopped and the computer is producing a continuous tone, enter the NOBEEP command to turn off the tone.

Things to do

- ✓ Write a morse code procedure with BEEP, NOBEEP, and WAIT (see the “More Fun with Sprites” chapter in the *TI LOGO User's Manual* for details).
- ✓ Have a rocket fire a missile, and beep while the missile is flying through the air. Using the WAIT command, have the missile blow up or disappear and stop beeping.
- ✓ Create a creature in the MAKESHape mode. Then give it commands to cause it to beep as it moves across the display.

TI LOGO CURRICULUM GUIDE

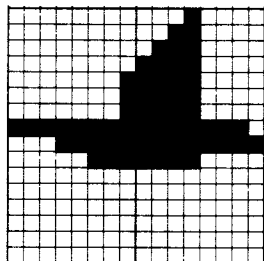
Concept: Making Movies

A movie is many still photographs viewed very quickly. Each picture is a little different from the one before, but when viewed in quick succession, the subject looks as if it is moving. Have you ever seen a book with a different picture on each page which makes a movie when you "flip" through the pages? A very simple movie can be created using one sprite and several shapes. One shape represents one page in the flip book. The sprite does the work of "flipping" through the shapes.

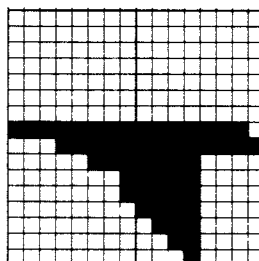
As an introduction to this concept, create a movie and let it be an example. There are two basic kinds of movies: a simple-to-do, long-running recursive movie, and a multishaped, non-recursive movie. Examples of each follow.

Recursive Movie

Think of a simple action that repeats itself over and over again, such as a person doing jumping jacks or a bird flapping its wings. Make a shape that shows one part of the action; then make a second shape that shows the other part of the action.



Shape 6



Shape 7

Combining the two procedures makes a short movie.

```
TO FLY
TELL SPRITE 0
HOME
SETCOLOR :BLUE
MOVE
END
```

```
TO MOVE
CARRY 6
WAIT 20
CARRY 7
WAIT 20
MOVE
END
```

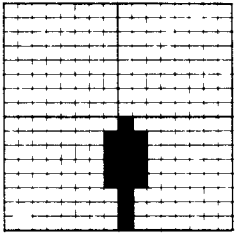
Enter FLY and watch the bird flap its wings. Press **BACK** to stop the recursive movie.

Concrete Level of Understanding

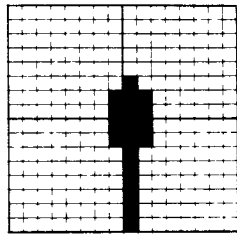


Non-recursive Movie

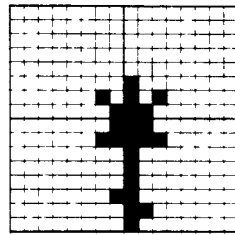
Think of a simple action that can be represented by a few different shapes, for example, a plant growing, fireworks exploding, or a dump truck dumping. With the MAKESHAPE command, make the shapes that show the action.



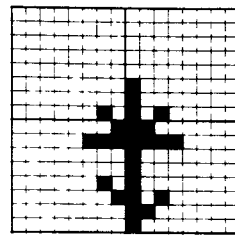
Shape 8



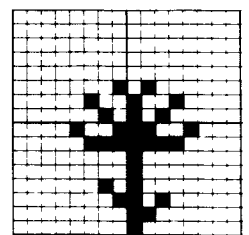
Shape 9



Shape 10



Shape 11



Shape 12

These procedures cause the action to happen. Press **BACK** to stop the PLANT procedure.

```
TO PLANT
TELL SPRITE 0
HOME
SETCOLOR :GREEN
GROW
END
```

```
TO GROW
CARRY 8
WAIT 20
CARRY 9
WAIT 20
CARRY 10
WAIT 20
CARRY 11
WAIT 20
CARRY 12
END
```

By keeping GROW in a separate procedure, it can be used to make *any* sprite or sprites show a movie of a growing plant. The next example demonstrates this capability. (*Note:* If you have already run the PLANT procedure, move Sprite 0 away from HOME so that Sprites 1, 2, 3, and 4 are visible.)

```
TO GARDEN
CALL [1 2 3 4] "PLANTS
TELL :PLANTS
HOME
SETCOLOR :GREEN
SETHEADING 90
TELL 2
FORWARD 20
TELL 3
FORWARD 40
TELL 4
FORWARD 60
TELL :PLANTS
GROW
END
```

TI LOGO CURRICULUM GUIDE

Bugs your students might encounter

- ✦ Remember the concept about priority sprites and make sure there are no “invisible” sprites preventing you from seeing the “movie.”
- ✦ It’s possible to create a movie by having one sprite carry each of the shapes. This method can become quite complicated and involved. A theatre movie only needs one projector and a TI LOGO movie only needs one sprite. When students become adept at movie making and at making big shapes, they may want to make “big movies.” Such a project involves more than one sprite but only the number of sprites needed to make the big shape.

Things to do

- ✓ Make a “moving” movie by giving a sprite a speed before showing the movie, for example, a bird flying while it flaps its wings.
- ✓ After making a movie, edit the procedure and add different colors.
- ✓ After making a recursive movie with two shapes, try making one with more than two shapes, for example, a bird flapping its wings — wings up, wings in the middle, and wings down.



Concept: Recursion

Recursion is the process of making a procedure repeat itself. If a procedure contains its own name as a statement line, then the action of the procedure is repeated. Playing the game “Recursive Procedures” in the “Games the Turtle and Sprites Play” section is an excellent approach for introducing this concept. The game demonstrates the concept that each procedure has a name and whenever the computer is told to do that procedure by name, it does it. It doesn’t matter if a user types in the name, a superprocedure has the name in it, or the procedure says its own name.

An example of a recursive procedure appears below.

```
TO FLASH  
COLORBACKGROUND :BLUE  
WAIT 10  
COLORBACKGROUND :RED  
WAIT 10  
FLASH  
END
```

This type of procedure is also considered to be in an “infinite loop” because it keeps repeating itself. To stop a recursive procedure, press **BACK**.

Recursive procedures can also be stopped with conditional statements (see “Concept: Conditionals”) and with variables (see “Concept: More with Variables”). Both of these methods involve complex concepts and can be discussed when your students ask about the possibility.

While there are many ways to tell the computer to repeat a procedure — using the REPEAT command, typing the name of a procedure more than once, writing a superprocedure that has the procedure name in it more than once — recursion only occurs when a procedure has its own name as a subprocedure.

Bugs your students might encounter

- 🐛 Some students may learn about recursion before their concepts of simple procedure writing are clear. Often, these students start making all procedures recursive. Help them learn when it is appropriate to make a procedure recursive and when it is not.
- 🐛 When students want part of a procedure to be recursive and the concept of subprocedures has not been introduced, that is the opportune time to introduce subprocedure writing. Be sure that only the part you want repeated is in the subprocedure. Then make the subprocedure recursive and the superprocedure simple. An example of this bug appears below.

```
TO DESIGN  
TELL TURTLE  
CLEARSCREEN  
FORWARD 25  
RIGHT 100  
DESIGN  
END
```

The bug in this procedure is that every time the Turtle draws one line, it clears the screen and starts over. One way to debug the procedure is to write a subprocedure that doesn’t contain the CLEARSCREEN command.

TI LOGO CURRICULUM GUIDE

```
TO DESIGN2  
TELL TURTLE  
CLEARSCREEN  
WORK  
END
```

```
TO WORK  
FORWARD 25  
RIGHT 100  
WORK  
END
```

Things to do

- ✓ Write a recursive procedure that causes several color changes to a sprite, a team of sprites, or the background.
- ✓ Write a recursive procedure for the Turtle that draws a design. Then edit the procedure to include different *states of the pens*.
- ✓ Write a recursive procedure that tells the Turtle to draw lines that wrap around the display. Then write a recursive procedure that tells the Turtle to draw a design that does not wrap around the display.



Concept: EACH

Play the game “Introducing EACH” in the “Games Sprites Play” section to help introduce this concept. When talking to a team or list of sprites, all of the sprites in that team follow commands at the same time. The EACH command talks to one sprite at a time within the team, giving the first sprite in the team the command or commands, then the next sprite, and so forth, until each sprite has done the commands one at a time. For example, if all of the sprites were moving in different directions, typing TELL :ALL EACH [SETCOLOR :CLEAR] turns them invisible one at a time. Remember, the list of things to be done by each sprite must be enclosed in square brackets.

A number of commands can be inside the square brackets. Anything a sprite can do individually, it can also do with the EACH command. Assuming that there are several sprites on the display with the attributes of shape and color, the following is an example using the EACH command.

```
TO JUMP
SETSPED 0
FORWARD 20
WAIT 10
BACK 20
END
```

```
TELL :ALL
EACH [SETCOLOR :RED JUMP]
```

The TELL :ALL command makes all 32 sprites the listeners. The JUMP procedure makes all the sprites jump — those with attributes and those without attributes. As a result, it appears as if nothing is happening while the sprites without attributes are jumping.

Things to do

- ✓ Get a team of sprites to change shape using EACH. Then change the attributes of color, speed, and heading with the EACH command.

TI LOGO CURRICULUM GUIDE

Concept: YOURNUMBER

To introduce the YOURNUMBER operation, play the "Introducing YOURNUMBER" game in the "Games Sprites Play" section.

Every sprite knows its number. You can tell it to print its number by entering PRINT YOURNUMBER. In the examples that follow, be sure Sprite 4 starts at HOME with no attributes.

```
TELL SPRITE 4  
HOME
```

Since Sprite 4 is the current listener, entering PRINT YOURNUMBER causes 4 to be displayed.

You can also tell a sprite to carry YOURNUMBER of a shape. Next enter,

```
CARRY YOURNUMBER
```

and Sprite 4 carries a ball (which is shape number 4). To see that Sprite 4 is carrying a ball, use YOURNUMBER with the SETCOLOR command.

```
SETCOLOR YOURNUMBER
```

Now Sprite 4 is carrying a blue (color number 4) ball. Give Sprite 3 attributes using the YOURNUMBER command.

```
TELL SPRITE 3  
HOME  
CARRY YOURNUMBER  
SETCOLOR YOURNUMBER
```

Now Sprite 3 carries a green rocket.

Two different sprites were given exactly the same commands and yet each one has different attributes. Using YOURNUMBER "personalizes" instructions because each sprite's number is unique.

Any command that a sprite can do that requires a number value can use YOURNUMBER for that value. You can even tell the computer to add to, subtract from, multiply by, or divide by YOURNUMBER.

Things to do

- ✓ With YOURNUMBER and SETSPEED, tell several sprites to go at different speeds.
- ✓ With YOURNUMBER and FORWARD or BACK, tell several sprites to be in different positions on the display.
- ✓ With YOURNUMBER and SETCOLOR, tell several sprites to have different colors.



Concept: EACH and YOURNUMBER Together

As an introduction to using EACH and YOURNUMBER together, play the “EACH and YOURNUMBER Together” game in the “Games Sprites Play” section. EACH lets you talk to a team, or a list of sprites, one at a time. YOURNUMBER is a way of personalizing a command so that each sprite in the list performs differently.

Using these two commands together lets you give every sprite in a list a different color, shape, speed, heading, and location. Try the example to see the effect of combining the EACH and YOURNUMBER commands.

```
CALL [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20] "TWENTY
TELL :TWENTY
HOME
CARRY :BALL
EACH [SETCOLOR YOURNUMBER]
SETSPEED 10
EACH [SETHEADING YOURNUMBER * 18]
```

Notice that the commands without EACH in front of them happen to all the sprites at the same time. Those commands with EACH and YOURNUMBER affect one sprite at a time, each doing something different.

The next example demonstrates this concept using the FORWARD command.

```
TELL :TWENTY
SETSPEED 0
HOME
SETHEADING 45
EACH [FORWARD YOURNUMBER * 16]
```

You can also subtract from, divide by, or add to the YOURNUMBER command.

Calculating the number to use with SETHEADING so that it causes the sprites to spread apart evenly depends on the number of sprites on a team. Divide the number of sprites into 360 (degrees in a circle) to determine that number. For example, EACH [SETHEADING YOURNUMBER * 11] causes all 32 sprites to disperse evenly. If only six sprites are on a team, EACH [SETHEADING YOURNUMBER * 60] causes the six sprites to disperse evenly. Or, you can talk to :ALL and spread out the 32 sprites in groups (for example, EACH [SETHEADING YOURNUMBER * 20]). Experiment with manipulating YOURNUMBER using different values.

Bugs your students might encounter

- ☛ With EACH [SETCOLOR YOURNUMBER], any sprite in a list that has the same color as the background color or is :CLEAR is not visible.
- ☛ With EACH [CARRY YOURNUMBER], any sprite with the same number as an empty shape is not visible.
- ☛ Because only four sprites can be visible on the same horizontal plane, some sprites seem to disappear until they move from HOME or the same horizontal plane with EACH [SETHEADING YOURNUMBER * *value*].

TI LOGO CURRICULUM GUIDE

Things to do

- ✓ With EACH and YOURNUMBER, tell a team of five sprites to stack one above the other, each one carrying a different shape.
- ✓ Create a circle of sprites or a diagonal line of sprites.
- ✓ Create four teams of sprites. Then give each team different attributes of shape and color.
- ✓ Have each sprite in a team go HOME one at a time. Then spread them out and tell the whole team to go HOME in unison.



Concept: Multicolored Designs with Sprites and Shapes

A shape can be only one color at a time and a sprite can carry only one shape at a time. But it is possible to create a situation that looks like one shape has more than one color.

Probably the easiest way to introduce this concept is simply to have a sample on the display and ask your class how they think it was done.

Present the following situation to your students. "You have a piece of green construction paper and a piece of red construction paper. How can you put red apples on a green tree?" There are many solutions, but three possibilities are as follows.

1. Cut out red apples and glue them on the green paper that's been made into a tree shape.
2. Cut apple-shaped holes in the green paper that's been cut into a tree shape and put red paper behind the green paper.
3. Cut apple-shaped holes in the green paper; then cut apple shapes out of the red paper and put the apples in the holes.

These three solutions are similar to some possible solutions for creating a multicolored design.

Essentially, the "color of construction paper" in the above example represents a shape and a sprite. So, the three possible solutions to creating the apple tree picture with the computer are as follows.

1. Make one shape look like several apples and another shape that of the tree. Be sure a smaller numbered sprite carries the shape of the apples.
2. Make one shape look like a tree with holes cut out (squares left blank) for apples. Then make another shape about the same size as the tree. Have the smaller numbered sprite carry the tree and a larger numbered sprite carry the red back portion which allows the red to show through the holes.
3. Make one shape look like a tree with holes cut out for apples. Make a second shape look like the apples, making sure they're in the same location on the grid as the holes in the tree. In this case, it doesn't matter what sprite carries either shape.

No one way is correct, so let your students determine their own strategy by experimenting.

This very simple concept can be used to create seemingly complex designs.

Bugs your students might encounter

- 🐛 With methods 1 and 2 for creating multicolored designs, be careful what sprite number carries a shape.
- 🐛 Be aware of the "no more than four sprites in a row" concept when creating multicolored shapes.

TI LOGO CURRICULUM GUIDE

Things to do

- ✓ Make a two-colored design such as an animal with colored eyes or a building with lights.
- ✓ Make a three-colored design such as a spotted animal with colored eyes, a traffic light, or a fruit tree with a trunk.
- ✓ Make a two-colored shape with one color that flashes, such as a winking animal or a building with lights that go off and on.
- ✓ Make a movie with two colored shapes, for example, a dog with colored eyes eating from a dish or someone swinging on a swingset.
- ✓ Make a big shape with more than one color, for example, a big spaceship with USA written down the side or a tall clown with red hair.

Concrete Level of Understanding



Concept: Variables in a Procedure

Play the game "Introducing Variables" in the "Games the Turtle and Sprites Play" section to introduce the concept of variables in a procedure.

A simple way to explain a variable is: A variable is a word that becomes a place holder for a value in a procedure. When the procedure is run, a value is given to each variable. A variable can have any name except the name of a primitive. It can contain numbers but can't be made up of all numbers. And, it must always be one word.

To include a variable in the procedure, type **TO**, space, and the name of the procedure. Then press **ENTER**. The display turns green, signalling that the computer is ready to be taught something. Press the **SPACE BAR**, type the name of the variable, and press **ENTER**. Then, continue teaching the computer the procedure.

To include a variable on a statement line, type **:** (dots) in front of the variable. This tells the computer to use the assigned value of the variable. When the procedure is run, type the name of the procedure, a space, and replace the variable with a number. Then press **ENTER**. If the name of the procedure is entered without giving the variable a value, the computer gives the message **TELL ME MORE**.

In the following example, **LENGTH** is the variable. It was added to the title line in the teaching mode.

```
TO TRIANGLE "LENGTH
FORWARD :LENGTH
RIGHT 120
FORWARD :LENGTH
RIGHT 120
FORWARD :LENGTH
END
```

In this example, **NUMBER** is the variable. **NUMBER** was added to the title line while the computer was in the teaching mode.

```
TO RACE "NUMBER
TELL SPRITE :NUMBER
CARRY :ROCKET
SETCOLOR :RED
SETSPEED 10
END
```

To run the procedures, replace the variables in **TRIANGLE** and **RACE** with a value as shown here.

```
TELL TURTLE
TRIANGLE 30
RIGHT 120
TRIANGLE 45
RIGHT 120
TRIANGLE 60
RACE 1
RACE 2
RACE 3
RACE 4
```

TI LOGO CURRICULUM GUIDE

Bugs your students might encounter

- ✘ If a space is left between : (dots) and the name of the variable (FORWARD : SIDE), the computer gives the message TELL ME HOW TO The procedure was never learned by the computer. To correct this bug, retype the procedure and eliminate the space.
- ✘ Each time a variable name is used in a procedure, it must be spelled exactly the same way as it appears in the title line. If it is inconsistent, the computer gives the message : . . . HAS NO VALUE.
- ✘ If a recursive procedure has a variable in the title line, the variable name must also be in the recursive line. The following example demonstrates this concept.

```
TO CIRCLE "SIZE  
TELL TURTLE  
FORWARD :SIZE  
RIGHT 5  
CIRCLE :SIZE  
END
```

The computer accepts a variable in a title line with quotes, dots, or nothing in front of the name. The quotes is the most logical signal to have in front since it's here that the name is assigned a value. Having nothing in front is usually the easiest to use.

Putting dots in front helps some students remember to use them in the procedure.

The computer understands all three of the following title lines.

```
TO CIRCLE "SIZE  
TO CIRCLE SIZE  
TO CIRCLE :SIZE
```

When running the procedure, if the value you're giving the computer is a word and not a number, it must have quotes in front of it.

Things to do

- ✓ Write short forms of your own for SETCOLOR, FORWARD, or any of the commands that take an input.

```
TO SETC :COLOR  
SETCOLOR :COLOR  
END
```

```
TO MOVE "DISTANCE  
FORWARD :DISTANCE  
END
```

- ✓ Write a Turtle procedure to draw a shape with a definite angle but variable sides. Then, write a Turtle procedure to draw a shape with definite side length but variable angles.



Concept: More with Variables

A procedure can have more than one variable. However, each name for a variable must appear on the title line of the procedure.

Type **TO** and the name of the procedure, and then press **ENTER**. The red cursor appears at the end of the title line. Press the **SPACE BAR**, type quotes, and then the name of the first variable. Then press the **SPACE BAR** again, type quotes, and then the name of the second variable. When the procedure is run, a value must replace each variable.

```
TO POLYGON "SIDE "ANGLE
TELL TURTLE
FORWARD :SIDE
RIGHT :ANGLE
POLYGON :SIDE :ANGLE
END
```

To run the **POLYGON** procedure, type **POLYGON 25 90** and press **ENTER**. Experiment with other numbers replacing the variables "SIDE and "ANGLE.

The computer treats variables exactly as they are entered into the computer. In this case, the first number is always the length of the side and the second is the degree of the angle.

Variables can also be manipulated by mathematical operations. You can add to, subtract from, multiply by, or divide by a numeric variable. The following example demonstrates this concept.

```
TO SPIRAL "SIDE
TELL TURTLE
FORWARD :SIDE
RIGHT 90
SPIRAL :SIDE + 5
END
```

In the preceding recursive procedure, the statement **SPIRAL :SIDE + 5** increases the assigned value of **SIDE** by 5 steps.

The following recursive procedure demonstrates the use of two variables. The "SIDE variable is manipulated by addition and "ANGLE variable is manipulated by subtraction.

```
TO STRANGE "SIDE "ANGLE
TELL TURTLE
FORWARD :SIDE
RIGHT :ANGLE
STRANGE :SIDE + 1 :ANGLE - 10
END
```

In both of the following recursive procedures containing conditionals, a variable is used to stop the procedure. To run the first example, replace the three variables with numerical values (for example, **POLY 20 45 8**). When running the second example, the "NUMBER variable needs to be replaced by a number (for example, **COUNTBACK 10**).

TI LOGO CURRICULUM GUIDE

```
TO POLY "LENGTH "ANGLE "#OFSIDES
TELL TURTLE
IF :#OFSIDES = 0 STOP
FORWARD :LENGTH
LEFT :ANGLE
POLY :LENGTH :ANGLE :#OFSIDES - 1
END
```

```
TO COUNTBACK "NUMBER
IF :NUMBER = 0 STOP
PRINT :NUMBER
COUNTBACK :NUMBER - 1
END
```

Things to do

- ✓ Write a Turtle procedure to draw a design that spirals larger and larger.

Concrete Level of Understanding



Concept: PRINTCHAR, CHARNUM, and MAKECHAR

A character is any letter, figure, or symbol used in writing or printing. In TI LOGO, the computer knows 72 designed characters. These are the letters, figures, and symbols that you see on the keyboard, plus characters used by the computer for its graphics. There are also 184 blank characters on which designs can be made.

Each character is identified by a number. A list of all of the characters and the number the computer assigns to each character appears in *Appendix C*. A character can be typed directly from the keyboard and appear on the display, or it can be placed on the display using the PRINTCHAR command.

The PRINTCHAR command must be used in conjunction with the character number. For example, PRINTCHAR 68 tells the computer to print the letter D on the next blank line of the display.

If you want to know the character number of a keyboard character, the operation CHARNUM takes the input of a character and outputs the character number associated with it. Quotes must be typed in front of the character so that the computer understands it's not a procedure. For this operation, even a digit needs to have quotes in front of it. Using the PRINT command with CHARNUM tells the computer what to do with the information and avoids the display of the message TELL ME WHAT TO DO WITH . . . For example, entering

```
PRINT CHARNUM "E
```

causes the computer to display 65.

As with the MAKESHAPE command, the MAKECHAR command also lets you teach the computer something. With this command, the computer can be taught any character shape on an 8-by-8 grid. Since MAKECHAR also identifies the character by a number, it must be used with a number.

Making a character requires the same special keys as making a shape. To make a character, type MAKECHAR *number* and press **ENTER**. An 8-by-8 square grid appears in the upper left-hand corner of the display. A black flashing cursor appears in the upper left-hand corner of the grid. If the number entered is a character design that the computer already knows (numbers 32 through 95), that character appears on the grid. Pressing **CLEAR** erases the grid. (*Note:* If a character is erased, it is not available for the rest of that session with TI LOGO unless it is redesigned. When the computer is turned on the next time, the original computer character appears.) If the number is one of the empty grids, no design appears on the grid.

The four arrow keys move the cursor in the direction indicated. Holding down the **FCTN** key on the TI-99/4A console, or the **SHIFT** key on the TI-99/4 console, and pressing an arrow key causes the cursor to fill in the square from which it moves. Moving the cursor without holding down **FCTN** or **SHIFT** moves the cursor without coloring squares. As with MAKESHAPE, pressing **BACK** returns the computer to the mode it was in before entering the MAKECHAR command.

After designing a character, you can then display that character with the PRINTCHAR command. Type PRINTCHAR, the number of the character, and press **ENTER**. The character appears on the next line.

Using the REPEAT command with PRINTCHAR lets you fill either a portion or the entire display with characters. The display is 30 characters wide and 24 characters high. This information should help you and your students with character placement.

TI LOGO CURRICULUM GUIDE

The **CLEARSCREEN** command erases all of the characters that are on the display and returns the cursor to the upper left-hand corner.

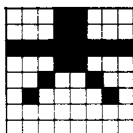
Characters and sprites can be combined in a design. If a sprite and character are at the same location, the sprite is seen and the character is underneath the sprite. It also can produce some interesting combinations of color, design, and motion. Note that characters cannot be given the attribute of speed.

Most characters cannot be printed on the part of the display used by the Turtle when the computer is in the Turtle mode. If you combine characters and Turtle graphics, most characters can appear only in the lower portion of the display where the Turtle does not draw. The Turtle graphics appear only in the Turtle's drawing area of the display. Characters 33 through 95 can be placed in the Turtle's drawing areas with the **PUTTILE** command (see "Concept: **PUTTILE**").

Character 32 is the "space" character and has no design on it. (*Note:* The display is made up of character 32s.) You can create a design on it (**MAKECHAR 32**) and see the interesting patterns that appear on the display. **PRINTCHAR 32** can be helpful in printing messages on the display. For example, it can be used to center messages, print blank lines, and center characters on the display.

The following demonstration should help you introduce **MAKECHAR** and **PRINTCHAR** to your students.

Type **MAKECHAR 160** and press **ENTER**. Then, with the arrow keys, recreate the following design.



MAKECHAR 160

When you are finished, press **BACK**. Now tell the computer to repeat the design on character 160 by entering the command **REPEAT 30 [PRINTCHAR 160]**. (*Note:* Because the display is 30 characters wide, **REPEAT 30** tells the computer to print one line of the character.)

The following example causes vertical black and blue stripes to appear on the display.

Fill in all of the squares of character 100. Now, using character 100 (which is black) and character 101 (which is blank), tell the computer to print character 100 and then print character 101 by entering the command **REPEAT 360 [PRINTCHAR 100 PRINTCHAR 101]**.



Bugs your students might encounter

- ☛ The Turtle draws its lines by making characters and putting them on the display. You can see how this happens by putting a Turtle design on the display that causes the message **OUT OF INK** to be displayed. Type **MAKECHAR 200** (or a number 2 through 31 or 96 through 255) and press **ENTER**. The line that appears on the grid is a segment of the line of the Turtle graphics. The bug results when a design is created on characters 2 through 31 or 96 through 255 with the **MAKECHAR** command and the Turtle draws on those characters. The designs erase when the Turtle draws over them. To help minimize this situation, try not to combine Turtle and character graphics. Or, redesign one of the characters on grids 33 through 95 since the Turtle does not draw on those.
- ☛ If you type two quotes when using **CHARNUM**, the computer outputs 84 which is the character number of the quotation mark. If a space is typed between the quotes and the character, the computer displays the message **TELL ME MORE**. If more than a single character is entered, the computer outputs the character number of only the first character.
- ☛ When you are typing, the computer automatically clears the next two rows of characters each time **ENTER** is pressed. All characters that are on those next two lines disappear. Thus, it is impossible to fill the entire display with characters, unless you fill the display with the space character (number 32).
- ☛ If a design has been created with the Turtle and a design is then made on one of the characters the Turtle drew on, that character design replaces the Turtle line.
- ☛ If the computer is in the Turtle mode and you are talking to the Turtle with the **TELL TURTLE** command, entering the **CLEARSCREEN** command causes the character grids 2 through 31 and 96 through 255 to be cleared. If the situation is the same but the active listener is not the Turtle, entering the **CLEARSCREEN** command does not clear the character grids.

Things to do

- ✓ Make a character with the **MAKECHAR** number command. Use **REPEAT** and **PRINTCHAR** to create a long skinny graphic or fill a specific portion of the display.
- ✓ Make two designs on two different **MAKECHAR** grids. Alternate the characters to create a checkerboard, houndstooth, check, or crazy quilt pattern.
- ✓ Change the letters (characters 33 through 96) to create your own alphabet. (See the "Tiles and Characters" chapter in the *TI LOGO User's Manual* for more information.)

TI LOGO CURRICULUM GUIDE

Concept: PUTTILE

Introduce the PUTTILE command after your students have made their own characters and printed them. This command puts a character at any location on the display.

The display is divided into 720 tiles, laid out in a series of rows and columns like a tile floor or wall. There are 30 columns (numbered 0 through 29) and 24 rows (numbered 0 through 23) of tiles. The PUTTILE command requires three values or inputs. The first input is the number of the character that is to be printed, the second number identifies the column, and the third number identifies the row.

A practice drill can include the use of a magnetic board and magnetic letters. Be sure the board has a grid drawn on it. Students can practice positioning the magnetic characters by identifying the character, column, and row.

Tiles are modulo (see "Concept: Modulo"). If a character is positioned in tile column 30, it appears on the left side of the display in column 0. Likewise, if a character is positioned in row 28, it appears at the top of the display in row 5.

Bugs your students might encounter

- ♣ Some students may have difficulty remembering the order in which the PUTTILE command accepts inputs. If a character does not appear in the position a student thought it was going to, have him or her check the order of the values.
- ♣ While there is no need to use negative numbers with the PUTTILE command, some students may choose to try it. In that case, the second and third values must be enclosed in parentheses.

Things to do

- ✓ Rewrite a PRINTCHAR procedure using the PUTTILE command instead of the PRINTCHAR command.
- ✓ Print a message in the middle of the display with the PUTTILE command.

Concrete Level of Understanding



Concept: Giving One Color to Characters

To color a set of characters, you first identify the character within the set with the TELL TILE *number* command. Then, with the SETCOLOR command, give the character one of the 16 colors the computer knows. All the characters in that group currently on the display and those that are typed after the SETCOLOR command is entered turn the indicated color.

Each group of eight characters is assigned a color as indicated on the character chart (see *Appendix C*). The pre-assigned colors can be easily changed to any one of your choice. For example, if you make character 200 look like blades of grass, REPEAT 30 [PRINTCHAR 200] creates an orange-colored lawn. Simply type TELL TILE 200 and SETCOLOR :GREEN, and press **ENTER**. The lawn becomes green.

The following chart provides information about the 256 characters available in TI LOGO. The chart indicates the character design, affects with Turtle graphics, pre-assigned color, and additional comments.

Character Number	Character(s)	Affected by Turtle Graphics	Pre-assigned Color	Comments
0	[]	no	black	Used to show MAKESHAPE and MAKECHAR grid.
1	[]	no	black	Cursor in MAKESHAPE and MAKECHAR mode.
2-10	used in master title screen	yes	black	
11-31	empty	yes	black	
32	empty "space"	no	black	Can be redesigned by user—fills screen.
33-95	keyboard characters	no	black	
96-127	empty	yes	black	
128-135	empty	yes	clear	
136-143	empty	yes	black	
144-151	empty	yes	green	
152-159	empty	yes	lime	
160-167	empty	yes	blue	
168-175	empty	yes	sky	
176-183	empty	yes	red	
184-191	empty	yes	cyan	
192-199	empty	yes	rust	
200-207	empty	yes	orange	
208-215	empty	yes	yellow	
216-223	empty	yes	lemon	
224-231	empty	yes	olive	
232-239	empty	yes	purple	
240-247	empty	yes	gray	
248-255	empty	yes	white	

TI LOGO CURRICULUM GUIDE

Bugs your students might encounter

- ☛ If you make a character in the 128 through 135 group and don't give it a color, the PRINTCHAR command types a blank space. This happens because the pre-assigned color for that group is clear. If the pre-assigned color of any character is the same as the background color, it is not visible until the color is changed. A simple solution is to assign the color of your choice to the character initially.
- ☛ When designing multicolored character graphics, characters in different sets need to be used for each color. Even if the characters look exactly alike, still select one in each group. For example, to make alternating horizontal green and yellow stripes on the display, color in the MAKECHAR 100 and MAKECHAR 108 grids. Then enter the GREENSTRIPE, YELLOWSTRIPE, and STRIPES procedures.



MAKECHAR 100



MAKECHAR 108

```
TO GREENSTRIPE
REPEAT 30 [PRINTCHAR 100]
END
```

```
TO YELLOWSTRIPE
REPEAT 30 [PRINTCHAR 108]
END
```

```
TO STRIPES
TELL TILE 100
SETCOLOR :GREEN
TELL TILE 108
SETCOLOR :YELLOW
REPEAT 12 [GREENSTRIPE YELLOWSTRIPE]
END
```

Things to do

- ✓ Change the color of a design created with PRINTCHAR and REPEAT.
- ✓ Create a multicolored design using different colored characters.
- ✓ Write a procedure to change the color of all the keyboard characters. Find a combination of background color and character color that you like best.



Concept: Giving Two Colors to Characters

If your students have used the ROAD and PAINT activities, they've experienced characters with two colors. Unlike shapes, the design can be assigned one color and the "undesigned" portion of the character a second color. For example, you can make the letters H, I, J, K, L, M, N, and O (the design) appear dark blue on white squares ("undesigned" portion) by entering the following commands.

```
TELL TILE 72  
SETCOLOR [4 15]
```

The commands are like the ones used to change the color of the character design, except for the input for SETCOLOR. To give the character two colors, use the SETCOLOR *[list]* command. The first number inside the square brackets is the number for the color of the design (the filled in squares on the MAKECHAR grid). In the above example the design is color number 4, blue. The second number inside the square brackets is the number for the color of the undesigned portion of the character (the parts not filled in on the MAKECHAR grid). Color 15 is the white part of the character.

Since characters 72 through 79 (see *Appendix C*) have been given the colors blue on white, every H, I, J, K, L, M, N, and O that is typed is a blue letter on a white square. Even though only one character has been made the active listener with the TELL TILE command, the entire group of characters listens.

All characters have a pre-assigned design color and a clear background color (see the chart in "Concept: Giving One Color to Characters"). That's why the display background color — cyan — can be seen through the character background.

Setting the color of a character to a clear design color and a black (or any of the 14 other colors) background color causes the designs to be the same color as the display background. The letters then look like stencils.

If the color of a blank character is set to be two colors and then printed, a square with no design on it appears. Only the background of the character appears on the display.

When the Turtle is the listener, you can tell it to draw lines in two colors with the SETCOLOR *[list]* command. The first number in the list is the color of the ink with which the Turtle draws. The second input is the color of the undesigned portion of the character grids. To make the Turtle draw a wide line, set both colors in the list to be the same color. (An example of this is the line with which the Turtle draws in the ROAD and PAINT activities.)

Bugs your students might encounter

- ☛ If students give the SETCOLOR *[list]* command two inputs and they're both the same number, the characters all look like little squares. The design and the undesigned portion of the character are the same color.
- ☛ All the characteristics that apply to giving a character one color still apply to giving a character two colors (see "Concept: Giving One Color to a Character"). When you are talking to one character with the TELL TILE *number* command, all the other characters in that group become active listeners.
- ☛ If the Turtle has drawn lines on the display in one color when it draws over them in a different color, the segment of the new line that intersects the old line is the same color as the old line.
- ☛ When the PENERASE command is used after the SETCOLOR *[list]* command, the Turtle only erases the thin line (the design on the character). To clear the other part of the Turtle's line, enter the CLEARSCREEN or NOTURTLE command.

TI LOGO CURRICULUM GUIDE

Things to do

- ✓ Create a two-color graphic design where one character has two colors (for example, white stars on a blue sky, red and white checkers, or black stripes on a yellow background).
- ✓ Create a background of characters (blue water) for a sprite scene (sprites carrying boat and fish shapes).
- ✓ Think of a design and try drawing it with the Turtle and then with characters and see which you like better.
- ✓ Design a Turtle graphic with the Turtle, drawing a narrow and a wide line simultaneously in two different colors.



Concept: Words and Lists

A word is a consecutive string of characters with no space between the characters. As far as the computer is concerned, a word can be just one character. A list is a group of words with a space between each word. The computer distinguishes a word from a list by the punctuation around it. A word has a quote mark in front of it (called quotes). A list is surrounded by square brackets. For example, "DOG is a word, [DOG CAT MOUSE] is a list. A list can also have just one word in it, for example, [DOG].

TI LOGO CURRICULUM GUIDE

Concept: PRINT and TYPE

The PRINT command takes the value of either a word or a list. The computer prints the characters in the word or list on the display and then *moves the cursor down to the next line.*

The TYPE command takes the value of either a word or a list. The computer types the characters in the word or list on the display and then *leaves the cursor at the end of the line.*

Try combining PRINT and TYPE with words and lists. Here are some sample procedures with which you and your students can experiment. After teaching the computer a procedure, and exiting the teaching mode, type the name of the procedure and press **ENTER**.

```
TO GREET
PRINT [HOWDY! HOW ARE YOU?]
END
```

```
TO FINE
PRINT [I'M SO GLAD.]
END
```

```
TO LOUSY
PRINT [I'M SO SORRY.]
END
```

```
TO HOWDY "FRIEND
PRINT "HOWDY!
PRINT :FRIEND
END
```

```
TO INFO
PRINT [YOU ARE TALKING TO SPRITE]
PRINT WHO
PRINT [IT IS CARRYING SHAPE]
PRINT SHAPE
PRINT [IT HAS COLOR]
PRINT COLOR
PRINT [IT HAS A SPEED OF]
PRINT SPEED
PRINT [IT HAS A HEADING OF]
PRINT HEADING
PRINT [ITS X-COORDINATE IS]
PRINT XCOR
PRINT [ITS Y-COORDINATE IS]
PRINT YCOR
END
```

The TYPE command does not leave a space after the character, word, or list. The cursor is in the next position after typing the character, word, or list. If, for example, a sprite and the Turtle are on the display and you want to know who the listener is, enter

```
TYPE [YOU ARE TALKING TO ] PRINT WHO
```

and the computer returns

```
YOU ARE TALKING TOTURTLE
```

You need to tell the computer to print a space between the output of the TYPE command and the following command. Character 32 is the "space" character. It's an empty character that is typed every time the **SPACE BAR** is pressed. (See "Concept: Tiles" for more information.) The command to print the character is PRINTCHAR and then the number of the character. Because you want a blank character, the command is PRINTCHAR 32.

Try the above example again, but this time put PRINTCHAR 32 between the TYPE and PRINT commands.



Bugs your students might encounter

- ♣ The computer can print 126 characters on a line in the teaching or edit and command modes. (Note: 126 characters equal four rows of type plus six characters on the fifth row.) Using the ENTER key to create a new line helps minimize excessively long lines.
- ♣ If you are writing a long PRINT or TYPE statement, it is tedious to calculate exactly where the statement breaks when it is displayed. For example, the line PRINT [TODAY IS THE TOMORROW YOU WORRIED ABOUT YESTERDAY.] breaks between two words in the list. But when ENTER is pressed and the computer prints the message, the break occurs in the middle of a word.
- ♣ The computer types or prints 30 characters on one line on the display. You can calculate where the break occurs in a long statement and place a hyphen at that point. Or, you can make the single, long statement into a number of smaller statements.

Continuing the example, type the following.

```
PRINT [TODAY IS THE TOMORROW YOU] PRINT [WORRIED ABOUT YESTERDAY.]
```

Now press **ENTER** and two lines of print appear on the display.

TI LOGO CURRICULUM GUIDE

Concept: READLINE

The operation READLINE tells the computer to wait for a value to be entered. When the READLINE command is entered, the prompt character > and the flashing cursor are displayed. (Note: This is the same prompt character that appears when END is omitted from a procedure.) Anything can be typed and entered in response to the prompt. The computer prints the message TELL ME WHAT TO DO WITH [. . .], with the information that was entered placed in the list. You need to give the READLINE command a name or tell the computer what to do with the information with the PRINT, TYPE, CALL, and MAKE commands. The following procedure demonstrates READLINE and CALL.

```
TO GREETINGS
PRINT [HOWDY! WHAT'S YOUR NAME?]
CALL READLINE "ANSWER
TYPE [THAT'S A NICE NAME,]
PRINTCHAR 32
PRINT :ANSWER
END
```

The READLINE command is often used in conditional situations. (See the "Concept: Conditionals" section for more information concerning the use of READLINE with conditionals.)

Bugs your students might encounter

- ✘ READLINE puts the information in a list. If you want to compare the information given in READLINE, the information you are comparing it to must be in square brackets.

Things to do

- ✓ Create a question and answer game with the CALL, READLINE, and PRINT commands.
- ✓ Tell the computer to "personalize" the responses it displays to a question and answer game by including the name of the player on each statement line.



Concept: Conditionals

Conditional statements tell the computer to test a certain situation to determine whether it is true or false and then direct the course of the test. Remind your class of daily conditional situations they encounter. For example: if it's raining, you'll go to the movies; if it's not, you'll go swimming. The condition you are testing is the weather — is it raining? You do one of two actions, depending on the results of the condition.

The computer can test many situations — do two sprites have the same x-coordinate, is one number greater than another, was the thing typed by you the information needed, is something a word, and so forth. Then, depending on whether the test is true or false, you can have the computer do whatever you want within the capabilities of the language.

There are two methods of conditional statements in TI LOGO. You can use TEST...IFT...IFF (test, if true, if false) or IF...THEN...ELSE. The major difference between the two methods is that the first one lists each part of the statement on a separate line, and the second method lists all the parts of the statement on the same line.

Both conditional statements have three parts. The first part tests the condition and relays the information of whether it is true or false. This part needs an operation that outputs TRUE or FALSE. If the input is omitted, the computer gives the message TELL ME MORE. If an input other than an operation is given, the computer gives the message ...DIDN'T OUTPUT. If the input is an operation but the information is not true or false, the message ...WAS GIVEN INSTEAD OF "TRUE OR "FALSE is displayed. (For more information concerning operations, see "Concept: Commands, Operations, and Variables.")

The other two parts of a conditional statement are signals to the computer. They require an input that is a command. IFT in TEST...IFT...IFF and THEN in IF...THEN...ELSE signal the computer to do the input command if the condition is true. ELSE and IFF signal the computer what to do if the condition is false. It is not necessary to have a false clause or a true clause as part of a conditional statement.

For example, to see if the active listener has a color of purple, you can write a procedure that would test the color. The two procedures below demonstrate both methods for testing the color.

TEST...IFT...IFF

```
TO VIOLET?  
TEST COLOR = :PURPLE  
IFT PRINT [YES]  
IFF PRINT [NO]  
END
```

IF...THEN...ELSE

```
TO PURPLE?  
IF COLOR = :PURPLE THEN PRINT [YES] ELSE PRINT [NO]  
END
```

The amount of information that needs to be included in the conditional statement is the determining factor in whether to use IF...THEN...ELSE or TEST...IFT...IFF.

Two more examples appear below demonstrating the use of both methods. The GAME procedure allows you to "drive" a sprite around the display by pressing the arrow keys. It checks if an arrow key (E, S, D, X) is pressed. If so, it changes the heading of the sprite to that direction. (Note: There is no ELSE clause because the situation does not require it.)

TI LOGO CURRICULUM GUIDE

The variable "DIRECTION must be entered after the computer is in the teaching mode. When you finish teaching the computer GAME, press **BACK**. To play GAME, be sure a sprite with the attributes of shape, color, and location is on the display. Then type GAME and press **ENTER**. To stop GAME, press **BACK**.

```
TO GAME
CALL READLINE "DIRECTION
IF :DIRECTION = [E] THEN SETHEADING 0
IF :DIRECTION = [S] THEN SETHEADING 270
IF :DIRECTION = [D] THEN SETHEADING 90
IF :DIRECTION = [X] THEN SETHEADING 180
GAME
END
```

The CONVERSATION procedure has longer commands because of the length of the messages to be printed so the TEST...IFT...IFF method is used.

```
TO CONVERSATION
PRINT [HOW ARE YOU?]
CALL READLINE "RESPONSE
TEST :RESPONSE = [FINE]
IFT PRINT [I'M SO GLAD]
IFF PRINT [I'M SO SORRY TO HEAR THAT.]
END
```

Enter CONVERSATION and answer the question.

It would be cumbersome to rewrite either of the above examples using the other conditional clause.

When you are introducing conditionals to students, the TEST...IFT...IFF method helps students break the overall conditional statement into its component parts. This type of thinking makes it easier to debug a conditional situation. However, the first experience for some students might be a simple IF...THEN statement without an ELSE clause.

The two methods of testing a condition can be used interchangeably in the same procedure. However, the parts cannot be combined into one statement.

There can be more than one IFT or IFF clause in the TEST...IFT...IFF conditional statement. If no action is to result when the condition is false, omit the IFF clause. If no action to result when the condition is true, omit the IFT clause.

Combining conditional statements with recursion can cause situations that may require strong debugging skills.

There are situations when you want a nonrecursive procedure with conditionals, a recursive procedure only under specific conditions, or a recursive procedure of which the conditionals are a part. The CONVERSATION procedure is an example of a situation that should not be recursive.

Concrete Level of Understanding



The following procedure is an example of using recursion under specific circumstances (see "Concept: READLINE" for more information).

```
TO MATHGAME
PRINT [WHAT IS 7 * 6?]
TEST READLINE = [42]
IFT PRINT [GOOD]
IFF PRINT [TRY AGAIN]
IFF MATHGAME
END
```

An example of a recursive procedure, of which the conditionals are a part, is the GAME procedure. The statement IF :DIRECTION = "Q STOP can be added to GAME, resulting in a conditional situation that stops the recursive procedure. (Note: A statement like this is included in all the preoperational activities on the diskette and cassette tape.)

You can also stop a recursive procedure with a variable. The next procedure demonstrates this capability. Remember to enter the variable ("TIMES) after the computer is in the teaching mode.

```
TO CHATTER "TIMES
IF :TIMES = 0 STOP
PRINT [CHATTER]
CHATTER :TIMES - 1
END
```

Recursive procedures can be stopped in a variety of ways. Including a conditional statement in a procedure is one method. Decide what condition has to be met and give the computer the STOP command when the condition is true.

Things to do

- ✓ Make a simple game where specific key presses control the speed, color, or direction of a sprite or the Turtle.
- ✓ Write a "conversation" procedure where certain words or phrases cause different messages to be displayed.
- ✓ Write a procedure that questions facts about sports, trivia, or things related to school events.

TI LOGO CURRICULUM GUIDE

Concept: READCHAR

The READCHAR operation tells the computer to recognize that a key on the keyboard has been typed ("read" a character) and then output that character. With READCHAR you can write a procedure that directs the course the computer takes when a certain key is typed. READCHAR is an operation, so when used by itself the message TELL ME WHAT TO DO WITH . . . appears on the display. It is generally used with the CALL command and a conditional. CALL gives a name to the key pressed and the conditional situation tells the computer what to do when the key is pressed.

To see an example of the READCHAR operation, load the LINE activity (PREOP/1 file) into the computer. This activity, and all the activities on the diskette and cassette tape, contain a recursive subprocedure (in this case, LINE1) with a CALL READCHAR "X statement, followed by one or more statements establishing the conditional (IF :X = "some letter such as IF :X = "B SETCOLOR :BLUE).

Unlike the READLINE operation which must be followed by pressing the **ENTER** key, READCHAR reads only one character and doesn't require the pressing of the **ENTER** key. Also, READLINE allows what you're typing to appear on the display and to erase any letters before pressing **ENTER**. However, with READCHAR the letter does not appear on the display. The first letter typed, even if it's not the letter you wanted, is the letter read by the computer.

If a procedure needs more than one character read at a time, READLINE is the operation that tells the computer to do this. If, for example, the statement IF :X = "NE SETHEADING 45 is a line in a procedure, when NE is typed and **ENTER** is pressed, nothing happens. That's because the computer took the first letter typed, N, and called it "X. Since the procedure doesn't have a conditional statement for the value of N, the procedure ignores the keypress and waits for one that corresponds to a conditional statement in the procedure.

Note: The READCHAR operation accepts one character, and the READLINE operation accepts any number of characters. If READLINE is used with a single character, the character that appears in the conditional statement must be in brackets. The following example demonstrates this format.

```
CALL READLINE "X
IF :X = [F] FORWARD 10
```

Bugs your students might encounter

- ☛ Be sure recursion is used properly with the READCHAR operation. The bugs that occur with READLINE also have the potential of occurring with the READCHAR operation (see "Concept: Conditionals").
- ☛ If you are going to have more than one conditional statement using the character from READCHAR (as in the LINE activity), give the READCHAR a name to avoid the bugs mentioned in READLINE.



Things to do

- ✓ Write a procedure for a sprite that uses one character inputs to control the movement of the sprite, the speed of the sprite, the direction of the sprite, the shape of the sprite, or the color of the sprite.
- ✓ Write a procedure for the Turtle which changes the direction the Turtle takes with a single keypress.

Example: TO CRAWL
 TELL TURTLE
 CALL READCHAR "X
 IF :X = "F FORWARD 5
 IF :X = "B BACK 5
 IF :X = "R RIGHT 5
 IF :X = "L LEFT 5
 CRAWL
 END

TI LOGO CURRICULUM GUIDE

Concept: More with SHAPE, SPEED, COLOR, and HEADING

Since the computer stores the shape, color, speed, and heading attributes using the appropriate number and not the word, games can be played with the computer by manipulating the number. The following procedures demonstrate how this can be done. Be sure at least one sprite has the attributes of shape and color before running the following procedures. If only a few shapes have been made, SHOWSHAPES makes the sprite seem invisible as it carries all the empty shapes. When it returns to shape 1, the five shapes the computer knows reappear in succession. The SHOWSHAPES, RAINBOW, and DANCE procedures are also good demonstrations of the modulo concept (see "Concept: Modulo"). Since the SETSPEED command is not modulo, the FASTER procedure stops when the sprite reaches the speed of 128. The message SETSPEED DOESN'T LIKE 128 AS INPUT AT LEVEL . . . LINE 1 OF FASTER appears on the display.

```
TO SHOWSHAPES
CARRY SHAPE + 1
WAIT 20
SHOWSHAPES
END
```

```
TO DANCE
SETHEADING HEADING + 5
SETSPEED 10
WAIT 30
DANCE
END
```

```
TO RAINBOW
SETCOLOR COLOR + 1
WAIT 10
RAINBOW
END
```

```
TO FASTER
SETSPEED SPEED + 1
FASTER
END
```

After teaching the computer the procedures, type the name of the procedure you want to run and then press **ENTER**. Since all four procedures are recursive (FASTER is recursive until the sprite reaches the speed of 128), you must press **BACK** to stop the procedure.

Concrete Level of Understanding



Concept: X- and Y-Coordinates

Any point on the display can be described by two numbers — its x- and y-coordinates. The x-coordinate is a point's horizontal location and the y-coordinate is its vertical location. Zero is the number associated with the middle coordinate in each direction. (The point HOME, in the very middle of the display, has an x-coordinate of 0 and a y-coordinate of 0.) The coordinates increase by 1 in one direction and decrease by 1 in the other direction. Using the commands SX (set x-coordinate), SY (set y-coordinate), and SXY (set x- and y-coordinates), you can position the Turtle and a sprite anywhere on the display.

Introducing x- and y-coordinates depends upon your students' age and previous exposure to coordinates. Students who have played the game BATTLESHIP or who have had exposure to graphing, only need to be reminded of those experiences and then given the appropriate TI LOGO commands. For students who have little or no experience in this area, explain the basic concept of a grid using the commands PRINT XCOR and PRINT YCOR.

Position the Turtle or a stationary visible sprite on the display. (Note: If a sprite has speed, the coordinates change. The computer calculates the coordinates the sprite was at when the command was entered.) Type PRINT XCOR and press **ENTER**. The computer prints the x-coordinate of the active listener — either a sprite or the Turtle. Then type PRINT YCOR and press **ENTER**. The computer prints the y-coordinate of the active listener.

To continue the explanation, strategically position several sprites on the display to help the students discover the format for the four quadrants. For example, place one sprite at HOME, and one at each of the following coordinates: 21 (-50), -28 60, -40 (-55), and 76 (-37). Now tell your students to use PRINT XCOR and PRINT YCOR to find the x- and y-coordinates. Remind them that they must have the attention of the sprite (TELL SPRITE *number*) in order to get the correct coordinates.

When your students understand that a sprite and the Turtle are at a point defined by two numbers, they are ready to use the SX and SY commands. The SX command gives the Turtle or sprite its x-coordinate and the SY command gives it the y-coordinate. Have your students select a location on the display to place a sprite. Then tell them to type SX, the appropriate number, and press **ENTER**. Then type SY and the appropriate number and press **ENTER**. Setting each coordinate separately reinforces that all x-coordinates are horizontal and all y-coordinates are vertical.

The command SXY takes two values. The first is the x-coordinate and the second is the y-coordinate. A negative y-coordinate must be put in parentheses: SXY 10 (-10), SXY -27 (-42).

Bugs your students might encounter

- ☛ If the parentheses are omitted around the negative Y input of the SXY command, the computer subtracts the second number from the first and considers the new number the x-coordinate. It also gives the message TELL ME MORE. This message is requesting the y-coordinate.
- ☛ Parentheses are not necessary around a negative x-coordinate. If they are typed, the value remains unchanged.

TI LOGO CURRICULUM GUIDE

Things to do

- ✓ Put a sprite with the attributes of shape and color on the display. Locate the x- and y-coordinates of that sprite using the PRINT XCOR and PRINT YCOR commands. Then place another sprite, carrying a smaller shape, in the same place using SX and SY. (An example is a box with a rocket on top of it at coordinates 56 101.)
- ✓ Write a procedure to draw a small Turtle design. Using PENUP, SX, SY, PENDOWN, and the procedure, move the Turtle to some location and draw the design. Then move the Turtle to a new location and draw the design again. Continue moving the Turtle and drawing the design.

Concrete Level of Understanding



Concept: Modulo

Modulo is a formal mathematical term defining a mathematical operation that yields the remainder function of division. An example of this is $39 \text{ modulo } 6 = 3$.

In the TI LOGO language, color assignments are an example of modulo. There are 16 colors with the numbers 0 through 15. If you enter SETCOLOR 26, the computer calculates what 26 is equivalent to (26 modulo 16 is 10). To explain this concept to students, tell them the computer uses one of the colors it knows. It picks that color by taking your number and continues to subtract 16 until the remainder is a number from 0 to 15. Then the computer takes that number and uses the corresponding color. A chart, like the one below, might further simplify the concept.

<i>Color</i>	<i>Number</i>	<i>Modulo Number</i>	<i>Modulo Number</i>
:CLEAR	0	16	32
:BLACK	1	17	33
:GREEN	2	18	34
:LIME	3	19	35
:BLUE	4	20	36
:SKY	5	21	37
:RED	6	22	38
:CYAN	7	23	39
:RUST	8	24	40
:ORANGE	9	25	41
:YELLOW	10	26	42
:LEMON	11	27	43
:OLIVE	12	28	44
:PURPLE	13	29	45
:GRAY	14	30	46
:WHITE	15	31	47

Let them see the pattern and make a similar chart in their journals. MAKESHAPE and the SETHEADING command also use the modulo system. For example, MAKESHAPE 28 is actually shape 2, and SETHEADING 361 is the same as SETHEADING 1.

Students can calculate what color a certain number gives them by dividing the number by 16 (the number of colors the computer knows). The remainder is the number of the color that the computer uses. Likewise, to calculate sprites, divide by 32 (the number of sprites) and the remainder is the sprite number. To calculate headings, divide by 360 and use the remainder; and to calculate shape number, divide by 26 and use the remainder.

The rationale for having these attributes be modulo is consistent with the philosophy of creating a positive learning environment. If students use numbers within the given range for each command, they'll never encounter the modulo capability. But for students who use numbers outside the range, whether or not they are randomly selected, the frustration of having a project stopped by a message from the computer saying SETCOLOR DOESN'T LIKE 34 AS INPUT, or something similar, is eliminated. It's part of the success factor built into TI LOGO. As much as possible, a student should be able to type something that causes something to happen. However, it should not be something that is random. There is a logical reason for everything that happens and that gives the student a successful feeling, knowing that she or he has command of the machine. If the computer randomly selected attributes, the computer would be in command.

TI LOGO CURRICULUM GUIDE

Negative numbers can also be given as values for the TELL SPRITE, MAKESHAPE, SETHEADING, and SETCOLOR commands. For example, TELL SPRITE - 10 is the same as TELL SPRITE 22. To help your students calculate the real number, draw a chart showing the positive numbers that correspond to the negative numbers. The following is an example of a chart for the SETCOLOR command.

<i>Color</i>	<i>Positive Modulo Number</i>		<i>Negative Modulo Number</i>	
:CLEAR	0	16	- 16	- 32
:BLACK	1	17	- 15	- 31
:GREEN	2	18	- 14	- 30
:LIME	3	19	- 13	- 29
:BLUE	4	20	- 12	- 28
:SKY	5	21	- 11	- 27
:RED	6	22	- 10	- 26
:CYAN	7	23	- 9	- 25
:RUST	8	24	- 8	- 24
:ORANGE	9	25	- 7	- 23
:YELLOW	10	26	- 6	- 22
:LEMON	11	27	- 5	- 21
:OLIVE	12	28	- 4	- 20
:PURPLE	13	29	- 3	- 19
:GRAY	14	30	- 2	- 18
:WHITE	15	31	- 1	- 17

Note: The SETCOLOR [*list*] command cannot accept negative numbers.



Concept: RANDOM

Entering the RANDOM command outputs a number from 0 to 9. If you enter RANDOM, the computer prints TELL ME WHAT TO DO WITH and a number.

You can write procedures and let the computer pick a number for you.

```
TO PAINT
TELL SPRITE 1
HOME
CARRY :ROCKET
SETCOLOR RANDOM
END
```

(Note: This procedure never uses colors 10, 11, 12, 13, 14, or 15 because the RANDOM command only generates a number from 0 through 9.)

The RANDOM command can be manipulated by a mathematical operation. To generate a random number larger than -1 and less than 100, use the following procedure.

```
TO HUNDREDS
PRINT RANDOM * 10 + RANDOM
END
```

This procedure generates a random tens digit and a random ones digit, adds them together, and gives a number from 0 through 99.

If you want a number within a certain range, you can write procedures to do that, too.

```
TO 5TO9
CALL RANDOM "TRY
IF :TRY > 4 THEN PRINT :TRY ELSE 5TO9
END
```

When your students ask how to generate a random number in a certain range, help them discover the process for themselves. There are many ways to do the same task, so let the students discover their way.

TI LOGO CURRICULUM GUIDE

Concept: PENREVERSE

Since this *state of the Turtle's pen* appears magical, it is probably better to introduce it after your students have a clear concept of the other *states of the pen*.

PENREVERSE puts the Turtle in a state where, if the Turtle is moved over a line it has already drawn, it erases that line. However, if the Turtle is moved over an area of the display where there is no line, it draws a line.

The following is a quick demonstration of this *state of the pen*.

```
TO DOODLE
TELL TURTLE
PENREVERSE
REPEAT 40 [FORWARD 25 RIGHT 90 WAIT 20]
END
```

Note: The WAIT command is included so that you and your students have time to see what the Turtle is doing. You can make the WAIT time longer if needed or omit it.

In the example, the Turtle draws a box because it was moving over a part of the display with no lines on it. The fifth time the Turtle moves FORWARD 25 it erases the first line of the box because it is moving over a line it drew previously. After eight FORWARD 25 RIGHT 90 commands, there are no lines on the display. The Turtle draws them and then penreverses them. This series of events repeats three more times.

PENREVERSE can turn a simple procedure into a very impressive, fun-to-watch procedure.

Bugs your students might encounter

- ☛ If the *state of the pen* is PENREVERSE and the Turtle never crosses any lines it has drawn, the *state of the pen* appears to be PENDOWN.
- ☛ If the Turtle's ink has been given two colors (for example, SETCOLOR [4 15]), PENREVERSE only affects the thin drawing line of the Turtle. The wide line, or the tile on which the character is printed, cannot be erased. It can be cleared from the display by entering the CLEARSCREEN or NOTURTLE command.
- ☛ The Turtle must be moved exactly over a line for the Turtle to erase it.

Things to do

- ✓ Write a recursive procedure that tells the Turtle to draw a geometric design, turn, and draw the design over again. Make the Turtle's *state of the pen* PENREVERSE and run the procedure again.

Concrete Level of Understanding



Concept: FIRST, LAST, BUTFIRST, and BUTLAST

FIRST is an operation that needs an input of a word, a list, or a number. If the input is a word, the command outputs the first character of the word. If the input is a list, the command outputs the first word in the list. If the input is a number, FIRST outputs the first digit in the number, however, the number must have " in front of it.

Introduce this concept by providing a situation and asking for predictions. For example, ask your students, "What is the first part of the word dog?" (ANSWER: D.) "What is the first part of the sentence WHAT'S FOR LUNCH?" (ANSWER: WHAT'S.) "What's the first part of the number 247?" (ANSWER: 2.)

Now demonstrate with the following examples how the computer does this.

```
PRINT FIRST "DOG
PRINT FIRST [WHAT'S FOR LUNCH]
PRINT FIRST "247
```

Once the students understand FIRST, the LAST operation is usually obvious. The BUTFIRST and BUTLAST operations can also usually be figured out by the students.

Repeat the exercise using the same examples that you used with the FIRST command. Ask your students, "What is everything but the last part of the word DOG?" (ANSWER: DO.) "What is everything but the last part of the sentence WHAT'S FOR LUNCH?" (ANSWER: WHAT'S FOR.) "What is everything but the last part of the number 247?" (ANSWER: 24.)

You can use combinations of FIRST, BUTFIRST, LAST, and BUTLAST. Have fun with your class as they predict results of combining two operations. (Note: The FIRST and BUTLAST operations and the LAST and BUTFIRST operations are generally not used together since the effect is minimal.)

```
PRINT FIRST BUTFIRST "FOUR [ANSWER: O.]
PRINT LAST BUTLAST "MUNCH [ANSWER: C.]
PRINT FIRST BUTFIRST BUTFIRST "PARTY [ANSWER: R.]
PRINT FIRST BUTFIRST [I LIKE YOU] [ANSWER: LIKE.]
```

These commands are useful in taking apart lists, words, and numbers.

The procedures below are a few examples of how to use the FIRST, LAST, BUTFIRST, and BUTLAST operations.

```
TO VERTICAL "LIST
IF :LIST = [] STOP
PRINT FIRST :LIST
VERTICAL BUTFIRST :LIST
END
```

For example, enter VERTICAL [MY NAME IS SUE] and the computer prints the items in the list in a vertical row.

TI LOGO CURRICULUM GUIDE

This procedure types a list backwards.

```
TO BACKWARDS "LIST
TYPE LAST :LIST
PC 32 ; THIS PUTS A BLANK SPACE BETWEEN WORDS
IF :LIST = [] STOP
BACKWARDS BUTLAST :LIST
END
```

Example: BACKWARDS [HOW NOW BROWN COW ?]

The following procedure tests to see if the item is a member of the given list.

```
TO MEMBER "ITEM "LIST
IF :LIST = [] OUTPUT [THE LIST IS EMPTY]
IF :ITEM = 1 OUTPUT FIRST :LIST
OUTPUT MEMBER :ITEM - 1 BUTFIRST :LIST
END
```

Example: PRINT MEMBER 4 [A B C D E F]

(Note: If the message YOUR LIST IS EMPTY is displayed and you are sure that the list is not empty, the number 0 may have been given as an item number. If so, the procedure continued subtracting one from the list until the list was empty.)

Concrete Level of Understanding



Concept: Commands, Operations, and Variables

All primitives (things understood by the computer when TI LOGO is selected from the master title screen) and things you've taught the computer can be divided into three categories: commands, operations, and variables.

There are two types of variables: local and global (see "Concept: Global and Local Variables"). All global variables, those that are primitives and those that have been created with the CALL and MAKE commands, can be seen by typing PN (Print Names) and pressing **ENTER**.

A command is a primitive or a procedure. Some commands do not require an input (for example, BEEP, FREEZE, HOME, and PENUP). Some commands take an input of a word, a number, or a list (for example, CARRY, EDIT, FORWARD, and PRINT). And some commands require more than one input (for example, REPEAT, SXY, and SENTENCE).

Operations are primitives or procedures that give information. They usually evaluate or calculate something, and then "output" the "answer." For example, $7 + 5$ adds seven to five and then outputs 12, and XCOR evaluates the x-coordinate of the Turtle or the current sprite and then outputs the x-coordinate.

If operations are used by themselves, the computer usually gives the message TELL ME WHAT TO DO WITH . . . Operations need a command to tell the computer what to do with the information. There are four categories of operations. Below is a chart of the categories and the commands that can be used with them. Note that RANDOM is in a category by itself. It can be used with any command that takes a number as input.

The following lists operations that evaluate whether a situation is true or false and then output "TRUE or "FALSE.

BOTH	NUMBER?
EITHER	RC?
IS	=
LESS	THING?
GREATER	WORD?
NOT	

The following lists commands that accept "TRUE or "FALSE as input.

PRINT	TYPE
TEST	BOTH
IF...THEN...ELSE	EITHER

The following lists operations that name a graphic object.

TURTLE	BACKGROUND
SPRITE	TILE

Note: TELL is generally the only command used with the above operations.

The following lists operations that output information about the status of the Turtle, the background, a character, a tile, or a sprite.

CHARNUM	WHERE
COLOR	WHO
HEADING	XCOR
NUMBEROF	YCOR
YOURNUMBER	XVEL
SHAPE	YVEL
SPEED	

TI LOGO CURRICULUM GUIDE

The following lists commands to be used with the previous operations.

CALL	SVEL
CARRY	SX
MAKE	SXV
PRINT	SXY
SETCOLOR	SY
SETHEADING	SYV
SETSPEED	TYPE

The following lists operations that manipulate the input and/or output of the computer.

BUTFIRST	RANDOM
BUTLAST	READCHAR
CHARNUM	READLINE
CONTENTS	SENTENCE
DIFFERENCE	SUM
FIRST	TEXT
FPUT	THING
JOY	WORD
LAST	+
LPUT	-
PRODUCT	*
QUOTIENT	/

The following lists commands that take the above operations as input.

CALL	OUTPUT
EITHER	PRINT
IF...THEN...ELSE	TEST
MAKE	TYPE

The following lists TI LOGO primitive commands.

BACK	LEFT	RIGHT
BEEP	LOOKLIKE	RUN
BYE	MAKE	SAVE
CALL	MAKECHAR	SETCOLOR
CARRY	MAKESHAPE	SETHEADING
CLEARSCREEN	NOBEEP	SETSPEED
COLORBACKGROUND	NOT	SHOWTURTLE
CONTINUE*	NOTURTLE	STOP +
DEFINE	OUTPUT	SV
DOT	PENDOWN	SX
EACH	PENERASE	SXV
EDIT	PENREVERSE	SXY
END +	PENUP	SY
ERASE	PA	SYV
FORWARD	PN	TELL
FREEZE	PO	TEST
GO +	PP	THAW
HIDETURTLE	PRINT	TO
HOME	PRINTCHAR	TRACEBACK*
IF...THEN...ELSE	PUTTILE	TYPE
IFF	RECALL	WAIT
IFT	REPEAT	

+ only used within a procedure

* PAUSE must be entered before these will work



Concept: OUTPUT

Before introducing OUTPUT to your students, familiarize yourself with the differences between commands, operations, and variables (see "Concept: Commands, Operations, and Variables"). An operation "gives," "returns," or "outputs" information. To write your own operations, OUTPUT has to be in the procedure. Any procedure that contains the OUTPUT command is an operation.

Operations are a way of having one primitive procedure calculate or manipulate something, and then pass on the information to another primitive or procedure. For example, procedures can be written to perform algebraic equations, create new number systems, and take words apart and put them back together.

If the OUTPUT command seems complicated, experiment with it before introducing it to your students. See the sample procedures to help you get started.

```
TO FORTYTIMES "NUM
OUTPUT 4 * :NUM
END
```

```
PRINT FORTYTIMES 7
```

```
TO MEMBER? "ITEM "LIST
IF :LIST = [ ] OUTPUT "FALSE
IF :ITEM = FIRST :LIST OUTPUT "TRUE
OUTPUT MEMBER? :ITEM BUTFIRST :LIST
END
```

```
PRINT MEMBER? "DOG [CAT RAT DOG]
PRINT MEMBER? "DOG [MOUSE CAT BIRD]
```

The following procedure renames the Turtle mouse.

```
TO MOUSE
OUTPUT TURTLE
END
```

Now enter TELL MOUSE.

This procedure generates a large random number.

```
TO BIGNUM
OUTPUT RANDOM*100 + RANDOM*10
END
PRINT BIGNUM
```

TI LOGO CURRICULUM GUIDE

Concept: Global and Local Variables

The two types of variables in TI LOGO are global and local. Global variables are those that are available after the variable is defined. Local variables are only available within a procedure where the variable is defined.

A global variable is created when the commands `CALL` or `MAKE` are used. These commands give a value to a name. The value can be used both inside and outside of a procedure. Entering `PN` (Print Names) tells the computer to print all the global variables that have been created during a session with TI LOGO. The names and values that are innate to the TI LOGO language are also printed.

A local variable must be part of a procedure. To generate a local variable, type the name of the procedure and press **ENTER**. Next, press the **SPACE BAR** and type the name of the variable. In the example `TO SQUARE SIDE`, `side` is the variable. A local variable only has value relative to the procedure of which it's a part. Once the variable is given a numeric value and the procedure is run, the variable no longer exists. You can verify this by telling the computer to print the value of the variable. In this example if you enter `PRINT :SIDE`, the computer displays the message `:SIDE HAS NO VALUE`.

The following procedures demonstrate the use of global and local variables that accomplish the same purpose.

Global variable that stops a recursive procedure

```
TO STOPCIRCLE1
CALL 0 "TOTALANGLE
CIRCLE1
END
```

```
TO CIRCLE1
TELL TURTLE
IF :TOTALANGLE = 360 STOP
FORWARD 10
RIGHT 10
CALL :TOTALANGLE + 10 "TOTALANGLE
CIRCLE1
END

PRINT :TOTALANGLE
```

Local variable that stops a recursive procedure

```
TO STOPCIRCLE2
CIRCLE2 0
END
```

```
TO CIRCLE2 "ANGLE
TELL TURTLE
IF :ANGLE = 360 STOP
FORWARD 10
RIGHT 10
CIRCLE2 :ANGLE + 10
END

PRINT: ANGLE
```

Bugs your students might encounter

- ☛ If a global variable is used within a procedure, be sure to set its starting value *before* the procedure is run. The message `TELL ME MORE` is displayed if the variable is not reassigned when the procedure is run again.



Concept: Velocity

Velocity affects both the rate and direction of movement of an object. In TI LOGO, the commands `SXV`, `SYV`, and `SV` set the velocity of a sprite. When using velocity, it is important to remember that one number affects both the direction and speed of motion.

Consider the sprite at the point 0 0 on a graph.

Assuming that the sprite is stationary, entering `SXV 20` moves the sprite in the direction of the x-axis at a speed of 20. Likewise, if the sprite were stationary and you entered `SYV 20`, the sprite would move along the y-axis at a rate of 20.

Values for velocity can range from -127 to 127, the same as the range available for the `SETSPEED` command. If the velocity is a positive number, the direction of motion is in a positive direction along the axis (to the right on the x-axis and up on the y-axis). If the velocity is a negative number, the direction of motion is in a negative direction.

It is possible with the command `SV` to set both the x- and y-velocities at the same time. The command takes two values; the first is the x-velocity, and the second is the y-velocity. However, this should be introduced at a later point when the students are familiar with `SXV` and `SYV`.

A sprite with a greater x-velocity than y-velocity moves in a direction somewhere in between the two axes but more toward the direction of the greater force (x).

Introduce velocity by playing the game "Introducing Velocity" in the "Games Sprites Play" section. Continue the introduction by putting a sprite in motion with `SETSPEED` and `SETHEADING` on the display. Then check the x- and y-velocity of that sprite with `PRINT XVEL` and `PRINT YVEL`. Encourage your students to change the speed or heading and then use the `PRINT XVEL` and `PRINT YVEL` commands again.

TI LOGO CURRICULUM GUIDE

Unlike XCOR and YCOR where the coordinates are stationary and independent, the SV, SXV, and SYV commands make a sprite move in a direction and at a speed. The commands are also dependent on current velocities. Letting students set one velocity and then the other and predicting where the sprites will go is an excellent way for them to internalize the concept.

Bugs your students might encounter

- ✦ Changing either the x- or y-velocity replaces the old velocity with the new one. (Remind your students that the SETSPEED command also has this characteristic. If a sprite has a speed of 20 and is given a new speed of SETSPEED 75, it stops going 20 and starts going 75.) However, with velocity, it's sometimes "counterintuitive" — not what you expect — to see the sprite change both speed and heading. Remember that velocity has two components that work together.
- ✦ When using the SV command with a negative y-velocity, the y-velocity must be in parentheses.

Things to do

- ✓ Move a sprite on the display using SETSPEED and SETHEADING. Then make another sprite duplicate the motion using the SXV and SYV commands.
- ✓ Set a number of sprites moving in different directions using SXV and SYV.
- ✓ Set four sprites moving in opposite directions using SXV and SYV.
- ✓ Set the x-velocity on several sprites to the same number. Then using the SYV command, make them move in different directions.



Concept: GO and Labels

With the TI LOGO language, you can repeat a command or a group of commands by several different methods. Some of the possibilities include the use of the REPEAT command, recursion, and subprocedures. Another method is to use the GO command with labels.

A label is a way of identifying a statement line in a procedure. Normally, the computer runs each line of a procedure in the order in which it appears in the procedure or in chronological order. With the GO command and labels, you redirect the order in which a procedure is run.

A label can be any name, number, letter, or combination of the three. It can be any number of characters up to 126.

The spacing and punctuation for a label are exact. A label name cannot contain a blank space. Dots (:) must be typed directly after the label name, without a space between the label name and dots. A space is typed between the dots and the first word in the label line. The GO command must be followed by a space, quotes ("), and then the label name. The label name must be spelled the same in the GO statement and the label statement.

In more complex, more involved, or longer procedures, the use of GO and labels can hamper the debugging process. Because of the construction of a procedure containing GO and labels, it is difficult to follow the logic of the procedure. For this reason, it is advisable to teach your students to use one of the other methods for redirecting the action of a procedure.

Students who have had experience with other computer languages may want to use GO and labels. Encourage them to experiment with recursion and subprocedures as alternate methods of accomplishing the same results.

The following is a simple example of GO and labels where KAREN is the label name. The same example using a subprocedure appears below.

GO and label

```
TO SHOW
PRINT [THIS IS THE FIRST LINE]
KAREN: PRINT [THESE ARE ALL THE REST OF THE LINES]
PRINT [THIS LINE IS IN BETWEEN]
GO "KAREN
END
```

Subprocedure

```
TO SHOWS
PRINT [THIS IS THE FIRST LINE]
SUBPROCEDURE
END

TO SUBPROCEDURE
PRINT [THESE ARE ALL THE REST
OF THE LINES]
PRINT [THIS LINE IS IN BETWEEN]
SUBPROCEDURE
END
```

TI LOGO CURRICULUM GUIDE



TI LOGO SIMULATION GAMES

Each of the games in this section covers an individual concept. With the instructions for each game is a suggested method for physically involving your students in discovering the concept. Games are not the only way to introduce new concepts, but they should help stimulate your creative thinking about ways to help students understand computer ideas.

The games are designed so that few props or materials are needed. You don't even need the computer. As you create games of your own or adapt the ones given here, remember to involve the students and give them analogies they can remember and relate to when they need to use a concept.

Each game is self-contained and can be as short or as long as your class necessitates. Included with some of the games are suggested adaptations. But with any game, feel free to adapt it to the needs of your class.

TI LOGO CURRICULUM GUIDE

Games the Turtle Plays

Introducing the Turtle Mode

Have a volunteer “student Turtle” stand in front of the group (or if you’re introducing this concept to one student, let the student be the Turtle). Then give the students the following explanation: “There’s a Turtle in the computer that can draw designs for us. We give it commands to move around the display. The Turtle, who has a pen, draws as it moves. Before we start talking to the Turtle, pretend you are the Turtle. I’ll talk to you the same way you can talk to the computer Turtle later.”

The first point to be made is that, as a Turtle, you can only understand four things: FORWARD, BACK, RIGHT, and LEFT. Tell the student to pretend that he or she can only understand those four things.

The next point is that FORWARD, BACK, RIGHT, and LEFT need numbers which tell the Turtle how many steps or turns to take. Give your student Turtle the command: FORWARD. Most students start walking forward. Some stop after a little bit; others stop when they run into a wall or some obstacle. You can say, “Great. Now you’re being a smart Turtle — you figured out for yourself when to stop. But the computer Turtle is so dumb, we have to tell it exactly how many steps FORWARD to go. Now pretend you’re a dumb computer Turtle. If I tell you to go FORWARD and don’t say how many steps to take, you just stand there.” Then give FORWARD and BACK commands, sometimes with a number of steps, sometimes without. Some student Turtles may not make the association that BACK also needs an exact number of steps.

Help your students make the same discovery for the RIGHT and LEFT commands. Show them a big RIGHT turn (bigger than a 90° turn) and a very tiny RIGHT turn. Ask them how they can tell the Turtle how big a turn to make. They should come up with the answer “use a number.” Explain that Turtle turns are very tiny turns, but they’re always the same size. The Turtle knows how much to turn because it is told how many Turtle turns to take.

Point out that when the Turtle turns, it doesn’t move. It stands in the same place and just changes direction. (*Note:* For some students, the concept of pivoting — turning while standing in one spot — is difficult to understand. Continue playing the game to help them internalize the concept.)

After giving the students Turtle directions, let the students take turns giving each other directions. Or, you pretend you are the Turtle and let your students give you directions.



The Human Turtle

This game should be used frequently when explaining Turtle graphics. Whenever a procedural bug occurs or any question comes up about the Turtle, don't tell your student the answer. Instead ask, "Why don't you figure it out by pretending you are the Turtle?"

A question or problem may be as simple as "Is that right or left?" If so, first make sure your student knows his or her own right and left. Then have the student lean his or her body as far as necessary so that he or she is looking at the display and heading in the same general direction as the Turtle. Then ask, "Do you want the Turtle to go right or left?" To help the child relate the concept of direction to his or her own body, you may need to emphasize the concept with a gentle nudge to the child's shoulder or arm that corresponds to the Turtle's direction.

This technique of repositioning a student's body to match the Turtle is easy for students to understand. Both the Turtle and the student's body are on the same vertical plane. When you do Human Turtle floor exercises, however, you're working on a horizontal plane, and some children need to have this phenomenon explained.

When a student asks a more complex question, like "How do you draw a circle?," ask the student, "How do you walk in a circle?" Have a student stand up and walk in a circle. Then ask him or her to watch carefully how he or she did it. Encourage the student to describe it in terms of steps and turns. He or she may walk in many circles before finally breaking that complex circle down into the simplest of components: a step, a turn of the body, another step of the same size, another turn of the body in the same direction, a step, a turn, a step, and so forth. Now a circle is broken down into terms the computer Turtle can understand: FORWARD or BACK a number of steps, RIGHT or LEFT a number of turns, and then repeating these steps until the circle is complete.

TI LOGO CURRICULUM GUIDE

Debugging with the Turtle

If a student encounters a bug in a Turtle procedure, you can help him or her debug it by suggesting that the student act out the procedure while you call out the steps. If, when acting out the debugging procedure, the student does what he or she wants to do instead of following the specific commands you give, suggest that the student call out the steps of the procedure and you be the Turtle. Point out what you're doing at each step and the commands that cause it.

If students are having problems relating to the Turtle's right and left directionality, set up a Human Turtle game to help them. You or a student can act as the Human Turtle, and the student with the directionality bug can be the one giving the commands.

As an additional activity, set up a game situation in which the student has to tell the Human Turtle to go around an obstacle. Tell the student to give the commands that move the Turtle around the obstacle. Have the student face the Turtle so that his or her right and left are opposite the Turtle's right and left. If this is too frustrating, let the student stand behind the Turtle and guide him or her through various commands. Then ask him or her to face the Turtle and do it again.



The Paper Turtle

Materials ■ A 3-inch square of cardboard cut on the diagonal to create a paper Turtle.

The following is an activity for making designs with the paper Turtle.

Draw a design on paper. Punch a small hole in the center of the paper Turtle and put the point of a pencil through the hole. Retrace the design. Keep the “nose” of your Turtle always in front. Watch the right and left turns it makes as you move your pencil.

Use the paper Turtle to introduce two *states of the pen* — PENUP and PENDOWN.

Demonstrating with the paper Turtle, ask your students, “If I draw a star on this side of the display and want to draw another star on the other side, how do I get the Turtle to move from one place to another?” Hopefully, somebody will say, “You pick up your pencil.” Encourage the student to be even more specific and to say, “Then move it over to where you want to draw the new star, put your pencil down on the surface, and draw the star.”

The Turtle can do the same thing, except that it uses a pen instead of a pencil. Ask the students, “What would you tell the paper Turtle if you wanted it to move without leaving a line?” Somebody might say, “Pick your pen up.” Your response can be, “Yes, PENUP is the computer command for the Turtle. Now move the Turtle to its new location using FORWARD, BACK, RIGHT, and LEFT. Next, what would you say to the Turtle when you want it to start drawing lines again?” PENDOWN is the command.

TI LOGO CURRICULUM GUIDE

Games Sprites Play

Introducing the Sprite Mode

- Materials
- 40 cutouts — 8 planes, 8 trucks, 8 rockets, 8 balls, 8 boxes cut out of construction paper; one shape in each of the eight colors — red, yellow, blue, white, green, orange, purple, and black.
 - An enlarged copy of a cue card the students use (see *Appendix A*).
 - A designated area of a room that is visible to all students and has defined borders (this is the display).

Say to your class or group, "A sprite is an invisible character that does work for you. You can't see the sprite, but when it starts to do work, you can see the work it's doing. There are 32 sprites in the computer. Each one has a number, like a player on a team, so you can tell one sprite from another. At first, we're going to talk to one sprite. The words we use to talk to a sprite are on the cue card." Ask the group what they would type. (ANSWER: TELL SPRITE 1 and then press **ENTER**.) (Note: This command is different from anything your students have typed with the preoperational activities. A space is typed after TELL and SPRITE to complete the command.) After typing the command, say, "Now Sprite 1 is listening. All sprites 'live' below the bottom of the display where you can't see them (even if they're doing work). HOME is a place in the middle of the display. Type HOME, and press **ENTER**. Sprite 1 goes there."

Hold up two cutouts of the same color, for example, a red box and a red truck. Ask, "How can we tell these two apart?" (ANSWER: They each have a different shape.) Next hold up two cutouts of the same shape but different colors, for example, a red truck and a blue truck. Now ask, "How can we tell these two apart?" (ANSWER: They each have a different color.)

Explain to your group, "Everything that we're able to see has color and shape. That's true of the things we see on the computer, too. We can tell a sprite to carry a shape and have a color. We still won't see the sprite, but we'll see the colored shape it is carrying."

Point to the five shapes on the cue card and say, "These are the five shapes the computer already knows. We can have a sprite carry one of them." Let a student pick a shape. Then, reading from the cue card, say the command that corresponds to that shape. Explain the use of dots (see "CONCEPT: Dots and Quotes"). Repeat this process for a student picking a color and then reading the command from the cue card that goes with it.

Before going to the computer, let a student be a sprite. Place the student sprite outside the designated perimeters of the "classroom screen." Next say, "Tell Sprite 1." The student sprite should look at you and maybe cup his or her hand over an ear to indicate that he or she is listening. Then say, "HOME" and the student should move to the center of the designated area for the screen.

TI LOGO Simulation Games



Let a student pick a shape and give the command that goes with that shape. The student sprite holds out his or her arms as if carrying an imaginary shape. Next say, "There's a shape there. Do you see it?" When they say no, ask why not. (ANSWER: The sprite doesn't have a color.) Let another student select a color by giving the appropriate command. Hand the student sprite the correct colored shape for him or her to hold.

"Now you can see the shape! What if you want a different shape?" Let a student pick a new shape by giving the correct command. Before giving the student sprite the effects of the command, ask the class to predict what happens. Then give the student sprite the new shape. The color of the sprite stays the same; only the shape changes. Now ask your students, "What would you do to change the color?" As a student gives the appropriate command, let the class predict what happens. Then give the student sprite the appropriate cutout.

Continue this activity until everyone has a turn giving a command or all students are comfortable with the concepts.

The following is some sample dialogue for this game.

<i>Command</i>	<i>Action By Student Sprite</i>
TELL SPRITE 1	looks attentive
HOME	moves to the center of the designated display
CARRY :TRUCK	holds out empty arms
SETCOLOR :PURPLE	holds purple truck cutout
CARRY :ROCKET	holds purple rocket cutout
SETCOLOR :RED	holds red rocket cutout
SETCOLOR :GREEN	holds green rocket cutout
CARRY :PLANE	holds green plane cutout

TI LOGO CURRICULUM GUIDE

Introducing More Than One Sprite

- Materials
- 40 cutouts — 8 planes, 8 trucks, 8 rockets, 8 balls, 8 boxes cut out of construction paper; one shape in each of the eight colors — red, yellow, blue, white, green, orange, purple, and black.
 - An enlarged copy of a cue card the students use (see *Appendix A*).
 - A classroom chalkboard or designated area of a room that is visible to all students and has defined borders (this is the display).

After your students are comfortable talking to one sprite, introduce the concept of talking to more than one sprite. Determining when to advance to talking to more than one sprite depends on the rate your students understand the concepts.

You need three student sprites and the colored paper cutouts used in the “Introducing the Sprite Mode” game. With the classroom board representing the computer display, mark the center of the board HOME. (HOME should be at a level that all students can reach and see.)

Discuss with your class the ability of talking to more than one sprite and giving it the same attributes that were given to Sprite 1. Ask your students, “How can you tell one sprite from another?” (ANSWER: Each has a different number.) Now tell your students, “When you finish talking to one sprite, you can talk to another by typing TELL SPRITE *number* and pressing **ENTER**. The number is the number of the different sprite. Since you are no longer talking to Sprite 1, it stops listening. You can no longer change any of its attributes. The new listener is the new sprite number. You can give it any attribute you want.”

Position the classroom-sized cue card so that all students can see it. After assigning a student to be Sprite 1, give that student the following commands.

```
TELL SPRITE 1  
HOME  
CARRY :BALL  
SETCOLOR :YELLOW
```

Student Sprite 1 holds the yellow ball at HOME on the blackboard.

Now ask, “How do you think we can talk to another sprite?” (ANSWER: TELL SPRITE 2.) Assign a student to be Sprite 2, and give him or her the following command.

```
TELL SPRITE 2
```

Now tell your students, “If we send Sprite 2 HOME, it will be on top of Sprite 1.” Assuming that your students have already given the FORWARD and BACK commands to the Turtle, ask your students, “What commands do you know from talking to the Turtle that move a sprite around the display?” (ANSWER: FORWARD and BACK.) Continue by giving Sprite 2 the following commands.

```
HOME  
FORWARD 20  
CARRY :TRUCK  
SETCOLOR :ORANGE
```

Sprite 2 holds an orange truck above Sprite 1, the yellow ball. Remind your students that “Sprites always start out facing the top of the display. FORWARD moves the sprite up and BACK moves the sprite down.”



Now show your students the cue card for talking to more than one sprite. Tell them, "The only difference between the cue card for talking to one sprite and the cue card for talking to more than one sprite is the number after TELL SPRITE. It changes each time you want to talk to a different sprite."

Ask the class, "What do you think is a good way to put a sprite carrying the green rocket below the yellow ball?"

```
ANSWER: TELL SPRITE 3
        HOME
        BACK 20
        CARRY :ROCKET
        SETCOLOR :GREEN
```

Student Sprite 3 positions the green rocket on the board.

Now ask them, "How could we tell Sprite 2 to stop carrying an orange truck and start carrying a blue box?"

```
ANSWER: TELL SPRITE 2
        CARRY :BOX
        SETCOLOR :BLUE
```

Student Sprite 2 now holds the blue box in the correct position on the board.

Tell your students, "Notice that we didn't tell Sprite 2 to go HOME and then move FORWARD. It already had the attribute of location. The only attributes we changed were shape and color. But now, let's move Sprite 2. How can we do that?" (ANSWER: Give Sprite 2 a FORWARD or BACK command.)

Remind the class, "The sprites whose attributes you want to change *must* be the listener."

Note: It's a good practice to send sprites HOME and then move them FORWARD and BACK. The 32 sprites are positioned just below the display in an area of the screen where they are not visible, even with the attributes of shape and color. From that location, FORWARD is up, but BACK moves a sprite to the top of the display and then down. This aspect initially may confuse some students.

Continue changing listeners and attributes of the sprites. You may want to increase the number of students involved in the game by changing student sprites or increasing the number of student sprites.

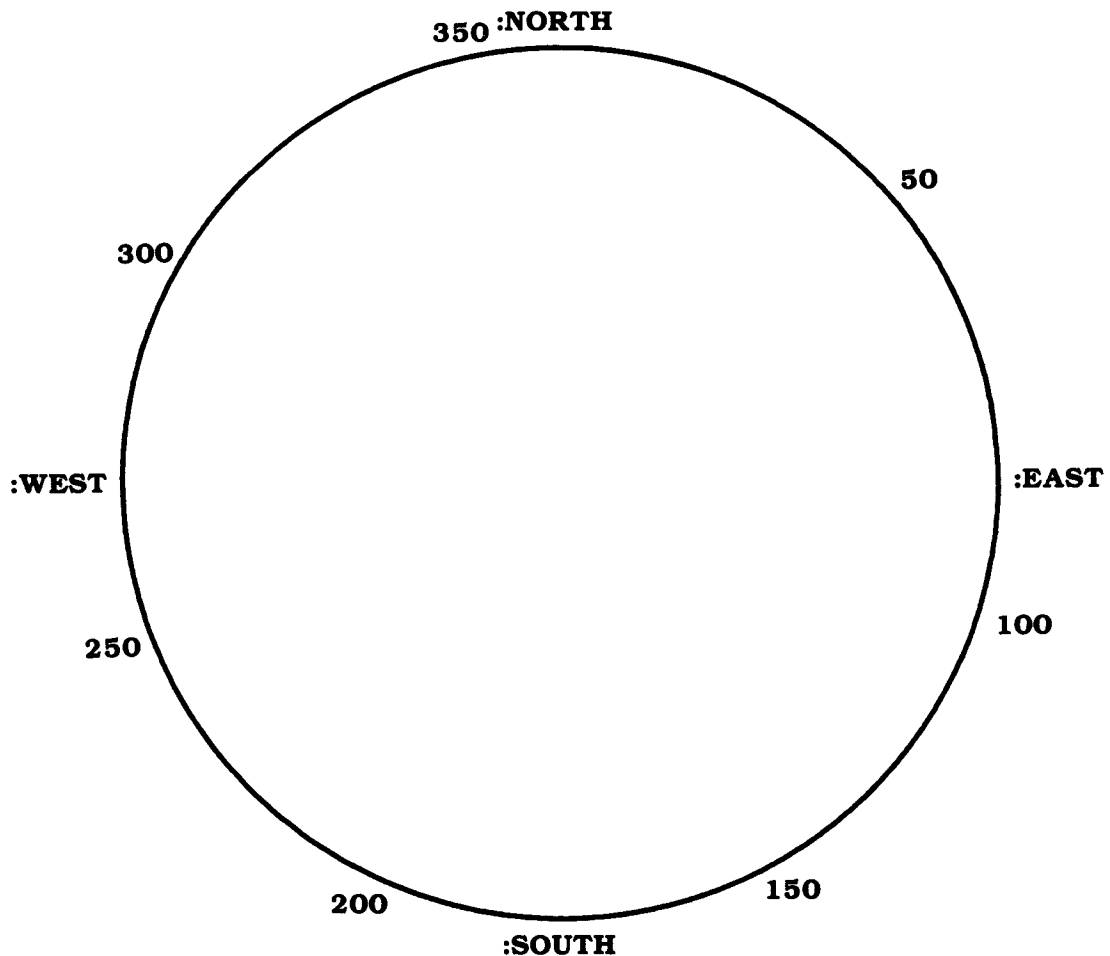
TI LOGO CURRICULUM GUIDE

Introducing SETHEADING

Material ■ At least 14 blank cards or pieces of paper.

Write each of the following numbers and words on a separate card: 50, 100, 150, 200, 250, 300, 350, :NORTH, :SOUTH, :EAST, :WEST.

Ask a group of students to sit in a circle on the floor. Make a "human compass" by putting the :NORTH, :SOUTH, :EAST, and :WEST cards in their appropriate places in the circle. Talk to the students about a compass, possibly even have one available to show. Explain that directions can be named by both words and numbers. Then put the numbered cards in their appropriate places in the circle.



Using the blank cards, ask the students to pick other numbers between 0 and 360. (This demonstrates that other numbers beside multiples of 50 are available.) Let them write the numbers on the cards and place themselves at the appropriate points in the circle.

Then pick a student to be a student sprite. Have the student sprite stand in the middle of the human compass. Let students take turns giving SETHEADING commands. If the words for the directions are used, remind the students to use dots (for example, SETHEADING dots EAST). The student sprite should turn and face the given heading. Suggest that the students use numbers other than those on the cards. When a student sprite has had enough experience, give him or her a speed. SETSPEED 10 lets the student sprite walk out of the circle in the direction he or she was facing. Then another student becomes the student sprite.

Remind students that, unlike the Turtle, when a sprite is given a new heading, the *shape* it carries does not turn to the indicated heading. The sprite accepts the heading, not the shape.



Depending on your students' abilities, it may be appropriate to explain and demonstrate any or all of the following concepts:

- Right and left are relative to where you are, whereas heading is a permanent direction. (North is always the same direction, but if you turn RIGHT 50, you may not face the same way as somebody else who turns RIGHT 50.)
- Both the Turtle and the sprites understand the RIGHT, LEFT, and SETHEADING commands. RIGHT and LEFT are generally used with the Turtle. Since speed is generally associated with a heading, the SETHEADING command is used more frequently when talking to sprites.

TI LOGO CURRICULUM GUIDE

Introducing EACH

When you are talking to a group of sprites, the EACH command gets the attention of one sprite at a time within the group.

Gather a group of students, assign each student a number, and tell them the whole group is called :EVERYONE (dots EVERYONE). Explain that when you say TELL :EVERYONE, the next command given is performed by every member of the group at the same time. Then explain that, in comparison, when you say EACH and then give another command, you are telling each sprite to do something *one* at a time, starting with the first and continuing in sequence to the last student.

To teach and reinforce the EACH concept, assign each student a sprite number and play the following game using Simon Says-type actions. Play as long as necessary.

<i>What To Say</i>	<i>How To Say It</i>	<i>What Happens</i>
CALL [0 1 2 3 ...] "EVERYONE	CALL square bracket 0 1 2 3 ... square bracket quotes everyone	assigns the name EVERYONE to the list of student sprites
TELL :EVERYONE	TELL dots everyone	all students assigned to EVERYONE listen
raise your right arm	raise your right arm	total group raises right arm in unison
EACH [lower your right arm]	EACH square bracket lower your right arm square bracket	each lowers right arm, one after the other, starting with first numbered sprite in the group
clap one time	clap one time	group claps one time in unison
clap three times	clap three times	group claps three times in unison
EACH [clap once]	EACH square bracket clap one time square bracket	each student claps one time, one after the other, starting with first numbered student in the list
stand up	stand up	group stands up
sit down	sit down	group sits down
EACH [stand up]	EACH square bracket stand up square bracket	each student stands up, one after the other, starting with first numbered student in the group
sit down	sit down	group sits down



At some point you may want to rearrange the order of the students, giving each student a chance to be the first sprite. It also gives the students experience in naming teams and demonstrating that EACH doesn't go in chronological order, but in the order the sprites are listed in the team list.

<i>What to say</i>	<i>How to say it</i>	<i>What happens</i>
CALL [30 29 28...] "GROUP	CALL square bracket 30 29 28...square bracket quotes group	assigns the name GROUP to the student sprites
TELL :GROUP	TELL dots group	all students assigned to GROUP listen
touch the table	touch the table	students touch the table in unison
EACH [touch your ear]	EACH square bracket touch your ear square bracket	starting with first numbered sprite in the list, each student touches his or her ear

TI LOGO CURRICULUM GUIDE

Introducing YOURNUMBER

Assign each student in the class a number. Depending on their academic level, you may want to make a number card for each student to hold. Explain that in this game they are to pretend to be the sprite with that number. Every sprite knows its number. Say to the student assigned number one, "TELL SPRITE 1, say YOURNUMBER." That student should reply, "One." Repeat this command with a few other students.

Then say to the student assigned number 10, "TELL SPRITE 10, clap your hands YOURNUMBER of times." Continue this process of telling a student to perform some action "YOURNUMBER" of times until everyone has a turn to do something. Then explain that sprites can also add to and subtract from their numbers. They can even multiply or divide their numbers. This time go around the room giving each student's sprite a math problem that uses YOURNUMBER. Be certain that the problem is within a student's capability.

Some sample commands appear below.

<i>Command</i>	<i>Student Response</i>
TELL SPRITE 1 say YOURNUMBER plus 8	9
TELL SPRITE 2 say YOURNUMBER times 6	12
TELL SPRITE 14 clap YOURNUMBER minus 4	claps 10 times
TELL SPRITE 9 jump YOURNUMBER divided by 3	jumps 3 times

TI LOGO Simulation Games



EACH and YOURNUMBER together

This game is similar to the games "Introducing EACH" and "Introducing YOURNUMBER." Assign each student in your class a number. Make sure they are comfortable with the concepts of EACH and YOURNUMBER separately before playing this game that combines the concepts.

Explain that combining EACH and YOURNUMBER is a very powerful and exciting concept. When a student talks to a team of sprites, each sprite can do something different with just one command.

Remind your students that the command EACH talks to a team of sprites one at a time in the order they are listed in the team. And remember, each sprite knows its own number.

Assign six students to be members of the TEAM. Then give the TEAM the following commands combining EACH and YOURNUMBER.

<i>Command</i>	<i>How To Say It</i>	<i>Student Response</i>
CALL [1 2 3 4 5 6] "TEAM	CALL square bracket 1 2 3 4 5 6 square bracket quotes TEAM	gives students 1, 2, 3, 4, 5, and 6 the name of TEAM
EACH [CALL OUT YOURNUMBER]	EACH square bracket call out YOURNUMBER square bracket	each calls out his or her number, one after the other
EACH [CALL OUT YOURNUMBER PLUS 2]	EACH square bracket call out YOURNUMBER plus 2 square bracket	each calls out his or her number plus 2, one after the other
EACH [CLAP YOUR HANDS TWICE]	EACH square bracket clap your hands twice square bracket	each student claps his or her hands twice, one student after the other
EACH [CLAP YOUR HANDS YOURNUMBER OF TIMES]	EACH square bracket clap your hands YOURNUMBER of times square bracket	each student claps his or her hands his or her number of times, one student after another
Trick question: CLAP YOUR HANDS TWICE	clap your hands twice	students clap their hands twice in unison

Next, change the members of the team and play the game again. Have fun giving your students a variety of commands.

TI LOGO CURRICULUM GUIDE

Priority Sprites

Assign each student in the class a number from 0 to 31 or more, depending on the number of students in the class. Pick a spot in your room to be HOME. Using the commands that the computer and students understand, say, "TELL SPRITE 1 HOME." Student 1 goes HOME and faces the class. Now say, "TELL SPRITE 2 HOME." Student 2 also goes HOME and stands directly behind Student 1. Next say, "TELL SPRITE 4 HOME." Student 4 goes HOME and stands behind Student 2. Then say, "TELL SPRITE 0 HOME."

Ask the class where they think Sprite 0 should stand. If they cannot figure it out, explain to them that the value of Sprite 0 is less than the value of Sprite 1. This means that Sprite 0 stands in front of Sprite 1 (Student 1). Have Student 0 stand first in line. (*Note:* At this point talk about how people have the dimension of space and sprites do not. Sprites have only two dimensions — shape and size; whereas people have three — shape, size, and depth.) Now say, "TELL SPRITE 3 HOME." Student 3 should take the position between Sprite 2 (Student 2) and Sprite 4 (Student 4). Next say, "TELL :ALL" to sit down. All of the students should sit down.

Before starting the second portion of the activity, be sure you have made large cutouts of the five primitive shapes. They can be any of the 16 colors the computer knows. Have two students be sprites numbered 2 and 10. Hold up a ball and box shape and ask your class, "How can you get these two sprites to carry these two shapes with the ball shape in front of the box shape? Can you tell me the commands to do this?"

ANSWER: TELL SPRITE 2
HOME
CARRY :BALL
SETCOLOR :BLUE
TELL SPRITE 10
HOME
CARRY :BOX
SETCOLOR :RED

To help demonstrate the effect, have the students hold the shapes over their head.

Now tell the students different shape combinations (for example, a red ball behind a black rocket, a blue plane in front of a black rocket, or a yellow box behind a green ball). Have the other students give commands to the sprites to accomplish these combinations. When the students are ready, change the number of sprites standing at HOME. You can also increase the number of shapes in the combinations (for example, a white plane in front of a red ball that is in front of a blue box).



Four Sprites All in a Row

The maximum number of sprites that can be located on the same horizontal axis is four. If more than four appear on the same line, the sprite (or sprites) with the largest number value disappears.

- Materials
- At least five students.
 - Sprite number cards for each student playing.
 - One cutout colored shape for each student playing.

Rules of the game: You can have as many sprites as you want standing side-by-side, but only four that have been talked to can face the front; any extra sprites have to face the back. The four sprites with the smallest numbers must face the front, whether or not they are carrying a shape.

Walk your class through a few examples, and then suggest that the students start predicting which sprites have to face which direction. Then, according to your students' experiences on the computer, give them more complex situations.

Here are some sample exercises.

Have student Sprites 1, 2, 3, 4, and 5 stand side-by-side, each carrying a colored shape. Since 1, 2, 3, and 4 are the four smallest numbered sprites in the group, they face the front while Sprite 5 faces the back.

Add Sprite 6 with a shape to the lineup. Sprite 6 also faces the back.

Take Sprite 2 out of the lineup.

Sprites 1, 3, 4, and 5 are now the four smallest numbered sprites. Sprite 5 turns around and faces the front.

Ask your students, "What can we do to make Sprite 6 face the front?" (ANSWER: Remove Sprite 1, 3, 4, or 5 from the lineup.)

Add Sprite 0 to the lineup. Now Sprite 6 turns back around, and Sprite 0 faces the front.

Put Sprite 2 back in the lineup. Now Sprite 5 turns and faces the back.

Remove Sprites 0, 1, 2, 3, 4, 5, and 6 and have students be Sprites 16, 17, 18, 19, and 20. Give each student a shape. Ask the class, "What direction is each sprite facing?" (Sprite 20's back is showing, while the other four are facing the front.)

Change the number of the sprites again. This time, tell Sprites 3, 7, 15, 4, and 30 to carry a shape. Ask your class, "Who faces the front and who faces the back?" (ANSWER: Sprite 30 faces the back and the other four face the front.)

Take away the shape of Sprite 3. Ask the class, "What happens now?" (ANSWER: Sprite 3 is still there on the row, so it counts as one of the four sprites.) We can only see three shapes, but four sprites are there.

TI LOGO CURRICULUM GUIDE

Introducing Velocity

- Materials
- Some strong string.
 - A cardboard cube made from a cardboard box.

To help your students visualize the effect of using x-velocity and y-velocity together, consider a cardboard cube with two strings attached — one on one face of the box and the other on the perpendicular face. The other end of the string should extend past the surface of the cube so that it can be held by students.

One string represents the x-axis while the other represents the y-axis. The cube represents a sprite.

To introduce the concept of velocity to your students, let a different student be in charge of each string. Make sure the box can slide easily on a level surface and that the strings are long and strong enough to be pulled on. Give the students the following situations:

- “X” — pull and “Y” — don’t pull at all;
- “Y” and “X” — both pull lightly; then
- “Y” — pull hard and “X” — don’t pull.

Watch where the box moves. To further assist your students, draw the “X” and “Y” axis on the surface of the cube or use masking tape to indicate the axes. This makes observation of the motion easier. With this exercise it’s very important that the “X” and “Y” strings are pulled in a straight line. For some students it’s easier to “back up” or walk backwards while holding the strings instead of just pulling each.

Continue by giving the students some sample commands with SXV and SYV.

<i>Command</i>	<i>What To Say</i>	<i>Student Response</i>
SXV 30	set X velocity 30	pulls the x-axis string
SXV 5	set X velocity 5	pulls the x-axis string
SYV 40	set Y velocity 40	continues pulling the x-axis string at a slow speed and simultaneously pulls the y-axis string at a medium speed

Note: When a sprite is given the same x- and y-velocity, the sprite moves in a direction halfway between the x and y. When a sprite is given different x- and y-velocities, the sprite moves in a direction in between but more in the direction of the greater velocity.



Games the Turtle and Sprites Play

Teaching the Computer

This classroom presentation can take the format of a simple lecture and discussion, or it can become elaborate and animated.

With an actual recipe in mind, something like baking a cake, write the ingredients on a large poster board and place the board where the students can read it. Then tell the students, "If you were to bake a cake, you would first go into the kitchen. Then I'd tell you all the steps needed to bake a cake. I'd say, for example, get out a mixing bowl, put two eggs in it, pour in 2 cups of flour and 1 cup of milk, stir, and so on. If you followed all the steps, you'd have a cake to eat. But could you do it again without my help? What could you do to help you remember how to bake a cake? That's correct, you could write the recipe on a card. Then you'd have the instructions available when you wanted to bake a cake."

"You teach the computer to do something just as you bake a cake. Let's try it!"

Tell your students to think of a list of instructions that cause the computer to do something. Then give the list of instructions a name. The order in which you list the instructions is the same order that the computer does the instructions.

Using the TO command, write a procedure with the students. Follow this example, or make one up yourself. The procedure can use either sprites or the Turtle. For example, this procedure creates a scene that looks like a stoplight.

```
TO STOPLIGHT
TELL SPRITE 1
HOME
CARRY :BALL
SETCOLOR :RED
SETHEADING 0
FORWARD 40
TELL SPRITE 2
HOME
CARRY :BALL
SETCOLOR :YELLOW
SETHEADING 0
FORWARD 20
TELL SPRITE 3
HOME
CARRY :BALL
SETCOLOR :GREEN
SETHEADING 0
END
```

When you finish typing the procedure, press **BACK**. After teaching the computer how to STOPLIGHT, the scene still isn't on the display. Type STOPLIGHT and press **ENTER**. (At this point you can make the comparison to baking a cake and say, "Bake a cake.") The computer follows the steps in the procedure and creates the stoplight (the same as reading a recipe card).

Every procedure must have a name so that the computer can tell it apart (just like cakes have different names — chocolate cake, pineapple upside-down cake, or Mrs. Jones' marble cake). Procedure names can be a single letter, a word, or, since the computer can't read (it just memorizes the order of the letters), a combination of words using numbers or punctuation marks. (*Note:* You cannot use the following punctuation marks /, -, +, [,], (,), :, , , =, and *.)

TI LOGO CURRICULUM GUIDE

Dots and Quotes

Play the game, "Dots and Quotes" to help your students understand the functional difference between dots (:) or quotes (") in front of a name.

Say to a student, "Say your name." The student probably responds with his or her first name. You then say, "No, I told you to say the words, 'your name.' Let's try it again. Say your name." This time the student should say, "your name." In a joking manner, respond, "Hi, your name, it's nice to meet you." Without embarrassing the student, explain that that's the point of the little joke — it's supposed to confuse people and help them think about what people mean. It's important to understand when someone asks a question if that person is asking for the words or the meaning of the words. Even if a question is asked in a strange manner, many times the other person can figure out what you want to know.

The computer, however, cannot assume anything. You have to tell it when you want it to use a word and when you want it to use the meaning or value of the word. Quotes (") in front of a word tells the computer to use the word exactly as it is. Dots (:) in front of a word tells the computer to find the value of the word and use the value.

For more practice with these concepts, go around the room telling each student either, "Say quotes your name," or "Say dots your name." They should respond accordingly with either, "your name" or their given name.

<i>Student's Name</i>	<i>What You Say</i>	<i>Student Response</i>
John	say quotes your name	your name
Theresa	say dots your name	Theresa
Nicole	say dots your name	Nicole
Clayton	say quotes your name	your name
Sonia	say quotes your name	your name
Chris	say dots your name	Chris



Introducing Variables

To draw a design that can be made different sizes, we have to write a procedure that uses a variable.

Write a procedure on a poster board or on the chalkboard for the class to see. Use the procedure TO SQUARE, or make your own design.

```
TO SQUARE
TELL TURTLE
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
END
```

Tell the class, "We're going to change this procedure so that we can draw different size squares with it."

Ask one student to be in charge of remembering what size the square is to be each time. For this example, PAT is used.

First, we have to let the computer know somebody is going to help with this procedure, so we put their name right after the title of the procedure. (Note: PAT must be added when the computer is in the Edit mode.)

```
TO SQUARE PAT
```

Ask the class, "Which numbers do we want to change each time we use the procedure — the steps or the turns?" (ANSWER: The steps.) We want the steps to change because every square, no matter how large, has four 90degree angles (or four corners each with a 90 degree Turtle turn).

Alter the SQUARE procedure on the chalkboard to look like this:

```
TO SQUARE PAT
TELL TURTLE
FORWARD :PAT
RIGHT 90
FORWARD :PAT
RIGHT 90
FORWARD :PAT
RIGHT 90
FORWARD :PAT
RIGHT 90
END
```

TI LOGO CURRICULUM GUIDE

In this procedure, PAT remembers the size of the square each time SQUARE is typed and **ENTER** is pressed. Assign a student to be PAT.

Enter the procedure into the computer. Then ask the class, "How big do you want the first square to be?" (For this example, 25 replaces PAT.)

"If we just enter SQUARE, the computer gives the message TELL ME MORE. It knows we have to give PAT a number to remember; that's why PAT is there." Print SQUARE 25 on the board. Because 25 is now the value of PAT, tell the student assigned to be PAT that he or she is to say "25" everytime :PAT (dots PAT) appears in the procedure. The first line of the procedure, FORWARD :PAT, tells the computer to move forward the value of :PAT. As Pat says his or her value, draw a line on the chalkboard 25 steps long. The second line tells the Turtle to turn RIGHT 90. Turn the chalk to indicate this action. Continue reading and doing the lines of the procedure. When you read a line that contains :PAT, Pat should call out "25," and you should draw a line the appropriate length.

When the procedure is completed, Pat has said "25" four times (or the computer has used the value for :PAT four times). Now ask your students, "Do you think we can change the value of :PAT? Let's try it and see."

When given a new value, :PAT forgets the previous value of 25, and remembers the current one. A variable (:PAT is the variable in this procedure) can only remember one value at a time.

The quotes in front of PAT in the title line of the procedure serve a different function from the dots in front of PAT in the FORWARD command lines. The quotes indicate that PAT is the name we are giving to the variable. When you type the name of the procedure followed by 25, PAT is given his or her value for the first time. It's similar to assigning the value for the first time by using the CALL command (CALL 6 "PAT).

When giving a value with the CALL command, the value is assigned to a sprite number. Then, to use the value, type :PAT to tell the computer to substitute a value for the word PAT.

After one demonstration, talk about using names that describe the variable. Write another procedure on the board. (Note: The variable SIDE has to be typed while in the Edit mode.)

```
TO TRIANGLE SIDE
TELL TURTLE
FORWARD :SIDE
RIGHT 120
FORWARD :SIDE
RIGHT 120
FORWARD :SIDE
RIGHT 120
END
```

Ask for a volunteer to be called SIDE. Have that person pretend that is his or her new name, and give a number to replace SIDE.



Recursive procedures

Pick a student; let's say Kay. Tell Kay, "Everytime you hear your name, raise your right hand and then lower it." Then say the name, "Kay." She should raise her right hand and lower it. Have several other people say her name, "Kay." She should raise her hand and lower it each time.

Then tell her you're going to change the instructions. The new ones are, "Each time you hear your name, raise your right hand, lower it, and say, 'Kay'." Then say, "Kay." She should raise her hand, lower it, and say "Kay," which causes her to start over again — raise her hand, lower it, and say "Kay," which starts the process over again. After she says, "Kay" you may need to help her figure out what to do next. Some guiding question might be, "Did you hear your name? You said it. What are you supposed to do when you hear your name?" Ask the class, "Can she ever stop raising her hand, lowering it, and saying her name?" (ANSWER: No.)

What causes a procedure to go on and on forever? (ANSWER: Have it say its own name.)

Use this same game to debug some common bugs in recursive procedures. Some common examples appear below.

Procedure

```
TO FLASH
COLORBACKGROUND :WHITE
COLORBACKGROUND :BLUE
COLORBACKGROUND :RED
FLASH
END
```

Game

```
TO JOHN
RAISE YOUR HAND
SAY, "JOHN"
PUT YOUR HAND DOWN
END
```

If John follows the directions in the above example in the correct order, he never gets to lower his hand. As soon as he says "JOHN," he has to start over again. To debug this one, JOHN needs to be the last command before END.

Procedure

```
TO FLASH
COLORBACKGROUND :WHITE
COLORBACKGROUND :BLUE
COLORBACKGROUND :RED
REPEAT 10 [FLASH]
END
```

Game

```
TO JAN
RAISE YOUR HAND
LOWER IT
SAY JAN 10 TIMES
END
```

This example shows a perpetually recursive procedure. Jan is never able to stop raising and lowering her hand. The "SAY JAN 10 TIMES" line makes it 10 times as recursive (if that were possible, that's like 10 times infinity).

TI LOGO CURRICULUM GUIDE

To debug, put the "repeat 10 times" in a superprocedure and take the recursive line out of the procedure.

Procedure

```
TO SUPERFLASH  
REPEAT 10 [FLASH]  
END
```

TO FLASH

```
COLORBACKGROUND :WHITE  
COLORBACKGROUND :BLUE  
COLORBACKGROUND :RED  
END
```

Game

```
TO SUPERJAN  
SAY JAN 10 TIMES  
END
```

TO JAN

```
RAISE YOUR HAND  
LOWER IT  
END
```

Other bugs will develop with recursive procedures. Creating similar bugs in human procedures can help students develop debugging skills by internalizing them.



CLASSROOM HINTS

Student Journal Writing

Encourage your students, as soon as they are capable of writing, to start keeping their own computer journal. In their journal they can keep references, procedures, ideas, designs for shapes, bugs, and any personal notes they may want.

You may have to help your students decide on a format for their journals. It's a good exercise in organization for students to use a consistent format and to develop one that is understandable, at least to them. The cue cards you've been providing them serve as an example of one style. There are shortcut formats that require less writing on the student's part and yet assume a certain familiarity with the computer and its use.

One format might be as follows.

(command)	TELL :ALL
(command)	TELL SPRITE and a number
(choice)	a number from 0 through 31
(command)	HOME
(command)	FORWARD and a number
(choice)	a number of sprite steps
(command)	BACK and a number
(choice)	a number of sprite steps
(command)	CARRY :a shape
(choice)	PLANE, TRUCK, BALL, BOX, ROCKET
(command)	SETCOLOR :a color
(choice)	OLIVE, PURPLE, ORANGE, YELLOW, SKY, RED, CYAN, RUST, GREEN, BLACK, BLUE, GRAY, WHITE, CLEAR, LEMON, LIME

Variations on this format could include using numbers instead of words, trying to portray both numbers and words, denoting when to press the **SPACE BAR** and **ENTER**, and including short forms for the commands.

Students may include the following in their journal: their own procedures or a friend's, a list of suggestions for things to change in a procedure, or a copy of a procedure seen on a computer to try out later. Ideas for a procedure often take the form of a list of commands to be tried out on the computer sometime in the future.

You might want to make available to your class some 16-by-16 graph paper on which students can create shapes. The graphs can then be stapled into the journal. Or suggest that students get graph paper and draw their own 16-by-16 square grid. Actually drawing the shapes aids in the creation of large shapes (piecing together several smaller shapes), intricate designs, and multi-colored shapes. It also provides a way to record successful designs for future reference.

In debugging projects, students may want to record several debugging tactics in their journals and then check them off as they try each one. They may want to record such results as what information was gained or how difficult it was to debug in that manner. This is especially helpful if the student has to stop working at the computer. When she or he returns, the student can pick up where she or he left off. Other personal notes include frustrations encountered (it's a good release and is fun to go back and read later when the problem has been conquered), triumphs, experiences, or anything else the student might want to record.

Within the classroom time frame, since not everyone can be on the computer at the same time, you might allow time for journal writing. This time can also be used by students to share ideas, catch up on writing down thoughts, recording a project's progress, and finding out about new commands or concepts.

TI LOGO CURRICULUM GUIDE

Teacher Journal Writing

As a teacher, you should keep a journal, too. It sets a good example for your students and serves many of the same reminder functions. Besides being a place to keep track of procedure ideas, procedures written, projects, shape designs, new commands and concepts, notes, bugs, and so forth, you can keep notes on your observations of your students. It would be next to impossible to write down every observation of every student, but there are some so exciting, unexpected, or interesting that you don't want to forget when and how they happened. It is also helpful and informative for future reference.



Computer Vocabulary and Its Importance

There are so many things that students “just can’t put into words,” that it’s nice for them to have some things they *can* talk about. TI LOGO uses logical words to describe many abstract thoughts. It encourages users to describe processes that are often taken for granted. In a very short time many students are verbalizing things they’ve never talked about before. They have a new vocabulary to serve a new purpose. It helps to give your students as much vocabulary to describe the computer and the concepts as possible. This shouldn’t be done in the form of vocabulary tests or words and definitions, but by using the words consistently and constantly so that students pick up the meanings and use the words themselves. If everyone in your class uses the same vocabulary (from your examples), it makes conversation about TI LOGO much simpler. Both the speaker and the user have the same connotation for words relative to TI LOGO.

The following is a list of words and definitions. The words should be used consistently when referring to a piece of machinery or a concept.

Cassette Interface Cable — the black cord that extends from the back right hand side of the computer to the side of the cassette recorder. It carries the information between the computer and the cassette recorder.

cassette recorder — device for storing and recalling TI LOGO procedures on cassette tapes.

cassette tape — medium for storing and recalling TI LOGO procedures.

character — one symbol of a set of symbols, such as those corresponding to the keys on a typewriter. The symbols usually include digits 0 through 9, the letters A through Z, punctuation marks, math operation symbols, and any other symbols which a computer may read, store, or write. In TI LOGO, characters are also the designs made on the 8-by-8 grid in the MAKECHAR mode.

computer — the TI-99/4 or TI-99/4A console with the keyboard. It is designed so that all the other units of the system easily connect to the console. (See the *User’s Reference Guide* for more details.)

Disk Drive Controller — device for telling the disk drive where to position the magnetic head in order to read or write information properly on a diskette. (See owner’s manual for details.)

Disk Memory Drive — device for reading information from and writing information on a diskette. (See owner’s manual for details.)

diskette — 5¼-inch, single-sided, single-density, soft-sectored flexible magnetic film for storing and recalling TI LOGO procedures.

display — information that appears on the screen.

grid — a matrix of lines used for specifying characters or designs.

Memory Expansion unit — a device that attaches to the right side of the computer. It adds the 32K bytes of memory space which are needed to run TI LOGO. (See owner’s manual for more details.)

monitor — video equipment used to display computer work. The TI Color Monitor or a color television with a TI Video Modulator attached can be used. (See owner’s manual for more details.)

TI LOGO CURRICULUM GUIDE

screen — the surface of the monitor on which information is displayed or stored temporarily.

shape — design created on the 16-by-16 square grid in the MAKESHAPE mode.

system — an assembly of components united to form an organized whole.

text — the characters that appear on the display.

TI LOGO Solid State Software™ Command Module — the black 3-by-5-by-1 inch plastic box that slides into the slot on the console. (See the *TI LOGO User's Manual* for more details.)

tiles — areas on the display defined by columns and rows on which characters are placed.

Thermal Printer — device for producing printed copies of procedures written in TI LOGO.

Monitor cable — the black cord with a seven-pronged plug on one end and two plugs on the other end. One end plugs into the computer and the other connects (with two plugs) to the monitor. This carries the information from the computer to be displayed on the monitor.

workspace — the "area" a person uses to design shapes or characters, give commands, or write procedures. It is always to be respected by others.

wrapping — process by which the Turtle or sprites encircle the display appearing on the screen again on the opposite side.



Computer Etiquette

Explain to your students the need to respect another student's TI LOGO work. If a student has designed a shape or written a procedure and is coming back later in the day to work on it, tell your students not to disturb the work. There is plenty of workspace or working memory to store several procedures. There are 26 grids on which to design shapes. Encourage your students to leave other students' work alone and to create their own. However, tell them it is perfectly all right to use someone else's shapes and procedures as long as they are not altered.

When your students are capable of saving procedures, characters, and shapes on cassette tape or diskette, erasing others' work won't be an issue. Stress that those who lose a shape or procedure know that they had the opportunity to save it if they had wanted it.

TI LOGO CURRICULUM GUIDE

Turning the System On and Off

If your students are very young or are using procedures that you are loading into the system, you should be the one to turn the equipment on and off. It should be established that *no one* else is to touch the on/off buttons because this action might result in harming something. For example, if the disk drive is turned off and a diskette is inside, the data on it could be erased.

Different monitors have slightly different shades of colors. For adjusting the color knobs, set the classroom policy as to who can and when.

Decide when you want the students to be responsible for turning the system on and off. As soon as your students are working with primitives or are capable of understanding the process, introduce them to the proper technique for turning the equipment on and off. (See the owner's manuals for details). The computer and its accessories can be left on all day without overheating or causing problems. It's better for the system *not* to be turned on and off continuously. If your students are at a stage of writing procedures but not saving them on cassette tape or diskette or with a hard-copy printer, discuss computer etiquette with them.

Another good policy to establish is that no one uses SHIFT Q (on the TI-99/4 console) or FCTN = (on the TI-99/4A console) or turns the system off unless an unrecoverable crash has occurred.



Placement of the Computer

Since every classroom is different in size and shape, move the computer around, trying different locations until you find the best spot for it. The location of the electric outlets in your room is a factor which affects the placement of the computer. Setting the computer with the monitor facing toward the wall, so that only the user sees the display, can be more distracting than if the monitor faces the class. Curiosity can cause students to leave their seats to see what is going on, especially when students start changing the background color of the display. The flashing colors reflect off the wall and the face of the user, and everyone can tell something exciting is going on.

If the computer's back is to the wall and the monitor faces the room, everyone can see what's going on. There is less distraction because students do not leave their seats to see what's going on. If most of the desks or tables in the classroom face in one direction, the computer can be behind all of the seats. If you plan to do demonstrations or have "show and tell" times when students share procedures and projects, the computer needs to be in a location where many can gather around it or where it is visible to all from their seats.

If you have more than one computer, set them next to each other so that some sharing can go on either by looking or low-level whispering. Students are usually able to contain the volume of such conversation. Some remarkable projects can result from students working together on side-by-side computers. With this configuration, you are able to dedicate time to those who really need your help.

BIBLIOGRAPHY

- Banet, B. "Computers and early learning." *Creative Computing*, 1978, 4 (5) pp. 90-95.
- Case, R. "Piaget's theory of child development and its implications." In *Readings in Early Childhood Education 79/80*. Edited by J.S. McKee. (Guilford, Conn.: Dushkin Publishing Group, Inc., 1979).
- Perlman, R. *Using computer technology to provide a creative learning environment for preschool children* (LOGO Memo No. 24). (Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory, May 1976).
- Watt, D. H. "A comparison of the problem solving styles of two students learning LOGO: a computer language for children." *Creative Computing*, 1979, 5 (12) pp. 86-88, 90.
- Larsen, S. G. "Kids and computers: the future is today." *Creative Computing*, 1979, 5 (9) pp. 58-60.
- Papert, S. A. *Uses of technology to enhance education* (Artificial Intelligence Memo No. 298). (Cambridge Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1973).
- Papert, S. A., & Weir, S. *Information prosthetics for the handicapped* (Artificial Intelligence Memo No. 496). (Cambridge, Mass." Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1978).
- Papert, Seymour. *Mindstorms — Children, Computers, and Powerful Ideas*. (New York: Basic Books, Inc., 1980).

Appendix A


CUE CARDS


A "cue card" is a simplified set of instructions indicating what keys the learner presses and the results of pressing those keys. It can be used to introduce an activity to a student and to assist a student in working independently.

A copy of each cue card available with this guide follows. At the bottom of each card is a note indicating which activity and file are appropriate for use with the card and giving permission for the card to be reproduced.

LINE **ENTER**

F = -

R = 

L = 

Q = Quit

PARK 

G = 

C = 

R O ↑

B O ←

Y O →

W O ↓

A = Another

Q = Quit

PEOPLE

R O ↑
B O ←
Y O →
W O ↓

A = Another

Q = Quit

DRAW

F = - forward

B = - back

R =  right

L =  left


D =  draw

N =  nodraw

Q = Quit

DALLAS ENTER

T = 

P = 

R O ↑ 0

B O ← 1

Y O → 2

W O ↓ 3

4

5

A = Another

Q = Quit

BUILD

R O ↑
B O ←
Y O →
W O ↓

A = Another

Q = Quit

PARK

G = 

C = 

R O ↑

B O

Y O ←

W O

G O →

O O

P O ↓

A = Another

Q = Quit

PEOPLE

R O ↑

B O

Y O ←

W O

G O →

O O

P O ↓

A = Another

Q = Quit

DALLAS

T =  P = 

R	O	↑	0
B	O		1
Y	O	←	2
W	O		3
G	O	→	4
O	O		5
P	O	↓	6
			7
			8
			9

A = Another

Q = Quit

BUILD ENTER

R O ↑
B O
Y O ←
W O
G O →
O O
P O ↓

A = Another

Q = Quit

MOVE **ENTER**

HALT **ENTER**

PARK ENTER

g =  c = 

r ○ ↑

b ○

y ○ ←

w ○

g ○ →

o ○

p ○ ↓

a = another

q = quit

PEOPLE

r	○	↑
b	○	
y	○	←
w	○	
g	○	→
o	○	
p	○	↓

a = another

q = quit

DALLAS

t =  p = 

r	○	↑	0
b	○		1
y	○	←	2
w	○		3
g	○	→	4
o	○		5
p	○	↓	6
			7
			8
			9

a = another

q = quit

BUILD

r ○ ↑

b ○

y ○ ←

w ○

g ○ →

o ○

p ○ ↓

a = another

q = quit

ROAD

r =  right

l =  left

f = — forward

b = — back

d =  draw

n =  nodraw

q = quit

AIM

r = turns right

l = turns left

f_23 = steps forward

b_6 = steps back

or

0	10	20
1	11	21
2	12	22
3	13	23
4	14	24
5	15	25
6	16	26
7	17	27
8	18	28
9	19	29

q = quit

PAINT

f = — forward

b = — back

r =  right

l =  left

p =  paint

n =  nopaint

s =  setcolor

c =  colorbackground

q = quit

GRID_17 **ENTER**

or

15	19	23
16	20	24
17	21	25
18	22	

□ = ↑ ← → ↓

■ = FCTN or SHIFT

↑ ← → ↓ ↑ ← → ↓

BACK = FCTN or SHIFT

9

Z

LISTEN

↑ = moves up

← = moves left

→ = moves right

↓ = moves down

q = quit

MOVE_21

or

0	10	20	30	40
1	11	21	31	41
2	12	22	32	42
3	13	23	33	43
4	14	24	34	44
5	15	25	35	45
6	16	26	36	46
7	17	27	37	47
8	18	28	38	48
9	19	29	39	49

HALT

TARGET

r_142 = turns right

l_39 = turns left

or

0	60	120	180	240	300
10	70	130	190	250	310
20	80	140	200	260	320
30	90	150	210	270	330
40	100	160	220	280	340
50	110	170	230	290	350
					360

f_11 = steps forward

b_86 = steps back

or

0	20	40	60	80	100
10	30	50	70	90	

q = quit

TELL_SPRITE_1

HOME

 CARRY_:TRUCK

 CARRY_:PLANE

 CARRY_:ROCKET

 CARRY_:BALL

 CARRY_:BOX

 SETCOLOR_:RED

 SETCOLOR_:BLUE

 SETCOLOR_:YELLOW

 SETCOLOR_:WHITE

 SETCOLOR_:GREEN

 SETCOLOR_:ORANGE

 SETCOLOR_:PURPLE

 SETCOLOR_:BLACK

[erase] = [FCTN] or [SHIFT]
 [3] [T]

TELL_SPRITE_2

TELL_SPRITE_3

TELL_SPRITE_4

HOME

HOME

HOME

BACK_40

BACK_20

FORWARD_20



CARRY_:TRUCK



CARRY_:PLANE



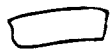
CARRY_:ROCKET



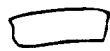
CARRY_:BALL



CARRY_:BOX



SETCOLOR_:RED



SETCOLOR_:BLUE



SETCOLOR_:YELLOW



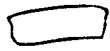
SETCOLOR_:WHITE



SETCOLOR_:GREEN



SETCOLOR_:ORANGE



SETCOLOR_:PURPLE








SETCOLOR_:BLACK

[erase] = [FCTN] or [SHIFT]
 [3] [T]

Appendix B

SHAPE AND COLOR CHARTS

The following chart gives the number and shape of each of the five sprites defined by TI LOGO. When you tell a sprite or a list of sprites to CARRY a shape, dots (:) must be typed in front of the word. No dots (:) is typed if you use the number. You can use either the word or the number since the computer understands both.

<i>Number</i>	<i>Shape</i>	<i>Design</i>
1	PLANE	
2	TRUCK	
3	ROCKET	
4	BALL	
5	BOX	

The following chart lists the 16 colors and their numbers. When you tell a sprite or a list of sprites to SETCOLOR, dots (:) must be typed in front of the word. No dots (:) is typed if you use the number. You can use either the word or the number since the computer understands both.

<i>Color</i>	<i>Number</i>	<i>Color</i>	<i>Number</i>
CLEAR	0	RUST	8
BLACK	1	ORANGE	9
GREEN	2	YELLOW	10
LIME	3	LEMON	11
BLUE	4	OLIVE	12
SKY	5	PURPLE	13
RED	6	GRAY	14
CYAN	7	WHITE	15

Appendix C

STANDARD CHARACTER CODES

All the characters that print on the display (letters, numbers, and symbols) are identified by numeric *character codes*. The standard characters are represented by character codes 32 through 95. These sixty-four codes are grouped into eight *character sets* for color graphics purposes.

Set 1		Set 2		Set 3		Set 4	
Code Number	Character	Code Number	Character	Code Number	Character	Code Number	Character
32	(space)	40	(48	0	56	8
33	!	41)	49	1	57	9
34	“	42	*	50	2	58	:
35	#	43	+	51	3	59	;
36	\$	44	,	52	4	60	>
37	%	45	-	53	5	61	=
38	&	46	.	54	6	62	<
39	'	47	/	55	7	63	?

Set 5		Set 6		Set 7		Set 8	
Code Number	Character	Code Number	Character	Code Number	Character	Code Number	Character
64	@	72	H	80	P	88	X
65	A	73	I	81	Q	89	Y
66	B	74	J	82	R	90	Z
67	C	75	K	83	S	91	[
68	D	76	L	84	T	92	\
69	E	77	M	85	U	93]
70	F	78	N	86	V	94	^
71	G	79	O	87	W	95	_

All alphabetical characters are displayed on the screen as capital letters. TI LOGO disregards the ALPHA LOCK key on the TI-99/4A computer, and certain symbols on the TI-99/4A keyboard are not available in TI LOGO.

Appendix D

QUICK REFERENCE GUIDE

Note that the key sequences required to access special functions depend on the type of computer console you have.

<u>TI-99/4</u>	<u>TI-99/4A</u>	
AID (SHIFT A)	AID (FCTN 7)	Causes the computer to pause.
BACK (SHIFT Z)	BACK (FCTN 9)	<ul style="list-style-type: none">■ Leaves "teaching," Save, and Recall Modes and returns the computer to the mode it was in.■ Stops a procedure.■ Leaves the Edit Mode, MAKESHAPE and MAKECHAR.
BEGIN (SHIFT W)	BEGIN (FCTN 5)	Moves the cursor to the beginning of the line in the Edit Mode.
CLEAR (SHIFT C)	CLEAR (FCTN 4)	<ul style="list-style-type: none">■ Clears the MAKESHAPE and MAKECHAR grids.■ Erases what is above and to the right of the cursor in the Edit Mode.
DELETE (SHIFT F)	DELETE (FCTN 1)	<ul style="list-style-type: none">■ Erases what is above the cursor.■ Moves a line up one line if the cursor is at the end of the line in the Edit Mode.
ERASE (SHIFT T)	ERASE (FCTN 3)	<ul style="list-style-type: none">■ Erases what is one space to the left of the cursor.■ Moves a line up one line if the cursor is under the first character of a line in the Edit Mode.
PROC'D (SHIFT V)	PROC'D (FCTN 6)	Moves the cursor to the end of the line in the Edit Mode.
↑ (SHIFT E)	↑ (FCTN E)	<ul style="list-style-type: none">■ Moves the cursor up one line in the Edit Mode.■ Blackens a square on the MAKESHAPE and MAKECHAR grids as the cursor moves up one square.
← (SHIFT S)	← (FCTN S)	<ul style="list-style-type: none">■ Moves the cursor left one space in the Edit Mode.■ Blackens a square on the MAKESHAPE and MAKECHAR grids as the cursor moves left one square.
→ (SHIFT D)	→ (FCTN D)	<ul style="list-style-type: none">■ Moves the cursor right one space in the Edit Mode.■ Blackens a square on the MAKESHAPE and MAKECHAR grids as the cursor moves right one square.
↓ (SHIFT X)	↓ (FCTN X)	<ul style="list-style-type: none">■ Moves the cursor down one line in the Edit Mode.■ Blackens a square on the MAKESHAPE and MAKECHAR grids as the cursor moves down one square.
SPACE	SPACE	<ul style="list-style-type: none">■ Leaves a blank space in the type in the Sprite and Turtle Modes.■ Reviews file names in the Save and Recall Modes.
{ (SHIFT 4)	}	Types a left bracket.
[(SHIFT 5)]	Types a right bracket.
QUIT (SHIFT Q)	QUIT (FCTN =)	Stops TI LOGO and returns to the master title screen.

IN CASE OF DIFFICULTY

1. Be sure that the diskette or cassette you are using is the correct one. For a diskette, use the Catalog command on your Disk Manager Command Module to check for the correct program; for a cassette tape, check the label.
2. Be sure that you have inserted the TI LOGO Command Module into the slot on the console.
3. If your computer does not respond to the RECALL command, be sure you have selected TI LOGO. If so, follow the instructions that appear on the display and press **BACK** to return to TI LOGO. Then retype RECALL and press **ENTER**.
4. Ensure that your cassette recorder or disk system is properly connected and turned on. Be certain that you have turned on all peripheral devices before you turn on the computer.
5. If your program does not appear to be working correctly, press **BACK** and remove the diskette from the disk drive or the cassette from the recorder. Reinsert the diskette or the cassette, and follow the "User Instructions" carefully. If the program still does not appear to be working properly, remove the cassette from the recorder or the diskette from the disk drive, turn the computer off, wait several seconds, and turn it on again. Then load the program again.
6. If you are having difficulty operating your Home Computer or are receiving error messages, refer to the "Maintenance and Service Information" and "Error Messages" appendices in your *User's Reference Guide* for additional help.
7. If you continue to have difficulty with your Texas Instruments computer or the TI LOGO CURRICULUM GUIDE package, please contact the dealer from whom you purchased the unit or package for service directions.

THREE-MONTH LIMITED WARRANTY COMPUTER SOFTWARE MEDIA

Texas Instruments Incorporated extends this consumer warranty only to the original consumer purchaser.

WARRANTY COVERAGE

This warranty covers the case components of the software package. The components include all cassette tapes, diskettes, plastics, containers, and all other hardware contained in this software package ("the Hardware"). This limited warranty does not extend to the programs contained in the software media and in the accompanying book materials ("the Programs").

The Hardware is warranted against malfunction due to defective materials or construction. **THIS WARRANTY IS VOID IF THE HARDWARE HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLIGENCE, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIAL OR CONSTRUCTION.**

WARRANTY DURATION

The Hardware is warranted for a period of three months from the date of original purchase by the consumer.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE THREE-MONTH PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE HARDWARE OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

PERFORMANCE BY TI UNDER WARRANTY

During the three-month warranty period, defective Hardware will be replaced when it is returned postage prepaid to a Texas Instruments Service Facility listed below. The replacement Hardware will be warranted for a period of three months from date of replacement. TI strongly recommends that you insure the Hardware for value prior to mailing.



TEXAS INSTRUMENTS CONSUMER SERVICE FACILITIES

Texas Instruments Service Facility
P. O. Box 2500
Lubbock, Texas 79408

Geophysical Services Incorporated
41 Shelley Road
Richmond Hill, Ontario, Canada L4C5G4

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information.

Texas Instruments Exchange Center
831 South Douglas Street
El Segundo, California 90245
(213)973-1803

Texas Instruments Consumer Service
6700 Southwest 105th
Kristen Square, Suite 110
Beaverton, Oregon 97005
(503)643-6758

IMPORTANT NOTICE OF DISCLAIMER REGARDING THE PROGRAMS

The following should be read and understood before purchasing and/or using the software media.

TI does not warrant the Programs will be free from error or will meet the specific requirements of the consumer. The consumer assumes complete responsibility for any decisions made or actions taken based on information obtained using the Programs. Any statements made concerning the utility of the Programs are not to be construed as express or implied warranties.

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAMS AND MAKES ALL PROGRAMS AVAILABLE SOLELY ON AN "AS IS" BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THE PROGRAMS AND THE SOLE AND EXCLUSIVE LIABILITY OF TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THE SOFTWARE MEDIA. MOREOVER, TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER BY ANY OTHER PARTY AGAINST THE USER OF THE PROGRAMS.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.



TI LOGO REFERENCE GUIDE

TABLE OF CONTENTS



GENERAL INFORMATION	R-1
TI LOGO PRIMITIVES	R-2
TI LOGO PROCEDURES	R-3
Recursive Procedure	R-4
Subprocedures	R-4
Labels in Procedures	R-4
Comments Included in Procedure Statements	R-5
Editing a Procedure	R-5
USER-DEFINED VARIABLES	R-5
ARITHMETICAL AND RELATIONAL EXPRESSIONS	R-6
Arithmetical Expressions	R-6
Relational Expressions	R-7
TI LOGO VOCABULARY	R-8
BACK	R-11
BACKGROUND	R-11
BEEP	R-12
BOTH	R-13
BUTFIRST	R-14
BUTLAST	R-15
BYE	R-15
CALL	R-16
CARRY	R-17
CHARNUM	R-17
CLEARSCREEN	R-18
COLOR	R-19
COLORBACKGROUND	R-20
CONTENTS	R-21
CONTINUE	R-21
DEFINE	R-22
DIFFERENCE	R-23
DOT	R-23
EACH	R-24
EDIT	R-25
EITHER	R-26
END	R-26
ERASE	R-27
FIRST	R-27
FORWARD	R-28
FPUT	R-29
FREEZE	R-29
GO	R-30
GREATER	R-30
HEADING	R-31
HIDETURTLE	R-31
HOME	R-32

TI LOGO REFERENCE GUIDE

IF...THEN...ELSE	R-33
IFF/IFT	R-34
IS	R-35
JOY	R-36
LAST	R-37
LEFT	R-38
LESS	R-39
LOOKLIKE	R-40
LPUT	R-40
MAKE	R-41
MAKECHAR	R-42
MAKESHape	R-43
NOBEEP	R-44
NOT	R-45
NOTURTLE	R-45
NUMBER?	R-46
NUMBEROF	R-46
OUTPUT	R-47
PA	R-48
PENDOWN	R-48
PENERASE	R-49
PENREVERSE	R-49
PENUP	R-50
PN	R-50
PO	R-51
PP	R-51
PRINT	R-52
PRINTCHAR	R-54
PRODUCT	R-54
PUTTILE	R-55
QUOTIENT	R-55
RANDOM	R-56
RC?	R-56
READCHAR	R-57
READLINE	R-57
RECALL	R-58
REPEAT	R-59
RIGHT	R-60
RUN	R-61
SAVE	R-62
SENTENCE	R-63
SETCOLOR	R-64
SETHEADING	R-66
SETSPEED	R-67
SHAPE	R-67

TI LOGO REFERENCE GUIDE



SHOWTURTLE	R-68
SPEED	R-68
STOP	R-69
SUM	R-69
SV	R-70
SX/SY/SXY	R-71
SKV/SYV	R-72
TELL	R-73
TEST	R-74
THAW	R-74
THING	R-75
THING?	R-76
TO	R-77
TRACEBACK	R-79
TYPE	R-80
WAIT	R-80
WHERE	R-80
WHO	R-81
WORD	R-81
WORD?	R-81
XCOR	R-82
XVEL	R-82
YCOR	R-83
YVEL	R-83
YOURNUMBER	R-84

TI LOGO REFERENCE GUIDE



GENERAL INFORMATION

In addition to the rich learning environment it creates for children, TI LOGO is a powerful computer programming language. The "primitives" (predefined commands, operations, graphics, and variables) of TI LOGO are easy to learn; yet they give you access to a wide range of computer operations, including the development of intricate graphics and animation. We believe that TI LOGO offers an exciting new dimension in computer programming for anyone interested in learning to communicate with a computer.

This manual provides a complete guide to the TI LOGO programming language. Each command or operation is discussed in detail with accompanying examples to illustrate its performance. The manual assumes that you have read the *TI LOGO User's Manual* and are generally familiar with the concepts of TI LOGO, such as the Turtle mode, sprites, and the structure of procedures. Beyond this assumption, however, the *Reference Guide* details how TI LOGO operates as a programming language.

The first five sections of this manual contain general information about TI LOGO primitives, procedures, variables, arithmetical and relational expressions, and vocabulary. The remainder of the manual, arranged in alphabetical order, describes each command or operation and its function.

Each discussion of a command or operation begins with a *format line* that illustrates its acceptable format and any parameters (inputs) that may or must be included with the primitive. The name of the command or operation is shown in capital letters, and the accompanying parameters are printed in italic type.

The conventions for parameters are:

<i>action</i>	An action to be performed based on the result of a conditional test.
<i>expression</i>	An arithmetical or relational expression to be evaluated by a command or operation.
<i>item</i>	A parameter that may be a name, a word, a list, or an expression, depending on the usage of the command or operation.
[<i>list</i>]	A group of alphanumeric characters (a <i>string</i>), including blank spaces, enclosed in brackets.
" <i>name</i>	A variable name, preceded by quotation marks.
<i>value</i> or <i>number</i>	A numeric value.
: <i>word</i>	A variable that has been assigned a value. It must be preceded by : (dots).

Below the format line is shown the "short form," if allowed, of the name of the command or operation.

The first paragraph of the discussion gives a brief description of the function of the command or operation, followed by a reference to other commands or operations (if any) that are related. The remainder of the discussion explains the examples and notes any restrictions or cautions that may be applicable to the use of the command or operation.

TI LOGO REFERENCE GUIDE

TI LOGO PRIMITIVES

A "primitive" is any word that is predefined in TI LOGO as a command, an operation, a graphic, or a variable name.

Commands are words that tell the computer to perform predefined sets of instructions (subprograms). In some cases they can be used alone, such as CLEARSCREEN; in other instances they require additional parameters, such as FORWARD 30 or SETCOLOR :BLACK. TI LOGO allows more than one command to be included in a statement in certain cases.

Operations are words that tell the computer to perform specific functions that return values (output). Note that an operation must be used in conjunction with a command. For example, in the statement

```
PRINT DIFFERENCE 45 39
```

the operation DIFFERENCE subtracts 39 from 45. Without the inclusion of the PRINT command, however, the computer displays

```
TELL ME WHAT TO DO WITH 6
```

to indicate that a command is required.

Certain *graphics* are predefined in TI LOGO. These include the five shapes that sprites can carry, the alphanumeric characters, and the other punctuation and special symbols that can be displayed on the screen. These graphics can be referenced by their predefined variable names (see below) or by their numeric codes. For example,

```
SETCOLOR :BLACK  
SETCOLOR 1
```

are equivalent since :BLACK is the predefined variable name for color 1 (black).

Variable names are preassigned in TI LOGO to the shapes, the colors, and certain other items, such as the four major compass directions (NORTH, SOUTH, EAST, and WEST). These predefined variables have numeric values and can be used as parameters in commands and operations. For example,

```
SETHEADING :NORTH  
SETHEADING :EAST + 10
```

are acceptable statements.

Predefined Variables

BALL	OLIVE
BLACK	ORANGE
BLUE	PLANE
BOX	PURPLE
CLEAR	RED
CYAN	ROCKET
EAST	RUST
GRAY	SKY
GREEN	SOUTH
LEMON	TRUCK
LIME	WHITE
NORTH	YELLOW

(See also "User-Defined Variables.")

TI LOGO REFERENCE GUIDE



TI LOGO PROCEDURES

A *procedure* (program) consists of a series of *statements* that direct the computer to accomplish some task. A *statement* contains one or more commands and/or operations, plus any parameters required to fulfill the commands or operations. Unlike some programming languages, TI LOGO allows multiple-command statements. In general, the format for a TI LOGO statement is

COMMAND parameter operation/COMMAND parameter...operation/COMMAND parameter

where anything following the first command may or may not be optional. For example, the command CLEARSCREEN requires no further inputs to perform its function. On the other hand, the statement

```
PRINT SUM :A :B
```

consists of a command followed by an operation and its parameters, and the statement

```
IF :A = :B THEN PRINT :B ELSE GO :LABEL
```

consists of three commands (IF...THEN...ELSE, PRINT, and GO) and their parameters.

To enter a procedure, type TO and the procedure's name, which can be any name except the name of a command or operation, and then press ENTER. The display screen turns green and shows

```
TO procedure-name  
END
```

The cursor (a flashing underline character) appears at the end of the first line. Press ENTER to advance the cursor to the next line, and type the statements in your procedure, pressing ENTER at the end of each statement. When you have typed your last statement, press BACK to leave the "teaching" (procedure entry) mode.

Note: The last statement in a TI LOGO procedure must be END. Each time you press ENTER at the end of a statement, the END statement that is automatically displayed advances to the next line. If you inadvertently erase END, be sure to retype it before you press BACK.

To run a procedure after you have entered it, simply type its name and press ENTER.

At times you may want to write a procedure that operates on values input from the keyboard. If so, after you enter the "teaching" (procedure) mode by typing TO *procedure-name*, press the SPACE BAR to move the cursor one space to the right of the procedure name. Then type beside the procedure name the variable name(s), separated by blank spaces, to which the input value(s) are to be assigned. Then press ENTER to advance to the next line, and enter your procedure statements.

To run the procedure, type the name and the value(s), separated by blank spaces, and then press ENTER. For example, assume that you have entered the following procedure.

```
TO TURN "A "B  
TELL TURTLE  
FORWARD :A  
RIGHT :B  
FORWARD :A  
END
```

When you run the procedure, type TURN 30 90 and press ENTER. The Turtle draws a right angle on the display.

TI LOGO REFERENCE GUIDE

Recursive Procedures

A procedure that "calls itself" by including its own name in a statement line is called a *recursive procedure*. Recursion causes the procedure to repeat until you press BACK to stop it.

```
TO DRAW
TELL TURTLE
REPEAT 4 [FORWARD 30 RIGHT 90]
RIGHT 10
DRAW
END
```

This procedure continues to draw a design until you press BACK.

Subprocedures

TI LOGO procedures can call other procedures simply by naming them. For example, a *superprocedure* may consist entirely of the names of the *subprocedures* it calls.

```
TO HOUSE
ROOF
SIDES
WINDOW
DOOR
END
```

This procedure calls four subprocedures; the computer automatically branches to each subprocedure as it is named, completes the subprocedure, and returns to the next line of the calling procedure.

In addition, a subprocedure can call other subprocedures, and they, in turn, can call others. The number of concurrent subprocedure calls allowed in TI LOGO is limited only by the amount of available memory space.

Labels in Procedures

A label and the GO command can be used to create a "loop" within a procedure.

```
TO COUNT
CALL 1 "A
CALL 10 "B
LOOP: PRINT :A
IF :A = :B THEN STOP
CALL :A + 1 "A
GO "LOOP
END
```

In this procedure, LOOP is used as a label. Note that the label name must be followed by dots (:), a space, and a command. When the label name is referenced in the GO statement, the name must be preceded by quotation marks (").

See the GO command for additional information.

TI LOGO REFERENCE GUIDE



Comments Included in Procedure Statements

In many cases programmers find it helpful to include in a procedure comments that document what certain statements are doing. TI LOGO allows comments at the end of statement lines, separated from the rest of the statement by a space and a semicolon (;).

```
TEST :A = :B
IFT CALL 1 "A ; RESET LOOP COUNTER
IFF CALL :A + 1 "A ; INCREASE A BY 1
```

These statements test two values for equality. The comments at the ends of the second and third statements describe the actions performed by the statements. When the procedure runs, the computer disregards anything in the statement after the semicolon.

Editing a Procedure

TI LOGO allows you to edit, or change, a procedure after it is entered. You can enter the Edit mode by typing either `TO procedure-name` or `EDIT procedure-name`. The screen turns green and a listing of the procedure appears on the display.

To edit, move the cursor by means of the left-, right-, up-, and down-arrow keys to the location you want to change. Then either press `CLEAR` to erase the old line and type the new one, or type the new line and use `DELEte` to delete the old characters, one at a time. When you have completed your changes, press `BACK` to leave the Edit mode.

For additional information, see the `EDIT` command.

USER-DEFINED VARIABLES

A *variable* is a name to which a value, a word, or a list is assigned by means of the `CALL` or `MAKE` command or by its inclusion in the name line of a procedure (see "TI LOGO Procedures"). The variable name can then be used in commands or procedure statements to represent the value, word, or list assigned to it. For example, if you enter

```
CALL "SAM "A
PRINT :A
```

the computer reponds by displaying `SAM` on the video screen. Likewise, if you enter

```
MAKE "B 1234
PRINT :B
```

the computer prints `1234` on the screen.

Variables defined by the `CALL` or `MAKE` command are called *global* variables, and they can be used both within the procedure where they are defined and outside of the procedure. *Local* variables are those that are stated in the name line of a procedure; their use is restricted to the procedure itself or to subprocedures called by that procedure.

Notice that, when you are assigning an item to a variable, quotation marks (") precede the variable name. Then, when you use the variable in a subsequent command or operation, its name is preceded by : (dots). Dots (a colon) tells the computer that the variable name has already been assigned to an item.

TI LOGO REFERENCE GUIDE

Certain identifying symbols indicate the type of item you are assigning to a variable. A numeric value requires no identifying symbol; a word (a "string" of characters *without* any blank spaces — also called a *name*) must be preceded by quotation marks (""); a list (a group of words or values separated by blank spaces) is enclosed in brackets ([]). Using the CALL command as an example, the formats for items are as follows.

CALL <i>value</i> " <i>variable-name</i>	Defines a numeric value as <i>variable-name</i> .
CALL " <i>name</i> " <i>variable-name</i> or CALL <i>:word</i> " <i>variable-name</i>	Defines a name or a word as <i>variable-name</i> .
CALL [<i>list</i>] " <i>variable-name</i>	Defines a list as <i>variable-name</i> .

A variable that has been assigned a numeric value may be used in a arithmetical operation or expression. For example,

```
CALL 45 "C
PRINT :C + 79
```

causes the computer to display 124, the sum of 45 and 79. If a variable has been assigned to a word or a list, however, it may not be used in arithmetical operations or expressions.

ARITHMETICAL AND RELATIONAL EXPRESSIONS

TI LOGO can evaluate a wide range of arithmetical and relational expressions, including addition, subtraction, multiplication, division, equal, not equal, greater than, and less than.

Arithmetical Expressions

There are two ways to perform arithmetic in TI LOGO. One way is to enter the expression in traditional mathematical format. The arithmetical operators (symbols) are + (add), - (subtract), * (multiply), and / (divide).

```
PRINT 45 + 30
PRINT 45 / 5
```

The other method involves the use of an arithmetical TI LOGO operation (SUM, DIFFERENCE, PRODUCT, or QUOTIENT).

```
PRINT DIFFERENCE 45 30
PRINT PRODUCT 9 5
```

The evaluated output of an arithmetical expression is the result of the mathematical operation in the expression. This value can then be printed on the display, tested to determine some further action, or used in other calculations.

TI LOGO REFERENCE GUIDE



Relational Expressions

Relational expressions are generally used in conditional testing to check or compare two values and to determine some further action based on the evaluated result.

```
IF :A = :B THEN SETSPEED 150
IF :A > :B THEN SETSPEED 150
```

The traditional operators (symbols) are = (equal), > (greater than), and < (less than). TI LOGO also includes the relational operations NOT (not equal, not greater than, not less than), GREATER (greater than), and LESS (less than). Note that the two statements

```
IF GREATER :A :B THEN SETCOLOR 1
IF :A > :B THEN SETCOLOR 1
```

are equivalent.

When evaluated, relational expressions return an output of either true or false, rather than a numerical result. This output can be displayed on the screen or used to determine some further action in a procedure.

```
TO CHECK "A "B
  IFT PRINT OUTPUT :A < :B
  IFF PRINT OUTPUT :A < :B
END

TEST :C = :D
IFT PRINT [THAT IS CORRECT.]
IFF PRINT [TRY AGAIN.]
```

TI LOGO REFERENCE GUIDE

TI LOGO VOCABULARY

TI LOGO has its own unique vocabulary which helps children, parents, and teachers relate to one another with specific terminology when discussing certain aspects of the computer. These terms are listed here for your reference.

Attributes — Characteristics that are given to the Turtle and the sprites. These include color, shape, direction, speed, velocity (relative X- and Y-coordinate speeds), and screen position.

Bug — Something in a procedure that prevents correct operation or that results in an action you don't want.

Character — An alphabetical, numerical, punctuation, or special symbol that can be displayed on the video screen.

Command — A primitive that tells the computer to perform predefined sets of instructions (subprograms).

Comments (;) — A descriptive remark typed at the end of a statement and separated by a semicolon (;). The purpose of a comment is usually to help you remember what the procedure does. The computer disregards characters that follow the semicolon in the statement line.

Cue Card — A simplified set of instructions indicating which keys operate a procedure and the results of pressing those keys. It assists students in working independently and when a new activity is being introduced.

Debugging — Editing a procedure to make the computer perform a desired action.

Default Values — The attributes assigned to the Turtle and sprites when you first turn on the computer or enter a specific mode. The Turtle is assigned the color 1 (black), a HEADING of 0 (NORTH), and X- and Y-coordinates of 0. Sprite 0 is the current listener when the computer is first turned on; its default values are COLOR 0 (clear), SHAPE 0, HEADING 0, SPEED 0, YCOR 0, XCOR -97, XVEL 0, and YVEL 0.

Dots (:) — The LOGO name for a colon (:). The colon always appears in front of a variable name that has already been assigned a value.

Expression — A group of values (or variables representing values) and arithmetical or relational symbols that can be evaluated by the computer, such as $16 + 38 + 45$ or $25 > 48$. Arithmetical expressions return numeric results when evaluated; relational expressions (equal, not equal, greater than, and less than) return true or false as output.

Label — A name that defines a subsection of a procedure. When a defined label is referenced in a GO statement, the procedure returns to the label and repeats the statements contained in the labeled section.

Operation — A primitive that instructs the computer to perform a specific function which returns a value; an operation must be used in conjunction with a command.

Primitives — Words and symbols that are predefined in TI LOGO as commands, operations, graphics, or names.

Procedure — A series of statements, including commands, operations, and their parameters, that "teaches" (or programs) the computer to perform a desired action.

Quotes (") — The LOGO name for quotation marks ("), the symbol that precedes a variable name when it is assigned a value or that delineates a string of characters as a PRINT item.

Recursion — The process of making a procedure call itself by including its name as the last statement before the END statement. As a result, the procedure continues to repeat itself until you stop it by pressing BACK.

TI LOGO REFERENCE GUIDE



Shape — A visual attribute given to a sprite. A shape is designed on a 16 X 16 square grid in the MAKESHape mode. TI LOGO has five predesigned shapes — :BALL, :TRUCK, :PLANE, :ROCKET, :BOX.

Short form — A shortened form of a primitive's name. For single-word names, the short form is the first and last letters of the word. If the name is a compound word, the short form is comprised of the first letter of each of the two words; for example, the short form of COLORBACKGROUND is CB. Not all primitives have short forms.

Sprite — A graphic that has the capability of motion. In order to be seen, a sprite must be given the attributes of COLOR and SHAPE (see "Concept: Talking to a Sprite"). There are 32 sprites available in TI LOGO.

State of the pen — The four conditions which can apply to the Turtle's pen. (See PENDOWN, PENERASE, PENUP, and PENREVERSE.) In addition, the color of the Turtle's ink is sometimes considered a state of the pen.

Subprocedure — A procedure that is called by another procedure.

Superprocedure — A procedure that calls other procedures (subprocedures).

Teaching the computer — A process of entering procedures that "teach" the computer to do something. "Teaching" the computer is another term for "programming" the computer.

Tile — A graphic, designed on an 8 X 8 square grid, which defines a character or symbol.

Turtle — A triangular sprite that has the ability to move, draw, and create geometrical designs.

Variable — A name to which a value, a word, or a list is assigned by means of the CALL or MAKE command or by its inclusion in the name of a procedure. A variable can be any one-word name containing any alphanumeric character. Blank spaces are not allowed in variable names. The two types of variables in TI LOGO are *global* and *local*. Global variables are those that are defined by a CALL or MAKE command. Local variables are defined in the name line of a procedure and are available only within that procedure and its subprocedures, if any (see "Concept: Global and Local Variables" in the TI LOGO Curriculum Guide).

Work space — The memory area used to design shapes or characters, give commands, or enter procedures; everything that exists in the Random Access Memory (RAM) of the computer after it has been turned on. If you load too much information into the computer or create too many designs or procedures, "OUT OF SPACE" appears on the display. At this point, everything you have entered into the computer cannot be accessed and is lost, unless it has been saved at an earlier time. The only way to continue is to press QUIT (or turn off the console) and start again.

Wrapping — A process by which the Turtle or sprites encircle the display, appearing on the screen again on the opposite side.

TI LOGO REFERENCE GUIDE

TI LOGO REFERENCE GUIDE



BACK

BACK *number*

Short form: **BK**

BACK is one of the four direction commands that can be given to the Turtle or a sprite. BACK must be followed by the number of steps the Turtle or sprite is to move; for example, BACK 30. The Turtle or the sprite then moves backwards.

See also: FORWARD, LEFT, RIGHT

In the first example to the right, the Turtle is told to go FORWARD 50 and then BACK 25. Without turning around or erasing the line, the Turtle moves BACK 25 steps. In the second example, the PARK procedure tells a sprite carrying the truck shape to go HOME and then move back.

Examples:

```
TELL TURTLE  
FORWARD 50  
BACK 25
```

```
TO PARK  
TELL SPRITE 1  
CARRY :TRUCK  
SETCOLOR :BLACK  
HOME  
SETHEADING 90  
BACK 110  
END
```

BACKGROUND

BACKGROUND

Short form: **BG**

The operation BACKGROUND can be used to define the color of the screen. The color of the screen is always cyan (a light blue) when the computer is turned on, but each of the 16 colors is available as a screen color.

See also: COLORBACKGROUND

When you type TELL BACKGROUND, the screen becomes the listener. Its color can then be changed by using SETCOLOR *:word* or *number*. The background is the listener until another TELL operation is entered, such as TELL SPRITE 6, TELL TURTLE, or TELL TILE 100.

The recursive procedure shown on the right causes the background to flash from one color to the next. Note that the sprite seems to disappear when the background is the same color as the sprite.

Examples:

```
TELL BACKGROUND  
SETCOLOR :RED
```

```
TO FLASH  
TELL SPRITE 1  
CARRY :BALL  
SETCOLOR :RED  
HOME  
TELL BACKGROUND  
SETCOLOR COLOR + 1  
WAIT 30  
FLASH  
END
```

TI LOGO REFERENCE GUIDE

BEEP

BEEP

The BEEP command tells the computer to produce a tone. Two other commands are used with it: WAIT *number*, which defines the duration of the tone, and NOBEEP, which stops the tone.

See also: NOBEEP, WAIT

In the first example, BEEP, WAIT, and NOBEEP create a tone that sounds five times.

The HONK procedure illustrates how you can include sound effects in LOGO activities. A black truck appears behind a red truck in the middle of the display. The black truck honks three times. When the red truck disappears, the black truck starts moving across the display.

Examples:

```
REPEAT 5 [BEEP WAIT 50  
          NOBEEP WAIT 50]
```

```
TO HONK  
  TELL SPRITE 1  
  CARRY :TRUCKHOME  
  SETHEADING 90  
  BACK 20  
  SETCOLOR :BLACK  
  TELL SPRITE 2  
  HOME  
  CARRY :TRUCK  
  SETCOLOR :RED  
  REPEAT 3 [BEEP WAIT 50  
            NOBEEP WAIT 50]  
  TELL SPRITE 2  
  CARRY 0  
  TELL SPRITE 1  
  SETSPEED 35  
  END
```



BOTH

BOTH *expression expression*

BOTH is an operation that compares two numeric or relational expressions. If both expressions are evaluated as true, the statement is true. If one or both of the expressions are evaluated as false, the statement is false. The true or false value returned by BOTH can then be printed on the display or used with conditional statements.

See also: EITHER, GREATER, IS, LESS, IF...THEN...ELSE, IFT/IFF, TEST

The relational expressions allowed are:

- = Equal
- > Greater than
- < Less than

In the first example on the right, the BOTH operation returns a value of TRUE since both expressions are true. In the second example, the statement is FALSE since the first expression is incorrect.

The CHECKSPEED procedure prints SPEED IS 10 since the speed of the sprite is both more than 9 and less than 11. If you change the speed, CHECKSPEED prints SPEED IS NOT 10.

Examples:

```
PRINT BOTH 5 + 3 = 8
          4 + 2 = 6
PRINT BOTH 5 + 1 = 4
          4 + 2 = 6
```

```
TO CHECKSPEED
  TELL SPRITE 1
  HOME
  CARRY 4
  SETCOLOR :RED
  SETSPEED 10
  TEST BOTH SPEED 9
          SPEED 11
  IFT PRINT [SPEED IS 10]
  IFF PRINT [SPEED IS NOT
            10]
  END
```

TI LOGO REFERENCE GUIDE

BUTFIRST

BUTFIRST [*list*]
BUTFIRST "*name*
BUTFIRST *:word*

Short form: **BF**

BUTFIRST is an operation that returns all but the first item of a list or all but the first character in a name or word. The value of a name or word cannot be numeric. BUTFIRST can be used more than once in a statement and can also be used with other operations.

See also: BUTLAST, FIRST, LAST

The first example prints DOG BIRD on the display. The second example prints BIRD.

BUTFIRST "*word* performs a similar function on a set of characters. The example on the right prints ROW on the display.

The procedure SECOND prints MAY, the second item in the list. The operations in the PRINT statement are performed from right to left, as explained below.

1. BUTFIRST creates a new list consisting of [MAY JUNE].
2. FIRST selects the first item in the new list.
3. PRINT prints the result, MAY, on the display.

Examples:

```
PRINT BUTFIRST [CAT DOG  
  BIRD]  
PRINT BUTFIRST BUTFIRST  
  [CAT DOG BIRD]  
PRINT BUTFIRST "GROW
```

```
TO SECOND  
CALL [APRIL MAY JUNE]  
  "MONTH  
PRINT FIRST BUTFIRST  
  :MONTH  
END
```



BUTLAST

BUTLAST [*list*]
BUTLAST "*name*
BUTLAST *:word*

Short form: **BL**

The BUTLAST operation returns all but the last item in a list or all but the last character in a name or word. The value of a name or word cannot be numeric. BUTLAST can be used more than once in a statement and can also be used with other operations.

See also: BUTFIRST, FIRST, LAST

The first example prints 1 2 3 on the display. The second example prints 1 2.

BUTLAST "*name* performs a similar function on a set of characters. The example on the right prints BUILDING on the display.

The procedure NEXTOLAST prints IS, the second item in the list. The operations in the PRINT statement are performed from right to left, as explained below.

1. BUTLAST creates a new list consisting of [TODAY IS].
2. LAST selects the last item in the new list.
3. PRINT prints the result, IS, on the display.

Examples:

```
PRINT BUTLAST [1 2 3 4]
PRINT BUTLAST BUTLAST [1
  2 3 4]
```

```
PRINT BUTLAST "BUILDINGS
```

```
TO NEXTOLAST
CALL [TODAY IS SUNNY]
  "ITEM
PRINT LAST BUTLAST :ITEM
END
```

BYE

BYE

The BYE command is used to leave TI LOGO at the end of a session. After you enter BYE, the computer responds with AND A PLEASANT DAY TO YOU and returns to the master title screen.

TI LOGO REFERENCE GUIDE

CALL

CALL "name "name

CALL value "name

CALL [list] "name

The CALL command allows you to assign a variable name, a value, or a list of characters to a variable name. CALL takes two inputs: the first is the name, value, or list to be assigned, and the second is the variable that receives the assignment.

See also: MAKE

The first example assigns the name CATHY to the variable GIRL. PRINT :GIRL then causes the display to show CATHY.

In the examples to the right, the name CHARLIE is given the value of 8, the name SAM is given the value of 10, and the name GEORGE is given the value of :CHARLIE + :SAM. When the computer is asked to PRINT :GEORGE, it adds 8 and 10 and prints 18, the value of GEORGE.

In the third example, groups of sprites are assigned to the names PLANES and ROCKETS. These names are then used to talk to the sprites in each group. In this procedure, all the sprites start at HOME and then travel across the display in a random fashion.

Examples:

```
CALL "CATHY "GIRL
PRINT :GIRL
```

```
CALL 8 "CHARLIE
CALL 10 "SAM
CALL :CHARLIE + :SAM
    "GEORGE
PRINT :GEORGE
```

```
CALL [7 8 9 10 11 12]
    "PLANES
CALL [13 14 15 16 17 18]
    "ROCKETS
TO SCENE
TELL :PLANES
HOME
CARRY :PLANE
SETCOLOR :GREEN
SETSPEED 40
TELL :ROCKETS
SXY 0 [- 40]
CARRY :ROCKET
SETCOLOR :BLACK
SETSPEED 90
TELL :ALL
EACH [SETHEADING
    YOURNUMBER * 13]
END
```

TI LOGO REFERENCE GUIDE



CARRY

CARRY *:word*

CARRY *number*

CARRY is an operation used to define or change the shape of a sprite. The sprite's shape can be defined by either a word, such as :BALL, or a number, such as 4. CARRY also can be used to have a sprite carry a shape that is created in the MAKESHAPE mode.

See also: LOOKLIKE, MAKESHAPE

In the first example, Sprite 7 is the active listener and is told to CARRY a :ROCKET which has the color of :BLACK.

In the SHAPES procedure, Sprite 13 first is told to CARRY a :PLANE and then to CARRY the shape whose value is one more than the shape it is currently carrying which, in this case, is the truck.

Examples:

```
TELL SPRITE 7
HOME
CARRY :ROCKET
SETCOLOR :BLACK
```

```
TO SHAPES
TELL SPRITE 13
HOME
CARRY 1
SETCOLOR :RED
WAIT 30
CARRY SHAPE + 1
END
```

CHARNUM

CHARNUM *character*

CHARNUM is an operation that returns the character code number of the given character. Only one character should follow the CHARNUM operation because the computer looks at the first character as if it is the only character. CHARNUM can be used with any command or operation that requires a number.

In the first example, the computer returns the character code number for S and prints 83 on the display.

The SUBTRACT procedure takes the two variables that are input and subtracts their values. The difference is then printed on the display.

Examples:

```
PRINT CHARNUM "S
```

```
TO SUBTRACT ONE TWO
CALL DIFFERENCE CHARNUM
:ONE CHARNUM :TWO "X
PRINT :X
END
```

TI LOGO REFERENCE GUIDE

CLEARSCREEN

CLEARSCREEN

Short form: **CS**

The CLEARSCREEN command clears the display of words and Turtle drawings; however, CLEARSCREEN does not erase sprites or change the color of the display. CLEARSCREEN can be included as a statement in a procedure and is a convenient command with which to begin a procedure. Then, when the procedure is performed, all of the words and Turtle lines are cleared from the display.

If CLEARSCREEN is used as the first statement in a Turtle procedure, the Turtle starts from HOME. If CLEARSCREEN is not included in the procedure, the Turtle starts where it came to rest as a result of previous commands.

In the example, the Turtle draws a box, turns LEFT, and then moves FORWARD 50 steps. The procedure then repeats with the CLEARSCREEN command clearing the display which sends the Turtle HOME each time.

In the CHANGECOLOR procedure, the color value changes each time SETCOLOR COLOR + 1 is repeated causing the plane to change colors four times.

Examples:

```
TO DESIGN
TELL TURTLE
CLEARSCREEN
REPEAT 4 [FORWARD 30
  RIGHT 90]
LEFT 10
FORWARD 50
DESIGN
END
```

```
TO CHANGECOLOR
TELL SPRITE 0
HOME
CARRY :PLANE
SETCOLOR :RUST
WAIT 20
REPEAT 4 [SETCOLOR
  COLOR + 1 WAIT 20]
END
```

TI LOGO REFERENCE GUIDE



COLOR

COLOR

The COLOR operation returns the number of the color of the active sprite, Turtle, or background. It does not give the color of characters on tiles.

See also: HEADING, SETCOLOR, SHAPE, SPEED

In the example, SETCOLOR 6 changes the color of the ink with which the Turtle is drawing from black to red. PRINT COLOR prints the number 6, which is the number for red, on the display.

COLOR also can be used to change the value of a sprite's color. In the second example, Sprite 1 carries the blue ball. The color becomes red when SETCOLOR COLOR + 2 is entered.

In this example, SETCOLOR 15 changes the color of the display. PRINT COLOR displays the number of the listener's color; the listener is the background.

In the CHANGECOLOR procedure, the color value changes each time SETCOLOR COLOR + 1 is repeated causing the plane to change colors four times.

Examples:

```
TELL TURTLE
SETHEADING 90
FORWARD 10
SETCOLOR 6
FORWARD 10
PRINT COLOR
```

```
TELL SPRITE 1
HOME
CARRY 4
SETCOLOR 4
SETCOLOR COLOR + 2
```

```
TELL BACKGROUND
SETCOLOR 15
PRINT COLOR
```

```
TO CHANGECOLOR
TELL SPRITE 0
HOME
CARRY :PLANE
SETCOLOR :RUST
WAIT 20
REPEAT 4 [SETCOLOR
          COLOR + 1 WAIT 20]
END
```

TI LOGO REFERENCE GUIDE

COLORBACKGROUND

COLORBACKGROUND *:word*
COLORBACKGROUND *number*

Examples:

Short form: **CB**

The command COLORBACKGROUND, followed by a word or number that represents one of the preset colors, changes the color of the display. This command does not change the current listener. If a sprite that is on the display is defined as having the same color as the display, it seems to disappear.

The color of the display also can be changed by using two commands: TELL BACKGROUND and SETCOLOR *:word* or *number*. Changing the color of the display with the TELL BACKGROUND command makes the background the current listener.

See also: BACKGROUND

In the first example, a blue ball is put in the middle of the display, and the background is changed from red to blue (the ball seems to disappear) and then to red again. The sprite then carries the box shape.

```
TO COLORS  
TELL SPRITE 4  
HOME  
CARRY :BALL  
SETCOLOR :BLUE  
COLORBACKGROUND :RED  
WAIT 50  
COLORBACKGROUND :BLUE  
CARRY :BOX  
END
```



CONTENTS

CONTENTS

CONTENTS is a command that prints on the display all of the predefined words that are given a value, plus the names of all of the procedures.

See also: PP, PN, PA

Assuming that you just turned on your computer, enter the DESIGN procedure. Next, enter PRINT CONTENTS and the following information is displayed.

```
DESIGN BOX BALL ROCKET TRUCK P
LANE WEST SOUTH EAST NORTH WHI
TE GRAY PURPLE OLIVE LEMON YEL
LOW ORANGE RUST CYAN RED SKY B
LUE LIME GREEN BLACK CLEAR ALL
TRUE FALSE
```

Examples:

```
TO DESIGN
TELL TURTLE
REPEAT 50 [RIGHT 20
  REPEAT 4 [FORWARD 30
    RIGHT 90]]
END

PRINT CONTENTS
```

CONTINUE

CONTINUE

Whenever you want the computer to pause *during a procedure*, press SHIFT A on a TI-99/4 console or FCTN 7 on the TI-99/4A console. CONTINUE is the command you enter to have the computer resume running a procedure after a pause. The procedure starts from the point at which it was paused.

When a procedure is paused, the word PAUSED appears, followed by the line and level number of any subprocedure at which the procedure paused. If any sprites are in motion, they continue to move even though the procedure stops. However, the Turtle does stop when the procedure is stopped.

See also: PAUSE

TI LOGO REFERENCE GUIDE

DEFINE

DEFINE "*procedure-name* [[] [*list*]]

DEFINE "*procedure-name* [*variable*] [*list*]]

Short form: **DE**

The **DEFINE** command lets you define a procedure without using **TO**. Defined procedures then can be incorporated into another procedure or used to create intricate designs on the display.

Almost any procedure that can be defined with **TO** also can be defined with **DEFINE**.

The list of commands in the procedure must be enclosed in brackets. If you are using one or more variables, they are listed in the first set of brackets and are separated by a space. The first list must be preceded by a set of empty brackets if no variable is being used.

See also: **TO**

Examples:

In the first example, the Turtle draws a box and then lifts up its pen and draws another box to the left.

```
DEFINE "BOX [[]][FD 30][RT
  90][FD 30] [RT 90][FD
  30][RT 90][FD 30]]
TO SQUARES
TELL TURTLE
BOX
PENUP
FORWARD 50
PENDOWN
BOX
END
```

In the second example, all the attributes for **Sprite 2** are listed within the brackets. To run the procedure, type **VAN** and press **ENTER**.

```
DEFINE "VAN [[]][TELL
  SPRITE 2 CARRY 2
  SETCOLOR :RED
  SETSPEED 45
  SETHEADING 90 HOME]]
```

If a variable (or variables) is used with the **DEFINE** command, it appears in the first set of brackets. Be sure a space appears after the variable name and before the **]**. More than one command can be listed in a set of square brackets, just as more than one command can be entered on a line in a procedure. In the **TRIANGLE** procedure, **SIDE** is the variable and appears in the first set of square brackets.

```
DEFINE "TRIANGLE [ [SIDE ]
  FORWARD :SIDE LEFT 120
  FORWARD :SIDE LEFT 120
  FORWARD :SIDE ] ]
TELL TURTLE
TRIANGLE 20
```



DIFFERENCE

DIFFERENCE *number number*

DIFFERENCE *:word :word*

*Short form: number - number
:word - :word*

DIFFERENCE is a subtraction operation that results in the numerical difference between two numbers, between two words that are assigned numerical values, or between a number and a word that are assigned a numerical value.

See also: PRODUCT, QUOTIENT, SUM

In the first example, the output is 15. In the second example, the difference between the values of red and blue is 2.

Examples:

```
PRINT DIFFERENCE 25 10
```

```
PRINT DIFFERENCE :RED  
:BLUE
```

DOT

DOT *x-coordinate y-coordinate*

DOT places a DOT on the display at specified x- and y-coordinates in the Turtle mode.

The negative y-coordinates must be enclosed in parentheses, not brackets. The first example causes a small dot to appear at the corresponding position.

DOT also can be incorporated into a procedure and mathematically manipulated to draw lines or place dots on the display at a variety of locations. The STRING procedure places a dot using the x and y variables you enter. If you enter STRING 36, a string of dots is placed diagonally across the display in rows.

Examples:

```
TELL TURTLE  
DOT 45 [- 45]
```

```
TO STRING N  
TELL TURTLE  
DOT :N :N  
STRING :N + 5  
END
```

TI LOGO REFERENCE GUIDE

EACH

EACH *[command]*

When you are talking to a group or list of sprites, the command **EACH** tells the computer to do something to each of the sprites in the order they appear in the list. If you are talking to **:ALL**, Sprite 0 is the first sprite to be told what to do.

EACH can be used with any command that gives a sprite attributes — **CARRY**, **SETCOLOR**, **SETSPEED**, **SETHEADING**, **SYV**, **SXV**, and **SV**. It also can be used with the **WAIT** command.

In the procedure **SCATTER**, you are talking to all of the sprites (0 through 31). Each sprite is given attributes by the **EACH** command.

Examples:

```
TO SCATTER
TELL :ALL
EACH [CARRY YOURNUMBER
      / 5]
EACH [SETSPEED
      YOURNUMBER * 3]
EACH [SETCOLOR
      YOURNUMBER]
EACH [SETHEADING
      YOURNUMBER * 11]
END
```

TI LOGO REFERENCE GUIDE



EDIT

EDIT *procedure-name*

EDIT tells the computer to enter the Edit mode so that changes can be made in the procedure. When you enter EDIT and the name of a procedure, the display turns green and the contents of the procedure are displayed. If the procedure is not defined yet, only the procedure name and END are displayed on the green screen. To change anything in the procedure, use the special key functions listed below.

See also: TO

<u>TI-99/4</u>	<u>TI-99/4A</u>	<u>Function</u>
SHIFT W (BEGIN)	FCTN 5 (BEGIN)	Moves the cursor to the beginning of the line.
SHIFT C (CLEAR)	FCTN 4 (CLEAR)	Erases everything in the line that is above and to the right of the cursor.
SHIFT F (DELETE)	FCTN 1 (DELETE)	Erases the one character or space directly above the cursor.
SHIFT V (PROC'D)	FCTN 6 (PROC'D)	Moves the cursor to the end of the line.
SHIFT E (↑)	FCTN E (↑)	Moves the cursor up one line.
SHIFT S (←)	FCTN S (←)	Moves the cursor to the left one space.
SHIFT D (→)	FCTN D (→)	Moves the cursor to the right one space.
SHIFT X (↓)	FCTN X (↓)	Moves the cursor down one line.
SHIFT Z (BACK)	FCTN 9 (BACK)	Leaves the EDIT mode.

TI LOGO REFERENCE GUIDE

EITHER

EITHER *expression expression*

EITHER is an operation that compares two numeric or relational expressions. If either expression is evaluated as true, the statement is true. If both of the expressions are evaluated as false, the statement is false. The true or false value returned by EITHER can then be printed on the display or used with conditional statements.

See also: BOTH, IFF, IFT, IF THEN ELSE, TEST, TYPE, PRINT

The relational expressions allowed are:

- = Equal
- > Greater than
- < Less than

In the first example on the right, EITHER returns a value of TRUE since both expressions are true. In the second example, the statement is TRUE since the second expression is correct. In the third example, however, since both expressions are incorrect, EITHER returns a value of FALSE.

The TESTSPEED procedure prints CORRECT SPEED! since the speed of the sprite is more than 9, even though the speed of the sprite is not more than 11. If you change the speed to 9 or less, TESTSPEED prints the response WRONG SPEED!

END

END

END is the command that defines the end of a procedure. It *always* must be the last command in any procedure and on a line by itself. If it is omitted, a > appears on the display after BACK is pressed. To correct this, press BACK and reenter procedure including the END command.

In the BOX procedure, END indicates the end of the procedure.

Examples:

```
PRINT EITHER 5 + 3 = 8
      4 + 2 = 6
PRINT EITHER 5 + 1 = 4
      4 + 2 = 6
PRINT EITHER 5 + 1 = 4
      4 + 2 = 1
```

```
TO TESTSPEED
TELL SPRITE 1
HOME
CARRY 4
SETCOLOR :RED
SETSPEED 10
TEST EITHER SPEED > 9 SPEED >
  11
IFT PRINT [CORRECT SPEED!]
IFF PRINT [WRONG SPEED!]
END
```

Examples:

```
TO BOX
TELL TURTLE
REPEAT 4 [FORWARD 30
  RIGHT 90]
END
```



ERASE

ERASE *procedure name*

ERASE is the command that allows you to erase a specific procedure from the computer's memory. To do so, enter ERASE and the name of the procedure you want to erase; this removes the procedure from the computer's workspace.

FIRST

FIRST [*list*]

FIRST "*name*

FIRST *:word*

Short form: **F**

The operation FIRST requires one input, a list, name, or word. If the input is a word or a name, FIRST returns the first character. If the input is a list, FIRST returns the first item in the list. Using FIRST more than once lets you access the first item in a list and then use the first item of the rest of the list in a different statement.

See also: BUTFIRST, BUTLAST, LAST.

Note: If FIRST is used with any of the above three operations, the computer performs the operations from right to left.

In these examples, the FIRST character of the name or the first item in a list is printed on the display.

In the procedure FINDNAMES, FIRST returns HE, the first item of the list [HE SHE IT]. The BUTFIRST of the list is SHE IT and the FIRST of that list is SHE.

Examples:

```
PRINT FIRST "ELEPHANT
PRINT FIRST "248
PRINT FIRST [SCOTT BILL
             JERRY]
```

```
TO FINDNAMES
PRINT FIRST [HE SHE IT]
PRINT FIRST BUTFIRST [HE
                     SHE IT]
END
```

TI LOGO REFERENCE GUIDE

FORWARD

FORWARD *number*

Short form: **FD**

FORWARD is one of four direction commands that can be given to the Turtle or a sprite. It must be followed by a number specifying how many steps the Turtle or sprite is to move.

Utilizing the BOX, 3D demonstrates the usage of FORWARD. The procedure draws a three dimensional box.

Examples:

```
TO 3D
BOX
RIGHT 45
FORWARD 30
LEFT 45
BOX
LEFT 135
FORWARD 30
LEFT 135
FORWARD 30
LEFT 45
FORWARD 30
LEFT 45
FORWARD 30
LEFT 135
FORWARD 30
RIGHT 45
FORWARD 30
RIGHT 135
FORWARD 30
END

TO BOX
TELL TURTLE
REPEAT 4 [FORWARD 30
  RIGHT 90]
END
```



FPUT

FPUT *item [list]*

FPUT inserts an item as the first item of a list. The item can be a name, a word, or a numeric value. FPUT is especially valuable in combining words and phrases in a CALL or PRINT statement.

See also: LPUT.

In the example, HELLO is inserted before MY FRIEND and is printed on the display.

Examples:

```
TO D
CALL "HELLO "X
CALL FPUT :X [MY FRIEND]
  "Y
PRINT :Y
END
```

FREEZE

FREEZE

The command FREEZE stops the motion of *all* sprites. All of the sprites remain motionless until the command THAW is entered.

See also: THAW

The MOVE procedure starts a green rocket moving across the display. After several seconds, all the sprites FREEZE.

Examples:

```
TO MOVE
TELL :ALL
HOME
EACH [CARRY :ROCKET
  SETCOLOR :GREEN
  SETSPEED 40
  SETHEADING
  YOURNUMBER * 20]
WAIT 50
FREEZE
END
```

TI LOGO REFERENCE GUIDE

GO

GO *"label*

GO is a command that sends the computer back to a specific line in the procedure which is identified by a label. Everything between the label and the GO statement is executed by the computer.

Note that when you use a label with a GO command, the label must be preceded by quotes. Also, when you list the label in the procedure, it must be followed immediately by dots. No space is typed between the last letter of the label name and the dots, but a space is typed before the command on that line. Also, it is not necessary to have anything following the label on the line.

For example, if you want to put a truck on the display and make it appear to accelerate smoothly, GO helps you achieve that type of animation without writing a subprocedure. In the procedure ACCELERATE, the red truck continues to increase its speed every time the computer goes to the HERE line and repeats everything between that line and GO.

Examples:

```
TO ACCELERATE
TELL SPRITE 2
HOME
CARRY :TRUCK
SETCOLOR :RED
SETHEADING 90
SETSPEED 0
HERE: SETSPEED SPEED + 5
WAIT 15
GO "HERE
END
```

GREATER

GREATER *:word :word*

GREATER *number number*

Short form: :word > :word
number > number

GREATER is an operation that compares two numbers or words that are assigned numeric values. If the first number is greater than the second, the output is TRUE; the output is FALSE if the second number is greater than the first.

See also: BOTH, EITHER, LESS

In the first two examples, the output is TRUE for the first and FALSE for the second.

The BIG procedure tests two inputs to determine which is greater. If the first input is greater, the computer prints YES. If the second number is greater, LITTLE is printed. If you enter BIG 27 83, LITTLE is printed on the display.

Examples:

```
PRINT GREATER 10 6
PRINT GREATER :LEMON
:PURPLE

TO BIG A B
TEST GREATER :A :B
IFT PRINT "YES
IFF PRINT "LITTLE
END
```



HEADING

HEADING

HEADING is an operation that returns the current heading or direction of the present listener.

See also: COLOR, SETHEADING, SHAPE, SPEED

The first example gives Sprite 1 the heading of 135. PRINT HEADING causes the computer to print 135.

The value of HEADING can be used with other commands. The Turtle and all of the sprites have a heading of zero when the computer is turned on. In the procedure OCTAGON, the Turtle stops drawing when its heading equals zero.

Examples:

```
TELL SPRITE 1
SETHEADING 135
PRINT HEADING
```

```
TO OCTAGON
TELL TURTLE
FORWARD 20
RIGHT 40
IF HEADING = 0 STOP
OCTAGON
END
```

HIDETURTLE

HIDETURTLE

Short form: **HT**

HIDETURTLE makes the Turtle disappear but still allows it to draw. After entering HIDETURTLE, the small triangle is not visible. To make the Turtle reappear, simply type SHOWTURTLE. *Note:* The Turtle draws a design faster if the HIDETURTLE command is used.

See also: NOTURTLE, SHOWTURTLE

In the example, the Turtle draws a circle, hides, and then draws a smaller circle inside the first circle.

Examples:

```
TO HIDE
TELL TURTLE
REPEAT 180 [RIGHT 5
  FORWARD 3]
HIDETURTLE
PENUP
RIGHT 90
FORWARD 10
LEFT 90
PENDOWN
REPEAT 120 [FORWARD 1
  RIGHT 3]
END
```

TI LOGO REFERENCE GUIDE

HOME

HOME

HOME is the center of the display which is the x- and y-coordinates 0 0. In the Turtle mode, entering HOME brings the Turtle back to the middle of the display with a heading of 0 and without drawing a line. Any words or lines drawn on the display remain there.

In the first procedure, the Turtle draws a Y and then disappears. Entering HOME in the Sprite mode causes the sprite that is the current listener to appear in the center of the display.

When the HALT procedure is run, a black truck travels from left to right starting from the HOME position. The second HOME command causes the rocket to appear in the center of the display.

Examples:

```
TO Y
TELL TURTLE
FORWARD 30
RIGHT 45
FORWARD 30
HOME
FORWARD 30
LEFT 45
FORWARD 30
HIDETURTLE
END
```

```
TO HALT
TELL SPRITE 2
HOME
CARRY :TRUCK
SETHEADING 90
SETCOLOR :BLACK
SETSPEED 50
TELL SPRITE 3
HOME
CARRY :ROCKET
SETCOLOR :RED
END
```



IF...THEN...ELSE

IF *expression* **THEN** *action* **ELSE** *action*

The IF...THEN...ELSE command gives the computer the ability to evaluate a numeric or relational expression and then act according to whether the expression is true or false. IF a relationship is true, THEN the computer carries out a specified action. IF a relationship is false, it performs the ELSE portion of the statement or goes on to the next statement if ELSE is omitted. Both THEN and ELSE are optional.

See also: IFF, IFT, TEST

In the example on the right, the background first turns white and then turns blue because 15 is the value for the color white. Notice that on line 2, the word THEN is included, but it is omitted on line 3. The command THEN is assumed by the computer.

ELSE provides an alternative action if the expression is evaluated as false. TRUE is the output of the first IF statement and FALSE is the output of the last statement.

Examples:

```
COLORBACKGROUND 15
IF 15 = :RED THEN
  COLORBACKGROUND :RED
IF 15 = :WHITE
  COLORBACKGROUND
  :BLUE

IF 5 + 5 = 12 THEN PRINT
  "TRUE ELSE PRINT "FALSE
IF 5 + 5 < 12 PRINT "TRUE
  ELSE PRINT "FALSE
```

TI LOGO REFERENCE GUIDE

IFF/IFT

IFF *action*

IFT *action*

IFF (IF False) and IFT (IF True) are the two commands used with TEST to determine the action to be taken when a condition is true or false. These three commands perform the same function as the IF...THEN...ELSE command. *Note:* Each of these commands is entered on separate statement lines.

See also: IF...THEN...ELSE, IS, TEST

Any number of IFT and IFF statements can be in a single test. Each statement must have IFT or IFF at the beginning of the line following TEST.

These two procedures work together to cause a red square to appear to be "eating" a blue ball. The CREATE procedure establishes the attributes of the sprites. CHECK looks to see if the x-coordinate of the box is greater than 62. If it isn't, CHECK continues to check XCOR. As soon as the x-coordinate is greater than 62, or the check is true, the red box stops moving and the blue ball "disappears."

In the next procedure, TEST is used to test the larger of two numbers. Enter FIND 1234 4321. The computer then tests the two numbers to determine if the first is greater than the second. 1234 is not greater than 4321; therefore, the condition is false and the computer prints the value of Y, 4321. If you enter FIND 4321 1234, the computer prints the value of X. When you are testing two numbers, the second of which is a negative number, parentheses must appear around the negative number.

Examples:

```
TO CREATE
TELL SPRITE 2
HOME
CARRY :BALL
SETCOLOR :BLUE
SETSPEED 0
SX 70
TELL SPRITE 1
HOME
CARRY :BOX
SETCOLOR :RED
SETSPEED 0
SX -70
SETHEADING 90
SETSPEED 10
CHECK
END

TO CHECK
TEST XCOR > 62
IFT TELL SPRITE 2 SETCOLOR
: CLEAR
IFT TELL SPRITE 1 SETSPEED
0
IFF CHECK
END

TO FIND X Y
TEST GREATER :X :Y
IFT PRINT :X
IFF PRINT :Y
END
```

TI LOGO REFERENCE GUIDE



IS

IS *item item*

IS is a prefix operation that compares two items to see if they are equal. The two items can be characters, values, names, words, lists, expressions, or any combination of these. If the two items are equal, the computer returns true. If the two items are not equal, the computer returns false.

See also: BOTH, EITHER, GREATER, IFF, IFT, IF...THEN...ELSE, LESS, TEST

The output for the first two examples is TRUE and the output for the second two is FALSE.

Within a procedure, IS is generally used with TEST. In the procedure QUIZ, you must input the day of the week. If the input is the word MONDAY, the computer prints HAPPY MONDAY!.

Examples:

```
PRINT IS 4 + 2 6
PRINT IS "ME "ME
PRINT IS [7] 7
PRINT IS 8 "8
```

```
TO QUIZ "DAY
PRINT [WHAT DAY IS
      TODAY?]
CALL READLINE "DAY
TEST IS :DAY "MONDAY
IFT PRINT [HAPPY
          MONDAY!]
END
```

TI LOGO REFERENCE GUIDE

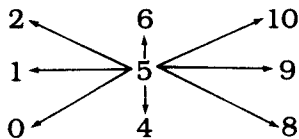
JOY

JOY 1 or JOY 2

JOY 1 or JOY 2 are joystick operations that can be used with the Turtle or with sprites. When used in conjunction with FORWARD, BACK, RIGHT, LEFT, and other operations that require numerical inputs, JOY 1 or 2 returns the numerical value of the joystick position.

Note: The Wired Remote Controllers (joysticks) must be attached to your system in order for you to use the JOY 1 and JOY 2 operations. JOY 1 represents one of the two joysticks and JOY 2 represents the other. In addition, if you are using a TI-99/4A computer console, the ALPHA LOCK key must be in the UP position.

Each joystick has nine positions, as illustrated in the graph below. Press the lever in the direction you want the Turtle or a sprite to move.



The JOYSTICK procedure demonstrates the use of JOY 1 and JOY 2. The computer returns to the number of the position you move the joystick and then prints that number on the display. *Note:* The output your joysticks return may be different from the above graph.

In the SPRITEJOY procedure, attributes are assigned to Sprite 1. The DIRECT procedure sets up conditionals for the joystick. When you move the joystick to a different position, the corresponding heading is given to the sprite. If the joystick is moved to position 5, the sprite freezes and then the procedure repeats itself.

Examples:

```
TO JOYSTICK S
TYPE JOY :S
WAIT 20
JOYSTICK :S
END
```

```
TO SPRITEJOY S
TELL SPRITE 1
HOME
CARRY :BALL
SETCOLOR :RED
SETSPEED 30
THAW
LABEL: DIRECT
IF :S = 5 GO "LABEL
SPRITEJOY :S
END
```

```
TO DIRECT
MAKE "X JOY :S
IF :X = 0 SETHEADING 225
IF :X = 1 SETHEADING 270
IF :X = 2 SETHEADING 315
IF :X = 4 SETHEADING 180
IF :X = 5 FREEZE
IF :X = 6 SETHEADING 0
IF :X = 8 SETHEADING 135
IF :X = 9 SETHEADING 90
IF :X = 10 SETHEADING 45
JOYSPRITE
END
```



LAST

LAST [*list*]

LAST "*name*

LAST *:word*

LAST is an operation that returns all but the last item of a list or all but the last character in a name or word. The value of a name or word cannot be numeric. LAST can be used more than once in a statement and also can be used with other operations.

See also: BUTFIRST, BUTLAST, FIRST

The first two examples print RAT and S.

LAST allows you to manipulate words and lists. The example on the right shows a way to determine if the last character in an input word is the letter S. If so, the computer prints PLURAL on the display; otherwise, the computer prints SINGULAR.

The NEWSPRITE procedure tells the computer to tell the LAST sprite in the list of sprites to CARRY a BALL. The PRINT WHO line identifies the current listener by printing 4, the last sprite in the list. A purple ball appears in the center of the display.

LAST also can be used with a BUTLAST command. The output for this statement is A. The operations in this statement are performed from right to left.

Examples:

```
PRINT LAST [DOG CAT RAT]
PRINT LAST
  "HIPPOPOTAMUS
```

```
TO SPELL "WORD
IF LAST :WORD = "S PRINT
  "PLURAL ELSE PRINT
  "SINGULAR
END
```

```
TO NEWSPRITE
TELL LAST [ 1 2 3 4 ]
CARRY :BALL
HOME
SETCOLOR :PURPLE
PRINT WHO
END
```

```
PRINT LAST BUTLAST
  "RAILROAD
```

TI LOGO REFERENCE GUIDE

LEFT

LEFT *number*

Short form: **LT**

LEFT is one of the four direction commands that can be given to the Turtle or a sprite. **LEFT** must be followed by the number of steps or degrees to turn.

In the first example, the Turtle draws a triangle.

In the second example, **LEFT** is used to make a sprite move around the display in a circle. A red ball starts at the center of the display, makes a loop around the center, and then moves diagonally across the display.

Examples:

```
TELL TURTLE
LEFT 45
FORWARD 45
LEFT 90
FORWARD 45
LEFT 135
FORWARD 65
```

```
TO ROUND
TELL SPRITE 4
HOME
CARRY :BALL
SETCOLOR :RED
SETSPEED 5
REPEAT 90 [LEFT 4 WAIT
  10]
END
```



LESS

LESS *:word :word*

LESS *value value*

LESS is an operation that compares two values or words that are assigned values. If the first value is less than the second, the statement is true; otherwise, the statement is false. When the PRINT command precedes LESS, the value of the operation is printed on the display. If the second value is a negative number, it must be enclosed by parentheses.

See also: BOTH, EITHER, GREATER, IF...THEN...ELSE, IFF, IFT, TEST

In the first example, the output is FALSE. In the next two examples, the output is TRUE.

The FIND procedure gives both sprites a different speed and assigns a name to the YCOR of each sprite. The procedure prints either TRUE or FALSE, depending upon whether or not :3Y is less than :4Y.

Examples:

```
PRINT LESS 7 4
PRINT LESS :RED :WHITE
PRINT LESS - 6 11

TO FIND
TELL [3 4]
HOME
CARRY :PLANE
SETCOLOR :PURPLE
SETHEADING 45
TELL SPRITE 3
CALL YCOR "3Y
SETSPEED 10
TELL SPRITE 4
CALL YCOR "4Y
SETSPEED 15
PRINT LESS :3Y :4Y
END
```

TI LOGO REFERENCE GUIDE

LOOKLIKE

LOOKLIKE *:word*

LOOKLIKE *number*

LOOKLIKE is one of two commands used to define the shape of a sprite. When using LOOKLIKE, you first must talk to a particular sprite and then tell it what shape to look like.

See also: CARRY

In the example, a sprite is first told to LOOKLIKE a plane and then proceed through all of the predefined shapes. In this procedure, a black plane appears and starts moving slowly towards the top of the display. It then carries a truck, a rocket, a ball, and finally a box. Then, the procedure repeats itself.

Examples:

```
TO CHANGE
TELL SPRITE 1
LOOKLIKE :PLANE
SETCOLOR :BLACK
SETSPEED 5
WAIT 30
LOOKLIKE 2 WAIT 30
LOOKLIKE 3 WAIT 30
LOOKLIKE 4 WAIT 30
LOOKLIKE 5 WAIT 30
CHANGE
END
```

LPUT

LPUT *item [list]*

LPUT inserts an item as the last item of a list. The item can be a name, a word, or a numeric value. LPUT is especially valuable in combining words and phrases in a CALL or PRINT statement.

See also: FPUT

In the example, DAY is inserted at the end of the list, IT'S A SUNNY.

Examples:

```
TO D
CALL "DAY "X
PRINT LPUT :X [IT'S A
  SUNNY]
END
```



MAKE

MAKE "name "name

MAKE "name value

MAKE "name [list]

MAKE "name :word

The MAKE command allows you to assign a variable name to a name, a value, a list of characters, or a word that is assigned a value. MAKE takes two inputs: the first is the variable name, and the second is the name, value, list, or word that is being assigned.

See also: CALL, READCHAR, READLINE

In the first example, the name FIRST is given the value of 4 and the name SECOND is given the value of 12. The name SUM is then given the combined values of FIRST and SECOND. The PRINT command tells the computer to print the value of SUM which is 16.

In this procedure, the Turtle moves forward 4 (JOHN), left 12 (ED), and then forward 16 (CHARLIE).

Examples:

```
MAKE "FIRST 4
MAKE "SECOND 12
MAKE "SUM :FIRST +
  :SECOND
PRINT :SUM

TO M
MAKE "JOHN 4
MAKE "ED 12
MAKE "CHARLIE :JOHN +
  :ED
TELL TURTLE
FORWARD :JOHN
LEFT :ED
FORWARD :CHARLIE
END
M
```

TI LOGO REFERENCE GUIDE

MAKECHAR

MAKECHAR *number*

MAKECHAR is a command that allows you to define your own set of 256 unique characters on the display. Each character that is to be defined must be given a number. If you select a number for a character that the computer already knows — the basic 64 ASCII (American Standard Code for Information Interchange) characters that define punctuation, number, and alphabet keys — the shape of that character appears on a grid. These characters are numbered 32 through 95. You can change the shape of the character to whatever shape you decide by using the special keys available in the MAKECHAR mode. Thus, it is possible to create your own code so that when you type the conventional alphabet, entirely different characters appear.

See also: CHARNUM, PRINTCHAR

If you enter MAKECHAR 65, the display turns green and an 8x8 grid appears with the capital letter "A" in the middle of it. While holding the SHIFT key on the TI-99/4 computer console or the FCTN key on the TI-99/4A computer console, press the arrow keys to blacken squares and create a design. Pressing the arrow keys without holding the SHIFT or FCTN key moves the cursor without filling an empty square or erases a square if it was already colored. Press BACK when you finish designing a character.

As another example, enter MAKECHAR 230. A blank 8X8 grid appears in the upper left-hand corner of the display. Create your own character to be identified by the number 230. The recursive procedure Z causes the computer to fill the display with the character you define as MAKECHAR 230.

Examples:

MAKECHAR 65

MAKECHAR 230

TO Z
PRINTCHAR 230
Z
END



MAKESHAPE

MAKESHAPE *number*

The MAKESHAPE command lets you design shapes for a sprite or a number of sprites to carry. There are 26 grids, numbered 0 through 25, on which you can draw designs.

See also: MAKECHAR

MAKESHAPE grids 1, 2, 3, 4, and 5 are the five predesigned shapes in TI LOGO: a plane, a truck, a rocket, a ball, and a box. However, these shapes can be redesigned. When the computer is turned off and then on, the original shapes return.

Enter the word MAKESHAPE followed by a number. A 16X16 grid appears in the upper left-hand corner of the green display. Hold down the SHIFT key on the TI-99/4 computer console or the FCTN key on the TI-99/4A computer console and press the appropriate arrow key to move the cursor from one square to another to create a design. If SHIFT or FCTN and an arrow key are pressed together, the squares are filled in. If you do not want to color a square or want to erase a square that is colored, only use the arrow keys.

When you finish your design, press BACK to leave the MAKESHAPE mode. Then tell any sprite to carry shape 17, and give the sprite attributes of color, speed, and heading.

Shapes can be joined together to create a design. However, no more than four sprites can be on one horizontal line. Any additional sprites disappear. You are not limited to the number of sprites that can be joined vertically.

Examples:

MAKESHAPE 17

```
TELL SPRITE 1
HOME
CARRY 17
SETCOLOR :BLUE
SETHEADING 90
SETSPEED 40
```

TI LOGO REFERENCE GUIDE

NOBEEP

NOBEEP

NOBEEP turns off the tone started by the command BEEP.

See also: BEEP

The `STREET` procedure utilizes the `ROAD` procedure to demonstrate a purple truck driving over a bumpy road. The `BEEP` and `NOBEEP` sequence creates the sound of driving over the bumps.

Examples:

```
TO STREET
ROAD
TELL SPRITE 7
SY 11
SX -100
CARRY :TRUCK
SETCOLOR :PURPLE
SETHEADING 90
SETSPEED 20
REPEAT 90 [BEEP WAIT 2
           NOBEEP WAIT 2]
END
```

```
TO ROAD
TELL TURTLE
CLEARSCREEN
HIDETURTLE
RIGHT 120
REPEAT 48 [LEFT 60
           FORWARD 3 RIGHT 60
           FORWARD 3]
END
```



NOT

NOT *expression*

The NOT operation evaluates a numeric or relational expression. If the expression is incorrect, NOT returns a value of TRUE; if the expression is correct, NOT returns FALSE.

See also: BOTH, EITHER, GREATER, IF...THEN...ELSE, IFF, IFT, LESS, TEST

The first statement on the right prints TRUE on the display since 5 is *not* less than 2. The second statement prints FALSE.

The MOVE procedure puts a black truck on the display. If the speed of the truck is not 30, the computer automatically changes the speed to 30.

Examples:

```
PRINT NOT 5 < 2  
PRINT NOT 5 > 2
```

```
TO MOVE  
TELL SPRITE 2  
HOME  
CARRY 2  
SETCOLOR 1  
SETHEADING 90  
SETSPEED 10  
WAIT 100  
IF NOT SPEED = 30 THEN  
    SETSPEED 30  
END
```

NOTURTLE

NOTURTLE

NOTURTLE is the command used to leave the Turtle mode. It erases everything from the display that the Turtle draws, along with any words that may be there. It does not, however, erase any sprites with the Turtle. Any procedure written in the Turtle mode remains available in the workspace until the system is turned off or the procedure is erased.

When NOTURTLE is entered, the cursor appears in the top left-hand corner of the display.

TI LOGO REFERENCE GUIDE

NUMBER?

NUMBER? *"name*

NUMBER? *number*

NUMBER? *[list]*

NUMBER? is an operation that tests an item to determine if it is a number. NUMBER? returns true for a number and false for a word or a list, even if integers are included in the word or list. This operation is useful in checking the status of variables in lengthy or complex procedures.

See also: THING?, WORD?

The adjacent examples demonstrate NUMBER? using lists, words, and numbers. The first statement returns a value of TRUE; the others print FALSE on the display.

If a numeric value is assigned to a variable name, the name is an acceptable input for NUMBER?. The example on the right prints TRUE on the display.

In the SEE procedure, if "87 is a number, the background color changes to white; otherwise, it changes to red. Because "87 is defined as a name in the CALL command, the answer is FALSE and the display turns red.

Examples:

```
PRINT NUMBER? 68
PRINT NUMBER? [22]
PRINT NUMBER? "DOG
PRINT NUMBER? "129
PRINT NUMBER? [1 2 3]
```

```
CALL 17 "SEVENTEEN
PRINT NUMBER?
:SEVENTEEN
```

```
TO SEE
CALL "87 "X
IF NUMBER? :X
  COLORBACKGROUND 15
ELSE COLORBACK-
GROUND 6
END
```

NUMBEROF

NUMBEROF *attribute*

NUMBEROF outputs the value of an attribute of the Turtle, a sprite, a tile, and the background. The attributes which can be tested are SPEED, HEADING, XCOR, YCOR, COLOR, WHO, SHAPE, XVEL, and YVEL. *Note:* The NUMBEROF command operates with a single sprite but not a list or :ALL.

In the example, Sprite 10 is given attributes. When each PRINT NUMBEROF statement is entered, the computer outputs the corresponding information.

Examples:

```
TELL SPRITE 10
HOME
CARRY :ROCKET
SETCOLOR :RED
SETHEADING 40
SETSPEED 10
PRINT NUMBEROF WHO
PRINT NUMBEROF COLOR
PRINT NUMBEROF HEADING
PRINT NUMBEROF YCOR
```



OUTPUT

OUTPUT *anything*

OUTPUT is a command that allows you to define a subprocedure which returns a value to the line and level from which the value is called.

In the example, the scores for three bowling games, A, B, and C, are added together and then printed. Then, the bowler's handicap, X, is added to the total by the TOTAL procedure. (Note: A, B, C, D, and X are variables and must be replaced with values when you run the SCORE procedure.)

You also can use OUTPUT to define your own operations. An operation can be as simple as your own short forms for primitive operations. (Any primitive or procedure that gives or "outputs" information is an operation.) The T procedure redefines "TURTLE" as "T"; therefore, TELL T tells the Turtle to listen.

In the X procedure, the output is the listener's x-coordinate.

The procedure at the right takes two inputs, an item, and a list. It checks to see if the item is a member of the list. Then the operation outputs TRUE or FALSE accordingly.

Examples:

```
TO SCORE A B C X
CALL :A + :B + :C "D
PRINT :D
PRINT TOTAL
END
```

```
TO TOTAL
OUTPUT :D + :X
END
```

(Press BACK)

SCORE 126 145 139 21

```
TO T
OUTPUT TURTLE
END
```

(Press BACK)

```
TELL T
```

```
TO X
OUTPUT XCOR
END
```

```
TELL SPRITE 1
PRINT X
```

```
TO MEMBER? "ITEM "LIST
LABEL: IF :LIST = [] OUTPUT
"FALSE
TEST :ITEM = FIRST :LIST
IFT OUTPUT "TRUE
IFF MAKE "LIST BUTFIRST
:LIST
GO "LABEL
END
```

```
PRINT MEMBER? "DOG [CAT
MOUSE ELEPHANT]
```

```
PRINT MEMBER? "CAT [CAT
MOUSE ELEPHANT]
```

TI LOGO REFERENCE GUIDE

PA

PA

PA is the command used to print all of the procedures and the predefined names in the computer's memory. When PA is entered, the computer first prints the contents of all procedures in memory. It then prints the names and numeric codes for the predefined shapes, directions, and colors.

PENDOWN

PENDOWN

Short form: **PD**

PENDOWN is one of the *states of the pen* of the Turtle. It tells the Turtle to place its pen down and to be ready to draw.

PENDOWN is the normal, or default, position for the Turtle's pen when you enter the Turtle mode. To draw a line, the Turtle must be told to move FORWARD or BACK a number of steps. A line appears on the display.

Note: If one of the other *states of the pen* is used, PENDOWN must be entered for the Turtle to return to its default position to draw.

See also: PENERASE, PENREVERSE, PENUP

The ROAD procedure tells the Turtle to draw a road across the display.

Examples:

```
TO ROAD
TELL TURTLE
CLEARSCREEN
RIGHT 90
FORWARD 241
RIGHT 90
PENUP
FORWARD 30
LEFT 90
PENDOWN
FORWARD 241
HIDETURTLE
END
```



PENERASE

PENERASE

Short form: **PE**

PENERASE is one of the *states of the pen* of the Turtle. It tells the Turtle to erase any lines it passes over. It is an easy way to erase part of a Turtle drawing without clearing the display and erasing the whole design.

See also: PENDOWN, PENREVERSE, PENUP

The procedure on the right causes the Turtle to "bounce" up and down in the center of the display.

Examples:

```
TO BOUNCE
TELL TURTLE
CLEARSCREEN
REPEAT 10 [FORWARD 90
  PENERASE BACK 90
  PENDOWN]
END
```

PENREVERSE

PENREVERSE

Short form: **PR**

PENREVERSE is one of the *states of the pen* of the Turtle; however, it is a unique command in that, as the Turtle travels along, it erases any line it travels over and draws a new line where none exists.

See also: PENDOWN, PENERASE, PENUP

The procedure BOX demonstrates a dramatic way to use PENREVERSE to create designs.

Examples:

```
TO BOX
REPEAT 4 [FORWARD 30
  RIGHT 90]
RIGHT 10
PENREVERSE
BOX
END
```

TI LOGO REFERENCE GUIDE

PENUP

PENUP

Short form: **PU**

PENUP is one of the *states of the pen* of the Turtle. It tells the Turtle to pick up its pen so it can be moved around the display without drawing a line.

See also: PENERASE, PENDOWN, PENREVERSE

The DASH procedure demonstrates another use of the PENUP command. This procedure draws a line of dashes on the display. Reading through the procedure, note that the computer is told to repeat a sequence 12 times: PENDOWN, FORWARD 10, PENUP, and FORWARD 10. (This procedure leaves the Turtle in a PENUP state.)

Examples:

```
TO DASH
TELL TURTLE
RIGHT 90
REPEAT 12 [PENDOWN
  FORWARD 10 PENUP
  FORWARD 10]
END
```

PN

PN

PN (Print Names) is the command used to print all of the names and their numeric values in memory. The list includes the names of all of the shapes, directions, and colors in the computer's memory, plus any names that are assigned a value.

See also: PA, PO, PP



PO

PO *procedure-name*

Examples:

The command PO (Print Out) followed by a procedure name instructs the computer to display the procedure on the display.

See also: PA, PN, PP

For example, if you enter PO ROAD and a procedure named ROAD is currently in the computer's memory, that procedure is displayed.

PO ROAD

PP

PP

The command PP (Print Procedures) instructs the computer to print on the display all of the procedures in the computer's memory.

See also: PA, PN, PO

TI LOGO REFERENCE GUIDE

PRINT

PRINT *[list]*

PRINT *number*

PRINT *numeric expression*

PRINT *:word*

PRINT *"name*

The PRINT command tells the computer to print a word, an ASCII character, a symbol, a list, a value, or the results of a numeric expression. When the computer completes printing the output, the cursor is placed at the beginning of the next line.

See also: FPUT, LPUT, SENTENCE, TYPE

To print a word, character, or symbol, you must include a quotation mark before the print item. The computer prints HELLO in the first example and A in the second example.

If a numeric expression is preceded by a quotation mark, the computer prints what follows the quotation mark. It does not perform the arithmetic operation. In the example on the right, the computer prints 5*5+5.

To print a list of words, numbers, or symbols, enclose the list in brackets.

Examples:

```
PRINT "HELLO  
PRINT "A
```

```
PRINT "5*5+5
```

```
PRINT [THIS IS HOW TO  
PRINT A LIST.]  
PRINT [1 2 3 4 5]
```

TI LOGO REFERENCE GUIDE



When numbers or the result of a mathematical expression are to be printed, quotation marks or brackets are not required. In these adjacent three examples, the outputs are 10234, 10, and 50.

The `SPRITESTATE` procedure is used to print information concerning the current listener. After giving Sprite 2 attributes of shape, color, heading, and speed, run the `SPRITESTATE` procedure.

`PRINT` can be used to print the value assigned to a variable name. The value of `:ORANGE` is 9, and the value of `:FRIEND` is FRED.

```
PRINT 10234
PRINT 5 + 5
PRINT 5*(5 + 5)
```

```
TO SPRITESTATE
PRINT [YOU ARE TALKING
      TO]
PRINT WHO
PRINT [THE COLOR IS]
PRINT COLOR
PRINT [THE SHAPE IS]
PRINT SHAPE
PRINT [THE SPEED IS]
PRINT SPEED
PRINT [THE HEADING IS]
PRINT HEADING
END
```

(Press BACK)

```
TELL SPRITE 2
CARRY :BOX
SETCOLOR :YELLOW
SETHEADING 45
SETSPEED 20
SPRITESTATE
```

```
PRINT :ORANGE
CALL "FRED "FRIEND
PRINT :FRIEND
```

TI LOGO REFERENCE GUIDE

PRINTCHAR

PRINTCHAR *item*

Examples:

Short form: **PC**

All of the letters and symbols that appear on the display are part of a code system called the American Standard Code for Information Interchange, or ASCII. The ASCII characters used in TI LOGO are numbered 32 to 95. The symbols used to create the title screen when the computer is turned on are numbered from 2 to 31. The Turtle draws designs on characters 2 to 31 and 96 to 255.

See also: CHARNUM, MAKECHAR

The computer can store 256 characters in memory, numbered 0 to 255. To print any one of those characters, use the command PRINTCHAR followed by a number, a numeric expression, or a variable which is assigned a value within this range. Each of the PRINTCHAR statements on the right prints the character A on the display.

```
PRINTCHAR 65
PRINTCHAR 30 + 35
PRINTCHAR 13 * 5
CALL 65 "LETTER
PRINTCHAR :LETTER
```

The computer does not make a carriage return after printing a character; the cursor remains on the same line immediately following the printed character.

PRODUCT

PRODUCT *:word :word*

Examples:

PRODUCT *number number*

The PRODUCT operation multiplies two numbers. The inputs can be integers or names that have numeric values.

See also: DIFFERENCE, QUOTIENT, SUM

When PRODUCT is used with a command, the computer is told to use the output for the desired action. With the PRINT command, the computer prints 10000.

```
PRINT PRODUCT 100 100
```

In the second example, the Turtle moves forward 63 steps.

```
TELL TURTLE
FORWARD PRODUCT 9 7
```

In the third example, the output is 80 and is printed on the display.

```
CALL 20 "TWENTY
CALL 4 "FOUR
PRINT PRODUCT :TWENTY
:FOUR
```



PUTTILE

PUTTILE *character-number column-number row-number*

Examples:

If you look at the descriptions of the MAKECHAR and PRINTCHAR commands, you see that it is possible to define up to 256 characters. Just as the computer knows five predesigned shapes, it also knows 65 predesigned characters which include letters, punctuation marks, numerical symbols, and other special symbols. Each of these characters is designed on an 8X8 grid.

See also: MAKECHAR, PRINTCHAR

The display contains 32 columns of tiles across and 24 rows of tiles down the display. Using the command PUTTILE you can place the character of your choice at any location within the 32X24 display. You must first give the number of the character and then the number of the column, followed by the number of the row. There is a space between each of these numbers but are no commas.

This example puts the letter B at HOME.

The procedure shows how to print several diagonal lines of C's across the display.

```
PUTTILE 66 16 12
```

```
TO LINE X Y  
MAKE "X :X - 1  
MAKE "Y :Y + 1  
PUTTILE 67 :X :Y  
LINE :X :Y  
END
```

```
LINE 12 8
```

QUOTIENT

QUOTIENT *number number*

Examples:

QUOTIENT *:word :word*

QUOTIENT is an operation that divides two numbers or values assigned to a word. The first value is the number to be divided and the second is the divisor. The result of the division is an integer; any remainder is truncated.

See also: DIFFERENCE, PRODUCT, SUM

When QUOTIENT is used with the PRINT command, the output (3 in these examples) is printed on the display.

```
PRINT QUOTIENT 33 11  
CALL 33 "A  
CALL 11 "B  
PRINT QUOTIENT :A :B
```

TI LOGO REFERENCE GUIDE

RANDOM

RANDOM

RANDOM is an operation that returns a randomly selected single-digit number from 0 to 9. RANDOM can be used with arithmetic operations to generate other numbers.

When RANDOM is used with the PRINT command alone, a random number appears every time you enter the command.

The CRAZY procedure tells a random sprite to carry a ball and then randomly sets the sprite's color, speed, and heading. The four PRINT statements output information concerning the active sprite. Each time you enter this procedure, the information changes. If you continue to enter CRAZY, more sprites appear on the display with different colors, speeds, and headings.

Examples:

```
PRINT RANDOM
```

```
TO CRAZY  
TELL RANDOM  
CARRY :BALL  
SETCOLOR RANDOM  
SETSPEED RANDOM * 4  
SETHEADING RANDOM * 5  
PRINT WHO  
PRINT COLOR  
PRINT SPEED  
PRINT HEADING  
END
```

RC?

RC?

RC? is an operation used with the command TEST or IF...THEN...ELSE to determine if a key has been pressed. Each time the computer checks to see if a key has been pressed, the output is true if a key has been pressed or false if it has not.

See also: NUMBER?, READCHAR, WORD?

In the NO procedure, the computer continues to print out DO NOT TOUCH THIS COMPUTER until any key is pressed. The message I TOLD YOU NOT TO TOUCH! is then displayed and the procedure is stopped.

Examples:

```
TO NO  
PRINT [DO NOT TOUCH  
THIS COMPUTER]  
TEST RC?  
IFT PRINT [I TOLD  
YOU NOT TO TOUCH!]  
STOP  
IFF NO  
END
```



READCHAR

READCHAR

Short form: RC

The READCHAR operation causes the computer to wait for a key to be pressed on the keyboard. READCHAR is generally used with the CALL or MAKE command to assign the input character to a variable name.

See also: RC?, READLINE

The CRAWL procedure moves the Turtle when F, B, T, or L is pressed.

Examples:

```
TO CRAWL
TELL TURTLE
CALL READCHAR "X
IF :X = "F FORWARD 5
IF :X = "B BACK 5
IF :X = "R RIGHT 5
IF :X = "L LEFT 5
CRAWL
END
```

READLINE

READLINE

The READLINE operation tells the computer to wait for a response to be entered from the keyboard. The response can be a single character or a string of characters including blanks. READLINE generally is used with CALL or MAKE to assign the input character(s) to a variable name.

See also: RC?, READCHAR

The procedure on the right illustrates the use of READLINE with CALL. When you run the procedure, the CALL READLINE "N statement stops the program and waits for your input, indicated by the prompting symbol (>). Then when you type a name and press ENTER, the name is assigned to "N and the procedure continues.

The next example illustrates the use of READLINE with the MAKE command.

Examples:

```
TO READ
PRINT [WHAT IS YOUR
NAME?]
CALL READLINE "N
PRINT "HELLO,
PRINT :N
END
```

```
TO STATE
PRINT [ANSWER TRUE OR
FALSE:]
PRINT []
PRINT [THE STATE OF OHIO]
PRINT [IS IN MAINE.]
MAKE "R READLINE
TEST :R = [FALSE]
IFT PRINT [VERY GOOD!]
STOP
IFF PRINT [TRY AGAIN]
STATE
END
```

TI LOGO REFERENCE GUIDE

RECALL

RECALL

To retrieve procedures, names, shapes, and tiles from a disk or a cassette tape, use the command **RECALL**.

After the command is entered, a menu appears on the display giving you the opportunity to select (1) **PROCEDURES**, (2) **SHAPES AND TILES**, or (3) **BOTH 1 AND 2**. After your selection, a second menu appears, asking if you want to recall from (1) **CASSETTE** or (2) **DISKETTE**.

If you are using a cassette recorder, be sure the recorder is correctly attached to your system. The computer prompts you through the operations required to retrieve information stored on the cassette tape. When the computer says to **REWIND**, position your tape to the location of the desired file. If you are using the TI Disk Memory System, you are asked for the filename you want to retrieve. Type the filename and press **ENTER**. To review the filenames on the disk, simply press the **SPACE BAR**. Each time you press the **SPACEBAR**, the next filename in alphabetical order is displayed. When the desired filename appears, press **ENTER**.



REPEAT

REPEAT *number* [*list*]

Rather than typing every step of a procedure, you can repeat a sequence of operations (enclosed in brackets) by using the REPEAT command. The number following REPEAT tells the computer how many times you want the sequence of operations to be repeated.

In the BOX procedure, REPEAT is used to shorten the process of telling the Turtle how to draw a box.

It also is possible to put one REPEAT command inside another, as shown in the DOUBLE procedure.

You also can use primitives or procedures within a REPEAT statement. The BOXES procedure utilizes the BOX procedure in the first example. Within the REPEAT statement, the Turtle turns RIGHT 10 degrees, draws a box, and then repeats the process 36 times.

Examples:

```
TO BOX
TELL TURTLE
REPEAT 4 [FORWARD 30
  RIGHT 90]
END
```

```
TO DOUBLE
TELL TURTLE
REPEAT 80 [RIGHT 10
  REPEAT 4 [FORWARD 30
    RIGHT 90]]
END
```

```
TO BOXES
TELL TURTLE
REPEAT 36 [RIGHT 10 BOX]
END
```

TI LOGO REFERENCE GUIDE

RIGHT

RIGHT *number*

Short form: **RT**

Right is one of the four directional commands you can give the Turtle or a sprite. It causes the Turtle or sprite to turn right a specified number of degrees. For example, **RIGHT 90** produces a right-angle turn.

This procedure tells the Turtle to draw an octagon on the display.

Examples:

```
TO OCTAGON  
TELL TURTLE  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
RIGHT 45  
FORWARD 30  
END
```



RUN

RUN *:word*

RUN [*list*]

RUN is a command that instructs the computer to perform an action. It can be used with a bracketed list or with a variable that is assigned the value of a list.

This example demonstrates another way to tell the Turtle to draw a box. One side of the box is drawn. The Turtle waits for one second, draws another side, waits, and then continues to repeat the process until you press BACK.

The RUN command also can be used to simplify complex procedures as shown in the TELL TURTLE second example. This procedure draws four equal squares connected by a smaller square in the center.

Examples:

```
TO BOX
TELL TURTLE
RUN [FORWARD 30
    RIGHT 90]
WAIT 60
BOX
END
```

```
TO BOXES
TELL TURTLE
CALL [REPEAT
    4[FORWARD 30
    RIGHT 90]] "X
CALL [LEFT 90
    FORWARD 20] "Y
PENUP
FORWARD 20
PENDOWN
RUN :X
RUN :Y
RUN :X
RUN :Y
RUN :X
RUN :Y
RUN :X
RUN :Y
END
```

TI LOGO REFERENCE GUIDE

SAVE

SAVE

To save any procedures and names in your workspace or any shapes and tiles you have defined, use the SAVE command. When you type SAVE and press ENTER, a menu appears on the display that gives you choices about what you want to save. Press 1 if you want to save only procedures and names. Press 2 if you want to save only shapes and tiles. Press 3 if you want to save everything (BOTH 1 AND 2). Next another menu appears giving you the choice of a cassette or diskette as a storage device.

If you are using a cassette recorder to save your work, pressing the number next to CASSETTE prompts you through the instructions for operating the recorder. Instructions also are provided to save your work on the TI Disk Memory System.

Procedures can be printed on the TI Thermal Printer. However, the printer cannot print the designs or the shapes and tiles used in a procedure. To save procedures, press 1 for PROCEDURES when the menu appears. Then when the next menu is displayed, press the appropriate number for the Thermal Printer and follow the directions on the display.



SENTENCE

SENTENCE *item item*

SENTENCE combines words or bracketed lists of words or numbers to form a sentence or new list. While it can be used for grammar exercises, it also can be used for some numerical exercises. For example, it can be used for comparison of two numerical lists. The order you enter the inputs is the order they appear in the new list.

See also: WORD

The SENTENCE operation must be used with a PRINT or other command. In the SAY procedure, the sentence prints on the display.

The second example prints a numerical sentence, 30 50.

The third example makes a sentence of a list and a value.

The fourth example shows how you can use SENTENCE to let you use words for setting the color of a character or the Turtle line to two colors.

Examples:

```
TO SAY
CALL [THIS IS A] "X
CALL [SAMPLE
      SENTENCE.] "Y
PRINT SENTENCE
      :X :Y
END
```

```
TO COMPARE
CALL 5*5+5 "X
CALL 5*(5+5) "Y
PRINT SENTENCE :X
      :Y
END
```

```
TO GREET "NAME
PRINT SENTENCE
      [HOW ARE YOU,]
      :NAME
END
```

```
TELL TURTLE
SETCOLOR SENTENCE
      :BLUE :RED
FORWARD 50
```

TI LOGO REFERENCE GUIDE

SETCOLOR

SETCOLOR *number*

SETCOLOR *:word*

SETCOLOR [*number number*]

Short form: **SC**

The SETCOLOR command assigns a color to the Turtle, a sprite, or a tile. The color can be indicated by a number or a word.

See also: COLOR, COLORBACKGROUND

The first example provides several examples of setting the color. When you enter the procedure DRAW, the display shows a red truck (sprite) driving on a black-edged road (the Turtle) with a white dotted line.

Examples:

```
TO DRAW
TELL TURTLE
HIDETURTLE
CALL [RIGHT 90 FORWARD
      241] "X
RUN :X
RIGHT 90
PENUP
FORWARD 30
PENDOWN
RUN :X
LINE
VAN
END

TO LINE
TELL TURTLE
PENUP
RIGHT 90
FORWARD 15
RIGHT 90
SETCOLOR 15
REPEAT 24 [PENDOWN
            FORWARD 5 PENUP
            FORWARD 5]
END

TO VAN
TELL SPRITE 2
CARRY :TRUCK
SETCOLOR :RED
SXY 0 [- 15]
SETHEADING 90
SETSPEED 35
END
```

TI LOGO REFERENCE GUIDE



The Turtle and tiles have a default color of black. A sprite's default color is clear; therefore, a sprite must be assigned a color value. The 16 color choices in TI LOGO can be assigned to the Turtle's ink, a tile, or a sprite by following the SETCOLOR command with the appropriate number or word.

When setting both the foreground and the background color of tiles, you must set the numbers within brackets. Numbers must be used rather than the name value of the color. The first number specifies the foreground color which sets the color of the character. The background color, the second number, sets the color of the background of that 8X8 tile. In addition, it is possible to set only the foreground color by entering a number or a word that has a value after the SETCOLOR command.

Tiles are grouped in 32 groups, each containing eight tiles. When the color of a tile is set, all of the other tiles in that group of eight characters appear that color. For example, if you set the colors for Tile 42 (asterisk) as a blue foreground and white background, all tiles numbered from 40 through 47 have the same foreground and background colors.

```
TELL TILE 42  
SETCOLOR [ 4 15 ]
```

TI LOGO REFERENCE GUIDE

SETHEADING

SETHEADING *number*

SETHEADING *:word*

Short form: **SH**

SETHEADING specifies the direction the Turtle or sprite moves. The SETHEADING command is followed by the number of degrees (0 through 360) or the words NORTH, SOUTH, EAST, and WEST. Due north is considered 0 degrees, east is 90 degrees, south is 180 degrees, and west is 270 degrees.

See also: HEADING

In the first example, the gray ball moves diagonally across the display.

In the DAY procedure, the Turtle draws a scene with grass and a tree. The SETHEADING command is used to position the Turtle in the desired direction. (Note that the DAY procedure consists only of the names of the subprocedures it is calling. This technique allows you to "build" long procedures in a modular fashion.)

Examples:

```
TELL SPRITE 12
HOME
CARRY :BALL
SETCOLOR :GRAY
SETHEADING 305
SETSPEED 10
```

```
TO DAY
GRASS
TREE
END
TO GRASS
TELL TURTLE
CLEARSCREEN
HIDETURTLE
SETHEADING 120
REPEAT 48 [LEFT 60
  FORWARD 3 RIGHT 60
  FORWARD 3]
END
TO TREE
SETHEADING 0
RIGHT 75
FORWARD 10
LEFT 40
FORWARD 5
SETHEADING :NORTH
FORWARD 30
SETHEADING 280
REPEAT 33 [RIGHT 10
  FORWARD 5]
SETHEADING :SOUTH
FORWARD 30
LEFT 40
FORWARD 10
END
```

TI LOGO REFERENCE GUIDE



SETSPEED

SETSPEED *number*

Short form: **SS**

Speed is one of the sprite attributes, and the SETSPEED command specifies how fast a sprite moves across the display. SETSPEED has a maximum positive speed of 127, which causes the sprite to move forward, or a maximum negative speed of -127, which causes the sprite to move backward. The Turtle cannot be given a speed.

See also: SPEED

The primitive SPEED can be used as a numeric variable which is assigned the value of a sprite's current speed. Thus the command SETSPEED SPEED + 5 increases the sprite's speed by 5. In the example on the right, the sprite stops when it reaches speed 60.

Examples:

```
TO ACCELERATE
TELL SPRITE 2
HOME
CARRY :TRUCK
SETHEADING :EAST
SETCOLOR :BLACK
HERE: SETSPEED SPEED + 5
IF SPEED = 60 GO "THERE
WAIT 30
GO "HERE
THERE: SETSPEED 0
END
```

SHAPE

SHAPE

SHAPE is one of the attributes that can be given to a sprite. The SHAPE operation returns the value of the shape the current sprite is carrying. The numeric value of SHAPE can be used in mathematical operations.

See also: COLOR, HEADING, SPEED

In the first example, Sprite 1 is carrying a plane which has the value of 1. When the CARRY SHAPE + 1 statement is entered, the sprite's shape changes to the next shape.

The WHAT procedure tests to see if the current sprite is carrying a plane whose value is 1. Depending on what the sprite is carrying, the computer prints the appropriate response. If the sprite is carrying shape number 1, THAT SPRITE IS CARRYING A PLANE is displayed. Otherwise, THAT SPRITE IS NOT CARRYING A PLANE is displayed.

Examples:

```
TELL SPRITE 1
HOME
CARRY :PLANE
SETCOLOR :RED
CARRY SHAPE + 1
CARRY SHAPE + 1

TO WHAT
TEST SHAPE = 1
IFT PRINT [THAT SPRITE
  IS CARRYING A PLANE]
IFF PRINT [THAT SPRITE IS
  NOT CARRYING A PLANE]
END
```

TI LOGO REFERENCE GUIDE

SHOWTURTLE

SHOWTURTLE

Short form: **ST**

The SHOWTURTLE command makes the Turtle reappear after it is hidden by the command HIDE TURTLE.

See also: HIDE TURTLE, NOT TURTLE

The STAR procedure hides the Turtle before it draws a star. After drawing a star, the Turtle reappears and draws another one overlapping the first.

Examples:

```
TO STAR
TELL TURTLE
HIDE TURTLE
RIGHT 18
CALL [REPEAT 5 [FORWARD
  30 RIGHT 144 FORWARD
  30 LEFT 72]] "STAR1
RUN :STAR1
PENUP
SHOWTURTLE
LEFT 30
PENDOWN
RUN :STAR1
END
```

SPEED

SPEED

SPEED is an attribute that can be given to a sprite. The SPEED operation returns the speed of the active sprite or sprites. The numeric value of SPEED can be used in mathematical and test operations.

See also: COLOR, HEADING, SETSPEED, SHAPE

In the first example, a black truck moves across the display at a speed of 90. When the second SETSPEED statement is entered, the sprite moves more slowly.

In the ACCELERATE procedure, a blue rocket moves up the display with its speed slowly increasing. When its speed reaches 120, the procedure is stopped and the rocket remains traveling at a speed of 120.

Examples:

```
TELL SPRITE 14
HOME
CARRY :TRUCK
SETCOLOR 1
SETSPEED 90
PRINT SPEED
SETSPEED SPEED / 3

TO ACCELERATE
TELL SPRITE 8
CARRY :ROCKET
SETCOLOR :BLUE
IF SPEED = 120 STOP
SETSPEED SPEED + 5
WAIT 30
ACCELERATE
END
```

TI LOGO REFERENCE GUIDE



STOP

STOP

The STOP command stops a procedure before all of the instructions are carried out. It can be used alone but is generally used in conjunction with a conditional statement. *Note:* If any sprites are moving when STOP is performed, they continue to move.

In the ACC procedure, a truck is traveling across the display. When the truck's speed reaches 60, the procedure stops; however, the truck continues to move across the display.

Examples:

```
TO ACC
TELL SPRITE 2
CARRY 2
HOME
SETCOLOR 1
SETHEADING 90
HERE: SETSPEED SPEED + 5
IF SPEED = 60 STOP
WAIT 30
GO "HERE
END
```

SUM

SUM *value value*

SUM *:word :word*

Short form: value + value
:word + :word

SUM is an addition operation that results in the numerical sum of two values, two words that have been assigned numerical values, or a value and a word that has been assigned a numerical value.

See also: DIFFERENCE, PRODUCT, QUOTIENT

In the first example, the output 35 is printed on the display. In the second example, the output is 76.

Examples:

```
PRINT SUM 23 12
CALL 45 "FROG
CALL 31 "TADPOLE
PRINT :FROG + :TADPOLE
```

TI LOGO REFERENCE GUIDE

SV

SV *:word :word*

SV *number number*

SV is used to set the velocity of a sprite. The first number is the x-velocity and the second number is the y-velocity. It is possible to change the velocity of a sprite for either or both of the x- and y-coordinate directions.

See also: SXV, SYV, XVEL, YVEL

SV, followed by two numbers ranging from 127 to -127, give the active sprite the specified x-velocity and y-velocity.

In this example, a blue box is put on the display at HOME. When you run the procedure inputting two variables, the speed and heading of the box are specified. If you enter VEL 0 60, the velocity on the x-coordinate plane is 0 and the velocity on the y-coordinate plane is 60. The box moves across the display at a speed of 60. If you change the variables to 60 60, the box moves across the display in a diagonal direction at a speed greater than 60. (Type PRINT SPEED to see at what speed the sprite is moving.) Note that the x and y "forces" combine to give the sprite new direction, but don't give it twice the speed.

Examples:

```
TO VEL VELX VELY
TELL SPRITE 5
HOME
CARRY :BOX
SETCOLOR :BLUE
SV :VELX :VELY
END

VEL 0 60
VEL 60 60
```



SX/SY/SXY

SX *number* **SX** *:word*

SY *number* **SY** *:word*

SXY *number number*

SXY *:word :word*

SX, SY, and SXY are commands used to position sprites on the display. The x- and y-coordinates divide the display into four quadrants, or four equal sections with HOME being the center.

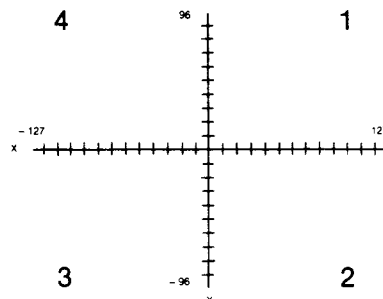
The x-coordinate plane is an imaginary horizontal line running across the display through HOME. The points designated on this line begin at -127 on the far left side of the display and end at 127 on the right side of the display.

The y-coordinate plane is an imaginary vertical line also running through HOME. The vertical line includes points from 96 to -96, top to bottom respectively.

The four quadrants are read clockwise, beginning with the upper-right section on the display. All x- and y-coordinate points in the first quadrant (upper right) are positive integers. In the second quadrant (lower right), the x-coordinates are positive and the y-coordinates are negative. The third quadrant (lower left) contains all negative x- and y-coordinates and the fourth quadrant (upper left) consists of negative x-coordinates and positive y-coordinates. Therefore, if you want to put a sprite into a particular quadrant, you must indicate whether the coordinates are positive or negative.

To place a sprite in a quadrant, you can specify one coordinate by using the SX or SY commands followed by a number. Or, you can specify both x- and y-coordinates with the SXY *number number* command. The first input is the x-coordinate, while the second input is the y-coordinate. *Note:* Negative y-coordinates must be enclosed in parentheses; otherwise, the computer evaluates the coordinate as a mathematical expression.

Examples:



```
TO V
TELL [1 2 3 4 5 6 7]
CARRY :BOX
SETCOLOR :BLACK
HOME
TELL SPRITE 1 SX -48 SY
  48
TELL SPRITE 2 SXY -32 32
TELL SPRITE 3 SXY -16 16
TELL SPRITE 4 SXY 48 48
TELL SPRITE 5 SXY 32 32
TELL SPRITE 6 SX 16 SY 16
TELL SPRITE 7 SXY 00
END
```

TI LOGO REFERENCE GUIDE

SXV/SYV

SXV *number* **SXV** *:word*

SYV *number* **SYV** *'word*

It is possible to specify a velocity for only one coordinate direction. By using SXV or SYV, followed by a speed from 127 to -127, the velocity of the active sprite changes the direction and rate of the specified coordinate plane.

See also: SV, XVEL, YVEL

In the first example, both the x- and y-velocities are set at 50. When SXV 20 is entered, only the x-velocity is reduced. The y-velocity changes after SYV 20 is entered.

Examples:

```
TELL SPRITE 5  
HOME  
CARRY :BALL  
SETCOLOR :RED  
SV 50 50  
SXV 20  
SYV 20
```

TI LOGO REFERENCE GUIDE



TELL

TELL *listener*

TELL is the command used to indicate who is to be the current listener. TELL can be used with a sprite, the Turtle, a tile, or the background. You can also TELL a list of sprites or tiles to be the listeners.

With the command TELL, you can address a sprite by a number or by a word. The complete form of address is TELL SPRITE *number* (0 through 31). However, the use of the word SPRITE is optional when addressing a single sprite. When talking to a list of sprites with the TELL command, the word SPRITE is not included in the list; only the numbers need to be included within the brackets.

TELL TURTLE is the command used to enter the Turtle mode. If active sprites are on the display, they remain there. It also is possible to add sprites while in the Turtle mode simply by telling a new sprite to perform an operation.

TELL BACKGROUND is used to tell the background to change color. It performs the same function as the COLORBACKGROUND command except TELL BACKGROUND makes the background the current listener of all commands.

You also can address all 32 sprites with the command TELL :ALL. In the example, all of the sprites are the current listener even if they are not assigned any attributes.

Examples:

```
TELL SPRITE 2  
HOME  
CARRY :ROCKET  
SETCOLOR :PURPLE
```

```
TELL [10 11 12]  
HOME  
CARRY :BALL  
SETCOLOR :YELLOW  
EACH [SETHEADING  
YOURNUMBER * 10]  
SETSPEED 40
```

```
TELL TURTLE  
NOTURTLE
```

```
TELL BACKGROUND  
SETCOLOR :BLUE
```

```
TELL :ALL  
SETSPEED 0  
HOME  
CARRY :BOX  
EACH [SETCOLOR  
YOURNUMBER]  
EACH [SETHEADING  
YOURNUMBER * 12]  
SETSPEED 80
```

TI LOGO REFERENCE GUIDE

TEST

TEST *expression*

The TEST command evaluates an expression and returns a value of true or false. Used with the commands IFF (IF False) and IFT (IF True), the value can be used in other conditional statements.

See also: IF, IFF, IFT

Because the color is set as black, or 1, in this procedure, the speed is set at 30 as directed in line 8 of CHECK.

Examples:

```
TO CHECK
TELL SPRITE 1
HOME
CARRY 2
SETCOLOR 1
SETHEADING 90
TEST COLOR = 1
IFF SETSPEED 0
IFT SETSPEED 30
PRINT SPEED
END
```

THAW

THAW

THAW is the command used to restart all of the sprites on the display that are stopped by the command FREEZE.

See also: FREEZE

In the MOVE procedure, all sprites stop moving after several seconds. After you enter THAW, the sprites begin to move again.

Examples:

```
TO MOVE
TELL :ALL
CARRY :ROCKET
SETCOLOR 6
SETSPEED 20
WAIT 70
FREEZE
END
THAW
```



THING

THING *"name*

THING *:word*

THING returns the numeric value of the given name or word. The name or word can represent a word, a number, a list, or even a procedure.

In the example on the right, the SHAPES procedure assigns four of the predefined shapes' names. The subprocedure SHAPES1 calls the input character "X. The second line of SHAPES1 tests to see if any numeric value is associated with the key that is pressed. If the key that is pressed is an S, C, R, or T, the corresponding shape appears on the display. If no value is associated with the pressed key, an error message appears.

Examples:

```
TO SHAPES
CALL 5 "S; S FOR SQUARE
CALL 4 "C; C FOR CIRCLE
CALL 3 "R; R FOR ROCKET
CALL 2 "T; T FOR TRUCK
TELL SPRITE 1
SETCOLOR :BLACK
HOME
SHAPES1
END

TO SHAPES1
CALL READCHAR "X
CARRY THING :X
SHAPES1
END
```

TI LOGO REFERENCE GUIDE

THING?

THING? :word

THING? "name

THING? is an operation that determines if a non-numeric value is assigned to the given name or word. If it has a non-numeric value, the computer returns true. The computer returns false if the name or word has a numeric value or has no value.

See also: NUMBER?, WORD?

In the first example, "C has no value so the background remains the same color. In the second example, "C is assigned the value of "B. When the IF statement is entered, the background turns white.

The SHAPES procedure gives the values of four of the predesigned shapes the names "S "C, "R, and "T. (Everything following the ; is for your reference only; the computer ignores this data. Refer to the "A Place for Comments—After ;" section in the Owner's Manual.) The subprocedure SHAPES1 causes the corresponding shape to appear when one of the four indicated letters is pressed. If no value is assigned a pressed key, the procedure calls itself and waits for another key to be pressed.

Examples:

```
IF THING? "C
  COLORBACKGROUND 15
CALL "C "B
IF THING? "B
  COLORBACKGROUND 15
```

```
TO SHAPES
CALL 5 "S; S FOR SQUARE
CALL 4 "C; C FOR CIRCLE
CALL 3 "R; R FOR ROCKET
CALL 2 "T; T FOR TRUCK
TELL SPRITE 1
SETCOLOR :BLACK
HOME
SHAPES1
END

TO SHAPES1
CALL READCHAR "X
TEST THING? :X
IFF SHAPES1
IFT CARRY THING :X
SHAPES1
END
```

TI LOGO REFERENCE GUIDE



TO

TO *name*

TO is the command used to teach the computer. When you enter TO and the name of a procedure, the display turns green. The name of a procedure can be a word, a single character, or a combination of characters, numbers, and symbols. A procedure cannot be named by a single digit, a primitive, or the following symbols: *, /, +, -, :, ;, and ``.

See also: EDIT

After you design a procedure, press BACK to leave the teaching mode. To run your procedure, enter the procedure's name.

The HOUSE procedure includes a number of other procedures to draw a house.

Examples:

```
TO HOUSE
FRONT
DOOR
WINDOW
CHIMNEY
END

TO FRONT
TELL TURTLE
HIDETURTLE
FORWARD 35
RIGHT 45
BACK 4
FORWARD 40
RIGHT 90
FORWARD 40
BACK 4
RIGHT 45
FORWARD 35
RIGHT 90
FORWARD 50
END

TO DOOR
SXY 20 0
SETHEADING 0
FORWARD 20
RIGHT 90
FORWARD 9
RIGHT 90
FORWARD 20
SXY 22 8
FORWARD 1
END

TO WINDOW
SXY 6 5 RECT
SXY 36 5 RECT
END
```

TI LOGO REFERENCE GUIDE

```
TO RECT  
SETHEADING 0  
FORWARD 12  
RIGHT 90  
FORWARD 8  
RIGHT 90  
FORWARD 12  
RIGHT 90  
FORWARD 8  
END
```

```
TO CHIMNEY  
SETHEADING 0  
SXY 30 55  
FORWARD 10  
RIGHT 90  
FORWARD 8  
RIGHT 90  
FORWARD 19  
END
```



TRACEBACK

TRACEBACK

Short form: TB

TRACEBACK is a command that can be incorporated in procedures and subprocedures to help test those operations. When the computer reaches the TRACEBACK statement in a procedure, it prints the statement, WE'RE NOW INSIDE...(the appropriate name of the procedure or subprocedure), the procedure that called that procedure, and the superprocedure before it if there is one. It lists all subprocedures within a superprocedure, beginning with the last subprocedure.

TRACEBACK is a useful debugging tool to trace the operation of a procedure. You can put it in a procedure, and then when it helps you debug, you can take it out of the procedure. It also can be used while you're paused. If you press SHIFT A or FCTN 7 while running a procedure, and then type TRACEBACK, the computer prints your message, WE'RE NOW INSIDE....

Using the procedure on the right, TRACEBACK tells you the subprocedures that are active from D to the superprocedure A and at what level you paused.

Examples:

```
TO A
B
END

TO B
C
END

TO C
D
END

TO D
TELL TURTLE
FORWARD 20
RIGHT 90
TRACEBACK
D
END
```

TI LOGO REFERENCE GUIDE

TYPE

TYPE *"name*

TYPE [*list*]

TYPE *:word*

The TYPE command tells the computer to print a word, an ASCII character, a symbol, a list, a value, or the results of a numeric expression. When the computer completes printing the output, TYPE leaves the cursor at the end of the line it prints.

See also: PRINT

The example on the right demonstrates the difference between TYPE and PRINT.

Examples:

```
TO WRITE
TYPE "BE
TYPE "FORE
PRINT "BE
PRINT "FORE
TYPE [HI SCOTT]
TYPE [HI] PRINT [SCOTT]
END
```

WAIT

WAIT *number*

WAIT causes the computer to pause for the specified number times 1/60th of a second. WAIT 60, as an example, causes a delay of one second in an operation of the procedure.

WHERE

WHERE

The WHERE operation returns the x- and y-coordinates, and the heading of the Turtle. The first value that is returned is the x-coordinate. The second value is the y-coordinate and the Turtle's heading is the third value.

In the example on the right, the PRINT statement outputs the Turtle's location on the display and its heading.

Examples:

```
TELL TURTLE
FORWARD 30
RIGHT 45
FORWARD 15
PRINT WHERE
```

TI LOGO REFERENCE GUIDE



WHO

WHO

WHO returns the name of the current listener. If you are talking to a sprite, the number of the active sprite is returned. If you are talking to more than one sprite, the numbers of all sprites are returned. If you are talking to the Turtle or the background, WHO returns their names.

See also: SHAPE, YOURNUMBER

In the example on the right, the computer prints 1 3 5 8 on the display when the PRINT statement is entered.

Examples:

```
TELL [1 3 5 8]
HOME
CARRY :BALL
SETCOLOR :BLUE
PRINT WHO
```

WORD

WORD "name "name

WORD :word :word

WORD is an operation that combines two inputs into one word. The inputs can be names or words with non-numeric values.

See also: SENTENCE

In the first example, the word BEFORE is printed on the display. The second example prints SUNFLOWER.

Examples:

```
PRINT WORD "BE "FORE
CALL "SUN "S
CALL "FLOWER "F
PRINT WORD :S :F
```

WORD?

WORD? "name

WORD? :word

WORD? is an operation that tests to see if the input is a word. If so, WORD? returns true. If the input is a list or a number, WORD? returns false. This operation is useful for testing variables in long or complex procedures.

See also: NUMBER?, THING?

In the example, since TEST is a list, the condition is false and NO is printed on the display.

Examples:

```
PRINT [THIS IS A LIST.]
CALL [THIS IS A LIST] "TEST
IF WORD? :TEST PRINT "YES
ELSE PRINT "NO
```

TI LOGO REFERENCE GUIDE

XCOR

XCOR

The operation XCOR returns the x-coordinate of the active sprite or Turtle.

See also: YCOR

XCOR returns a numerical value. In the first example, the value of the x-coordinate is 40. The SETSPEED XCOR command gives the sprite the speed of 40.

The XCOR operation determines the current location of the Turtle or a sprite on the display and outputs the numerical value of the x-coordinate.

Examples:

```
TELL SPRITE 6
CARRY :BALL
SETCOLOR :YELLOW
SETSPEED 0
SX 40
PRINT XCOR
SETSPEED XCOR
SETSPEED 0
HOME
PRINT XCOR
TELL TURTLE
HOME
PRINT XCOR
FORWARD 40
PRINT XCOR
RIGHT 90
FORWARD 40
PRINT XCOR
FORWARD XCOR
PRINT XCOR
SX -30
PRINT XCOR
```

XVEL

XVEL

XVEL is an operation that returns the x-velocity of the active sprite by determining the east and west components of the sprite's speed.

See also: SXV, SYV, YVEL

In the example on the right, certain attributes are given to the sprite. The PRINT statement outputs the sprite's x-velocity which is 36.

Examples:

```
TELL SPRITE 17
HOME
CARRY :ROCKET
SETCOLOR :BLACK
SETHEADING 54
SETSPEED 45
PRINT XVEL
```



YCOR

YCOR

The YCOR operation returns a numerical value determined by the location of the Turtle, sprite, or sprites on the display.

See also: XCOR

YCOR may be a positive or negative number. The horizontal coordinates above HOME (YCOR 0) have a positive value. The coordinates below HOME have a negative value.

In the Turtle mode, the y-coordinates are 88 and -48 before the Turtle wraps around the display. These coordinates create a smaller grid than in the sprite mode because the lower portion of the display is reserved for commands. In the sprite mode, the y-coordinates are 96 and -94 before the sprite wraps around the display.

Examples:

```
TELL SPRITE 10
CARRY :ROCKET
SETCOLOR :PURPLE
HOME
PRINT YCOR
FORWARD 50
PRINT YCOR
BACK 90
PRINT YCOR

TELL TURTLE
FORWARD 40
PRINT YCOR
BACK 50
PRINT YCOR
BACK 38
PRINT YCOR
BACK 20
PRINT YCOR
BACK YCOR
PRINT YCOR
```

YVEL

YVEL

YVEL is an operation that returns the y-velocity of the active sprite by determining the north and south components of the sprite's speed.

See also: SXV, SYV, XVEL

In the example, the y-velocity of the sprite is printed on the display.

Examples:

```
TELL SPRITE 30
HOME
CARRY :PLANE
SETCOLOR :PURPLE
SETHEADING 117
SETSPEED 69
PRINT YVEL
```

TI LOGO REFERENCE GUIDE

YOURNUMBER

YOURNUMBER

Short form: **YN**

YOURNUMBER is an operation that returns the number of the current sprite. The 32 sprites (0 through 31) are identified by a specific number.

See also: NUMBEROF, SHAPE, WHO

YOURNUMBER can be used in several ways. In the first example, SPRITE 20 is told to set certain attributes depending on its assigned number.

If you are talking to more than one sprite, the computer does not know what to do with the value of the first sprite in the list. If you address the list of sprites with the command EACH, the computer prints the corresponding numbers on the display.

Examples:

```
TELL SPRITE 20
HOME
CARRY YOURNUMBER
SETCOLOR YOURNUMBER
SETHEADING YOURNUMBER
  * 100
TELL [10 20 30]
PRINT YOURNUMBER
EACH [PRINT YOURNUMBER]
```

NOTES

