

Additions to Tiny Pilot

These additions to Tiny Pilot include code to input a numeric variable, generate a random number, and call a machine language subroutine. A complete sample Pilot program is included.

Bob Applegate
Box 148
Bordentown, NJ 08505

Nicholas Vrtis' Tiny PILOT is a neat way to move up to a high level language, but it does have some drawbacks. One of the biggest problems is the lack of a method to input into a variable from running program. All values must be preset by a C: command. This can be a real hassle for some applications.

Another useful addition would be a machine language subroutine call. It would allow you to write programs using functions that standard Pilot doesn't have, like having a beeper rather than a "?" for a prompt. Or maybe comparing the contents of two variables and setting a flag to indicate which is larger.

One more function that could be added is a random number generator. Some games (until my KIM takes over the world, I'll resort to playing games on it!), such as HI-LO and CRAPS, can be played only if a random value can be created. If any of these problems bother you, then read on!

These routines will solve all of these problems. Before I start detailing them, realize that they will take away from memory space for the source (in Pilot). This will be no problem if your system has extra memory, but my 2K is filled really fast with a long program! Don't use a lot of remarks and long strings to conserve space.

Let me start by describing what modifications are needed to Tiny

PILOT for these programs to work. Make the following corrections:

```
027E 4C 16 05
0281 EA
```

That just tells the interpreter to try to match the current command with our new ones before it checks its own. The instructions that we just wiped out are replaced at 0516. Correct the following:

```
048C A9 06
```

That tells the interpreter that the Tiny PILOT source begins at page 6, not page 5. Addresses 04FA to 0515 are just relocated versions of the subroutines previously described by me (MICRO, 21:41). If your system doesn't need them, relocate the rest of the program to 04FA. If you will be using them, remember to correct all the I/O calls in the Pilot interpreter. Here are the new instructions:

I:x Input a positive number into variable x (can be any from A to Z). Prints a "?" as a prompt.

P:x Puts a random number into variable x (can be any from A to Z). The number will be in the range 0 to 99.

L:x Calls machine language subroutine x (can be any name from A to Z). The starting address of the subroutine is stored in the following table:

Name	Zero page address	
A	A0,	A1
B	A2,	A3
C	A4,	A5
.	.	
.	.	
.	.	
Y	D0,	D1
Z	D2,	D3

Here's how they work. 0516 to 0519 only replace what we destroyed at 027E to 0281. The first four instructions see if the next command is an L: command. If not, it jumps to 0531 for the next command. If it is the right one, it jumps to the subroutine at 0494 to get the index for the label name. Then it uses the index to get the starting address of the subroutine from the table (low-order first). Then it puts the values at the appropriate locations (052C, 052D) and makes a jump to the subroutine. This routine can't be PROMmed.

There is probably a better way to execute that jump, but this way is easy, and it works. Finally it jumps back to 0279.

I can't take credit for the random number generator (053A to 054B). It is a slightly modified version of the one presented by Jim Butterfield on page 172 of *The First Book of KIM*. I suggest that you look there for the theory behind it. Addresses 0531 to 0534 just check to see if we are executing the correct command. A call is made to 0494 for the index. The X register is stored for future use at

008D. Then the random number is produced. The result is in the A register. X is loaded again; then the value of A is stored in the proper variable. It finished by jumping to 0279

All that is left now is the I: command. If it's not an I, the program jumps to 0591. The next five instructions output a prompt character ("?") and clear the temporary work area (00DA, 00DB). Then it gets the ASCII input. If it is a CR, it jumps ahead to 0580. Otherwise, it subtracts \$30 to get a decimal number. Next, it rolls 00DA and 00DB four places to the left, to make room for the new digit. The value of the A register is added to 00DA to achieve the new number. The program jumps back to 0564 to get the next character.

Once a CR input, the program goes to 0580. Then it jumps to 0494 for the index. The contents of 00DA and 00DB are stored at the proper variable. Then the program outputs a CR and LF, and finally jumps back to 0279.

If the command didn't match any of those, the program goes back to 0282, where it looks through the standard Pilot instructions. Additional commands can be added from 0591 and up. The A register will already contain the command character, so just use a CMP instruction to see if it is the one you want. The Y register already points to the character after the ":", so just use a B1 97 to load it into the A register. The last instruction should be 4C 79 02. The very last instruction after your additional routines must be 4C 82 02.

I hope that these new commands will increase the use of Tiny PILOT. It is really a good language, considering its small size. I have included some sample Tiny PILOT programs to demonstrate what it can do.

μ

Bob Applegate is seventeen years old, an 11th grade student. He has been accepted to a local college where he plans to major in computer science. He has been working with computers for about four years, starting with BASIC, at Princeton University.

His one-year-old KIM is about to be upgraded to 16K, with OSI BASIC-in-ROM.

New Pilot Commands

0516-	85 87	STA	\$87	CLEAR HIGH BIT FOR EDITOR
0518-	08	INY		POINT TO ":"
0519-	08	INY		AND THE NEXT CHARACTER
051A-	C9 4C	CMP	#\$4C	IS IT THE L COMMAND
051C-	D0 13	BNE	\$0531	NO, GO TO 0531
051E-	20 94 04	JSR	\$0494	COMPUTE THE INDEX
0521-	B5 A0	LDA	\$A0,X	GET THE HIGH-ORDER BYTE FROM TABLE
0523-	80 2C 05	STA	\$052C	PUT IT BEHIND THE JSR
0526-	B5 A1	LDA	\$A1,X	GET THE LOW ORDER BYTE
0528-	80 20 05	STA	\$0520	PUT IT BEHIND THE JSR
052B-	20 00 00	JSR	\$0000	EXECUTE THE SUBROUTINE
052E-	4C 79 02	JMP	\$0279	ALL DONE, RETURN TO PILOT
0531-	C9 50	CMP	#\$50	IS IT THE P COMMAND
0533-	D0 20	BNE	\$0555	NOPE, GO TO 0555
0535-	20 94 04	JSR	\$0494	COMPUTE THE INDEX
0538-	86 80	STX	\$80	STORE THE INDEX FOR NOW
053A-	F8	SED		DECIMAL NUMBERS ONLY, PLEASE
053B-	38	SEC		CARRY ADDS VALUE 1
053C-	A5 D5	LDA	\$D5	LAST VALUE
053E-	65 D8	ADC	\$D8	ADD B+CARRY
0540-	65 D9	ADC	\$D9	ADD C
0542-	85 D4	STA	\$D4	NEW NUMBER
0544-	A2 04	LDX	#\$04	MOVE 5 NUMBERS
0546-	B5 D4	LDA	\$D4,X	GET FIRST NUMBER
0548-	95 D5	STA	\$D5,X	MOVE OVER 1
054A-	CA	DEX		NEXT NUMBER
054B-	10 F9	BPL	\$0546	ALL MOVED?
054D-	D8	CLD		EVERYTHING BACK YO NORMAL
054E-	A6 80	LDX	\$80	PICK-UP THE INDEX
0550-	95 54	STA	\$54,X	STORE THE NUMBER AT VARIABLE
0552-	4C 79 02	JMP	\$0279	BACK TO PILOT
0555-	C9 49	CMP	#\$49	SEE IF IT'S THE I COMMAND
0557-	D0 38	BNE	\$0591	GO TO 0591 IF NOT
0559-	A9 3F	LDA	#\$3F	GET A "?"
055B-	20 02 05	JSR	\$0502	OUTPUT IT SAS A PROMPT
055E-	A9 00	LDA	#\$00	CLEAR THINGS OUT
0560-	85 DA	STA	\$DA	ESPECIALLY THIS TEMPORARY VARIABLE
0562-	85 DB	STA	\$DB	DITTO
0564-	20 FA 04	JSR	\$04FA	GET INPUT
0567-	C9 00	CMP	#\$00	CR?
0569-	F0 15	BEQ	\$0580	YES, GO TO 0580
056B-	38	SEC		GET READY TO SUBTRACT
056C-	E9 30	SBC	#\$30	TURN ASCII INTO BCD
056E-	A2 04	LDX	#\$04	GET READY TO MULTIPLY BY 10
0570-	18	CMC		CLEAR THINGS FIRST
0571-	06 DB	ASL	\$DB	MULTIPLY
0573-	26 DA	ROL	\$DA	MULTIPLY
0575-	CA	DEX		AGAIN?
0576-	D0 F8	BNE	\$0570	YES, THEN 0570
0578-	18	CMC		CLEAR THINGS UP AGAIN
0579-	65 DB	ADC	\$DB	ADD THE NEW DIGIT
057B-	85 DB	STA	\$DB	STORE THE ANSWER
057D-	4C 64 05	JMP	\$0564	DO IT ALL OVER AGAIN
0580-	20 94 04	JSR	\$0494	GET THE INDEX
0583-	A5 DA	LDA	\$DA	GET FIRST PART OF ANSWER
0585-	95 53	STA	\$53,X	STORE IT AT VARIABLE
0587-	A5 DB	LDA	\$DB	GET THE NEXT PART
0589-	95 54	STA	\$54,X	AND STORE THAT
058B-	20 0A 05	JSR	\$050A	START A NEW LINE
058E-	4C 79 02	JMP	\$0279	ALL DONE, RETURN TO PILOT
0591-	4C 82 02	JMP	\$0282	NOT A NEW COMMAND, CHECK OLD ONES

Some Relocated Subroutines (See MICRO 21:41)

KIM					
07DC	20	G			
04FA-	84	EE		STY	\$\$\$E
04FC-	20	5A 1E		JSR	\$\$\$E5A
04FF-	A4	EE		LDY	\$\$\$E
0501-	60			RTS	
0502-	84	EE		STY	\$\$\$E
0504-	20	A0 1E		JSR	\$\$\$EAO
0507-	A4	EE		LDY	\$\$\$E
0509-	60			RTS	
050A-	86	ED		STX	\$\$\$E
050C-	84	EE		STY	\$\$\$E
050E-	20	2F 1E		JSR	\$\$\$E2F
0511-	A6	ED		LDX	\$\$\$E
0513-	A4	EE		LDY	\$\$\$E
0515-	60			RTS	