SLIDE

(TI Contest version. **NOT TO BE PUBLICALLY DISTRIBUTED**)

1) Introduction

1.1) What is SLIDE?

SLIDE is a new application programming language for the TI-83 Plus calculator. It combines the power of assembly language programming with the ease of BASIC programming. It provides an easy to use extendible command language in the familiar ZDS interface. No knowledge of assembly language is required, though assembly language subroutines can be integrated into SLIDE programs.

1.2) Why was SLIDE created?

SLIDE was created to allow programmers and non-programmers alike to create high quality applications for the TI-83 Plus. Assembly programming is very difficult to master, and BASIC programming is very limited. SLIDE was created to bridge the gap between the these two languages.

1.3) Who should use SLIDE?

Anyone who wants to create an application for the TI-83 Plus calculator can use SLIDE. It is easy enough to use that even people with no programming experience will be able to use SLIDE after reading through the brief programming tutorial. Those with programming experience in BASIC will find SLIDE more powerful, and very easy to understand and use. Even assembly programmers may consider using SLIDE to remove the tedium from simple routines and allow them to concentrate on the more difficult and important aspects of their programs (which they may write in SLIDE or assembly language).

1.4) Feature list

- Easy to understand command syntax
- Extremely extendible using libraries
- Interpreted language allows compile-time and run-time error checking
- Uses familiar ZDS interface/compiler
- Only necessary libraries are linked, saving space
- Much faster execution speed than BASIC programs, often nearing assembly

2) Getting Started

2.1) Requirements

- ZDS - Zilog Developer Studio (downloadable from http://www.zilog.com)
- TI-83 Plus calculator for on-calculator testing
- TI Flash Debugger/Simulator for on-computer testing (downloadable from http://education.ti.com)

2.2) Creating a SLIDE project

SLIDE comes with a template project to help you get started creating your own SLIDE application. Create a new folder for your project, then copy all the files in the template directory to this folder. Rename the 'SlideTemplate.zds' file to whatever you want your project name to be. Open this file and ZDS will load. Once ZDS loads you'll see a large list of files in the FileView. Don't let this overwhelm you; most of these files are library files that you'll never need to be concerned with. The two files that you need to know about are 'slidetemplate.asm' and 'settings.inc'.

The 'slidetemplate.asm' file holds your actual slide program. Double click it to open the file. You'll see something that looks like this:

```
;Sample SLIDE application
 INCLUDE "slides.inc"
 INCLUDE "mainlib.inc"
 INCLUDE "gfxlib.inc"
 INCLUDE "textlib.inc"
 INCLUDE "misclib.inc"
 INCLUDE "mathlib.inc"
 INCLUDE "silverlib.inc"
 INCLUDE "floatlib.inc"
 INCLUDE "appvarlib.inc"

 DefineMainSlide MainSlide
       LargeText 10,28,"SLIDE Template"
       WaitKey
       Return
```

The first line starts with a semicolon, indicating that it is a comment. You can put comments anywhere in your program. All text on a line after a semicolon will be ignored. The next line includes the 'slides.inc' file into your project. This is a necessary file that must be included in all projects. The next eight lines include different library files, each of which provides commands that you can use in your SLIDE program.

Now the real program starts. The **DefineMainSlide** command, followed by a label, indicates where the beginning of the application is. The next line executes a **LargeText** command, which displays text to the screen in the large font. Then **WaitKey** pauses until a key has been pressed, and **Return** exits the program and returns to the operating system. More information on these and other commands can be found in the command reference.

The 'settings.asm' file holds configuration information for the application, and will look like this:

```
 IFNDEF _SETTINGS_
 DEFINE _SETTINGS_
 DEFINE TI83P
; DEFINE TI73
```

```
Version  EQU 1
AppName EQU  "SlideTmp"
MaxSlideDepth EQU 10
NumVariables EQU 26
; DEFINE FASTSPEED
; DEFINE LIMITEDUSE
; DEFINE TIOSCHECK
;MajorTIOSVersion = 1
;MinorTIOSVersion = 14
 ENDIF
```

The first two lines are unimportant, they're just for the compiler. The next line, **DEFINE TI83P**, indicates that the application should be created for the TI-83 Plus calculator. If you want to compile the SLIDE application for the TI-73 calculator instead, comment this line (by putting a semicolon in front of it), and uncomment the next line (by removing the semicolon). SLIDE programs should compile identically for the TI-83 Plus and TI-73, but each compile must be done separately, you can not compile for both platforms at the same time.

The next line defines the version number of the program. This is a number 1-255 that is placed into the header of the application. You don't need to change this if you don't want to. The next line defines the name of the application (as it will appear on the APPS menu). This name must be eight characters long (you can pad it with spaces if you want a shorter name) and must be enclosed in quotes. The rest of the lines are for other compiling options which are detailed later in this document.

2.3) Compiling and testing

Compiling your SLIDE project is easy, just press the F7 key in ZDS. Compile information will display in the build window, and you can double click on any errors to jump to the line where they occurred. Some errors are triggered by SLIDE's error checking and should be descriptive as to what the error might be caused by. However, other errors may be triggered by ZDS itself, and these will be more cryptic, but they should still point to the line number in question.

Once the project has been compiled, you can test it. If you want to test on a computer, you can use the Flash Debugger/Simulator and load the .hex file.

3) Libraries

3.1) Introduction to libraries

Without libraries SLIDE applications would be unable to do anything. Libraries provide all commands in the SLIDE language; no commands are built into the language. The SLIDE template starts with the Main, Math, and Text libraries in the project. You will need to add additional libraries if you need to use commands not included in the defaults. Because libraries take up space in your SLIDE application, it is good practice to include only the libraries you need in your project.

To add a library to your project, choose "Project : Add to Project : Files…" from the ZDS menu bar. Find the .asm file for the library and select it. Then, in your SLIDE program file, add another line near to the top to INCLUDE the .inc file for the library. For example, if you wanted to use the floating point library, you would add the line

```
 INCLUDE "floatlib.inc"
```

(note: there is a blank space before the "INCLUDE")

To remove a library from your project, click on the .asm file for the library in the FileView panel, and then choose "Project : Remove from Project" from the ZDS menu bar. Then remove the INCLUDE line for the .inc file associated with the library from the SLIDE program file.

3.2) Main Library

The Main library provides basic language commands. All SLIDE programs must include the Main library; without it there is no way to return to the operating system. Commands included in the main library are:

- Copy - Copies data to a variable (see: Load)
- Exec - Executes an assembly language sub-routine
- IfEqu - Executes next command if conditions are equal
- IfGreater - Executes next command on greater condition
- IfLess - Executes next command on less than condition
- IfNEqu - Executes next command if conditions are not equal
- Jump - Jumps to a label in the SLIDE program
- Load - Loads data to a variable (see: Copy)
- LoopToNum - A looping mechanism (similar to For loop)
- LoopToZero - A simpler and faster looping mechanism
- Quit - Returns control to the operating system
- Return - Returns control to the calling function
- Slide - Calls a SLIDE sub-routine

3.3) GFX Library

The GFX library provides graphics and display related commands. For text commands see the Text library. Commands included in the GFX library are:

- ClearScreen - Clears the screen
- IfPointOn - Executes the next command if a screen pixel is black
- InvertScreen - Inverts the current screen image
- Line - Draws a line

- Point - Draws a point
- Rect - Draws a rectangle
- Sprite - Displays a graphic onto the screen

3.4) Text Library

The text library provides commands to display text, characters, and numbers in the large or small fonts. Commands included in the text library are:

- LargeChar - Displays a character in large font
- LargeNum - Displays a number in large font
- LargeText - Displays text in large font
- SmallChar - Displays a character in small font
- SmallNum - Displays a number in small font
- SmallText - Displays text in small font

3.5) Math Library

The math library provides simple integer math functions for use with the built-in variable type. Commands included in the math library are:

- BoolAnd - Logical AND function
- BoolOr - Logical OR function
- BoolXor - Logical XOR function
- Dec - Decrements a variable
- Div - Divides two numbers
- Inc - Increments a variable
- Minus - Subtracts two numbers
- Mult - Multiplies two numbers
- Plus - Adds two numbers
- Rand - Generates a random number

3.6) Float Library

The float library adds a variable type for floating point numbers, and math and miscellaneous functions for use with float variables. Commands in the float library are:

- FDec - Decrement a floating point variable
- FDiv - Divide two floating point numbers
- FFPart - Returns the fractional part of a floating point number
- FIfEqu - Executes next command if numbers are equal
- FIfGreater - Executes next command if numbers in greater condition
- FIfLess - Executes next command if numbers in less condition
- FIfNEqu - Executes next command if numbers are not equal
- FIPart - Returns the integer part of a floating point number
- FLargeNum - Displays a floating point number in large text
- FLoad - Loads a value to a floating point variable
- FMinus - Subtract two floating point numbers
- FMult - Multiply two floating point numbers
- FPlus - Adds two floating point numbers

- FRand - Generates a random floating point number
- FSmallNum - Displays a floating point number in small font
- FSqRoot - Calculates the square root of a floating point number
- FSquare - Squares a floating point number
- FToVar - Converts a floating point number to an integer variable

3.7) Misc Library

The misc library provides miscellaneous, especially keyboard related, commands. Commands in the misc library are:

- Delay - Pauses the calculator for a length of time
- IfAnyKey - Executes next command if any key was pressed
- IfKey - Executes next command if a certain key was pressed
- IfNKey - Executes next command if a certain key was not pressed
- LoadKey - Loads the current key press to a variable
- LoadWaitKey - Waits for a key to be pressed then loads it to a variable
- LoadWaitTIOSKey - GetKey-style keyboard input (allows 2^{nd}/Alpha).
- PauseForKey - Pauses until a certain key is pressed
- WaitKey - Waits for any key to be pressed

3.8) Silver Library

The silver library provides commands related to the Silver Edition version of the TI-83 Plus. Commands in the silver library are:

- DelayMilli - Delays for a certain number of milliseconds (SE only)
- FastSpeed - Sets CPU to 15 MHZ if executing on Silver Edition
- IfNSilver - Executes next command if not running on Silver Edition
- IfSilver - Executes next command if running on Silver Edition
- SlowSpeed - Sets CPU to 6 MHZ

3.9) Appvar Library

The appvar library provides commands to create, read, and write data to/from application variables. Commands in the appvar library are:

- IfAppVarExist - Executes next command if this app's appvar exists
- IfAppVarNExist - Executes next command if this app's appvar doesn't exist
- MakeAppVar - Create appvar for this application.
- ReadAppVar - Read a value from this application's appvar.
- WriteAppVar - Write a value to this application's appvar.

4) Limitations

Some of these limitations are imposed by the ZDS compiler, some are limitations of this SLIDE implementation. Some will be modified or removed in future SLIDE versions, some are here to stay. In any case, here they are:

- There are only 26 available variables. (More variables can be added by changing the NumVariables setting)
- Variables are positive integer-only and can not be greater than 32767. (Use floating point variables if negative, non-integer, or larger numbers are necessary).
- There are 8 floating point variables (more can be added).
- Floating point variables are limited to $\pm 9*10^{99}$ and have a 14 digit precision.
- Sub-routine depth is limited to 10 levels. (This can be modified by changing the MaxDepth setting).

5) Frequently Asked Questions

Q. Can SLIDE be used to create multi-page applications?

A. No, the current version of SLIDE does not support multi-page applications. Multi-page support may be added in a future version.

Q. Does SLIDE include all the commands in a library automatically?

A. Yes, SLIDE includes all commands in libraries you include by default. If you would only like to use some of the commands you may remove the remainder to save space. Information this procedure is in the Tips and Tricks section below.

Q. Is this version of SLIDE complete?

A. No, this is not the final release version of SLIDE. Almost all features are implemented, but error checking is not completely implemented, and more libraries will be available for the release version. For that reason, this version of SLIDE is **NOT TO BE DISTRIBUTED**.

6) Tips and Tricks

Coming Soon!

7) Command Reference

Name of Command	Library
Description	
Syntax	
Examples/Notes	

BoolAnd	Math
Logical AND function	
Syntax: BoolAnd value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to Value1 (and in this case Value1 must be a variable)	

BoolOr	Math
Logical OR function	
Syntax: BoolOr value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to Value1 (and in this case Value1 must be a variable)	

BoolXor	Math
Logical XOR function	
Syntax: BoolXor value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to Value1 (and in this case Value1 must be a variable)	

ClearScreen	GFX
Clears the screen	
Syntax: ClearScreen	

Copy	Main
Copies data to a variable	
Syntax: Copy value1, variable	
Note: This function is the same as the Load function, but with the arguments reversed. It is included for BASIC programmers who are used to the → (STO) command.	

Dec	Math
Decrements a variable	
Syntax: Dec variable	
Example: Dec VarA	

Delay	Misc
Delays for a length of time	
Syntax: Delay value1	
Note: The length of delay is not particularly consistent (use DelayMilli if you need an exact time), but a value of 1 should be approximately 1/200 of a second.	

DelayMilli	Silver
Delays for a number of milliseconds	

Syntax: DelayMilli value1	
Note: This command is only available on the Silver Edition (it will immediately return if it is executed on the TI-73 or TI-83 Plus). DelayMilli uses the Silver Edition's crystal timer for accurate timing.	

Div	Math
Divides two numbers and store the result to a variable	
Syntax: Div value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to Value1 (and in this case Value1 must be a variable)	

Exec	Main
Executes an assembly language sub-routine	
Syntax: Exec label	
The assembly language sub-routine does not need to take any special precautions (other than overwriting variables) or actions, and should just terminate with a ret when it is done.	

FastSpeed	Silver
Sets the Silver Edition CPU speed to 15 MHZ	
Syntax: FastSpeed	
Note: This command has no effect on the TI-73 and TI-83 Plus.	

FDec	Float
Decrements a floating point number	
Syntax: FDec float_variable	
Example: FDec Float1	

FDiv	Float
Divides two floating point numbers	
Syntax: FDiv float_variable1, float_variable2[, float_destination]	
Note: If float_destination is omitted, result will be stored to float_variable1	

FFPart	Float
Returns the fractional part of a floating point number	
Syntax: FFPart float_variable	

FIfEqu	Float
Executes next command if numbers are equal	
Syntax: FIfEqu float_variable1, float_variable2	

FIfGreater	Float
Executes next command if numbers are in greater than condition	
Syntax: FIfGreater float_variable1, float_variable2	

FIfLess	Float
Executes next command if numbers are in less than condition	
Syntax: FIfLess float_variable1, float_variable2	

FIfNEqu	Float
Executes next command if numbers are not equal	
Syntax: FIfNEqu float_variable1, float_variable2	

FIPart	Float
Returns the integer part of a floating point number	
Syntax: FIPart float_variable	

FLargeNum	Float
Displays a floating point number in large text	
Syntax: FLargeNum XCoord, YCoord, float_variable	
Note: Up to seven digits will be displayed	

FLoad	Float
Loads a value to a floating point number	
Syntax: FDiv float_variable, value	
Note: Value is limited to size of integer variables (IE. Must be 32767 or less)	

FMinus	Float
Subtracts two floating point numbers	
Syntax: FMinus float_variable1, float_variable2[, float_destination]	
Note: If float_destination is omitted, result will be stored to float_variable1	

FMult	Float
Multiplies two floating point numbers	
Syntax: FMult float_variable1, float_variable2[, float_destination]	
Note: If float_destination is omitted, result will be stored to float_variable1	

FPlus	Float
Adds two floating point numbers	
Syntax: FPlus float_variable1, float_variable2[, float_destination]	
Note: If float_destination is omitted, result will be stored to float_variable1	

FRand	Float
Generates a random floating point number	
Syntax: FRand max_value, float_variable	
Note: Loads float_variable with a floating point value between 0 and max_value minus one	

FSmallNum	Float
Displays a floating point number in small text	
Syntax: FSmallNum XCoord, YCoord, float_variable	
Note: Up to seven digits will be displayed	

FSqRoot	Float
Calculates the square root of a floating point number	
Syntax: FSqRoot float_variable	

FSquare	Float
Squares a floating point number	

Syntax: FSquare float_variable

FToVar	Float
Converst a floating point number to an integer and stores it to a variable	
Syntax: FToVar float_variable, variable	
Note: value of float_variable is limited to 9999	

IfAnyKey	Misc
Executes next command if any key was pressed	
Syntax: IfAnyKey	

IfAppVarExist	Appvar
Executes next command if this application's appvar exists	
Syntax: IfAppVarExist	
Note: Does not check any data in the appvar, only checks to see if an appvar with the proper name exists	

IfAppVarNExist	Appvar
Executes next command if this application's appvar does not exist	
Syntax: IfAppVarNExist	

IfEqu	Main
Executes the next command if numbers are equal	
Syntax: IfEqu value1, value2	

IfGreater	Main
Executes the next command if numbers are in greater than condition	
Syntax: IfGreater value1, value2	

IfKey	Misc
Executes the next command if a certain key is pressed	
Syntax: IfKey key_value	
Example: IfKey skEnter	

IfLess	Main
Executes the next command if numbers are in less than condition	
Syntax: IfLess value1, value2	

IfNEqu	Main
Executes the next command if numbers are not equal	
Syntax: IfNEqu value1, value2	

IfNKey	Misc
Executes the next command if a certain key was not pressed	
Syntax: IfNKey key_value	
Note: This command could be used to make sure a user has let go of a repeating key	

(like Del)	
IfNSilver	Silver
Executes the next command if application is not executing on a Silver Edition	
Syntax: IfNSilver	
Note: TI-73's count as not Silver Edition	

IfPointOn	GFX
Executes the next command if a screen pixel is black	
Syntax: IfPointOn XCoord, YCoord	

IfSilver	Silver
Executes the next command if application is executing on a Silver Edition	
Syntax: IfSilver	

Inc	Math
Increments a variable	
Syntax: Inc variable	

InvertScreen	GFX
Inverts the screen display (black becomes white, white becomes black)	
Syntax: InvertScreen	

Jump	Main
Jumps to a label in the SLIDE program	
Syntax: Jump label	

IfSilver	Silver
Executes the next command if application is executing on a Silver Edition	
Syntax: IfSilver	

LargeChar	Text
Displays a character in large font	
Syntax: LargeChar XCoord, YCoord, character	
Note: Only characters numbered 0-127 may be used	

LargeNum	Text
Displays a number in large font	
Syntax: LargeNum XCoord, YCoord, value	

LargeText	Text
Displays text in large font	
Syntax: LargeText XCoord, YCoord, text[, color]	
Note: If color is omitted, normal black-on-white text is displayed. If color is TextWhite, white-on-black text is displayed.	

Line	GFX
Draws a line	
Syntax: Line X1, Y1, X2, Y2, color	
Note: Color should be one of: LineOn, LineOff, or LineXor	

Load	Main
Loads data to a variable	
Syntax: Load variable, value	
Note: This performs the same function as the Copy command, but is meant for assembly programmers accustomed to using the ld opcode.	

LoadKey	Misc
Loads the current key press to a variable	
Syntax: LoadKey variable	
Note: If no key is pressed a value of zero will be loaded to the variable	

LoadWaitKey	Misc
Waits for a key to be pressed then loads it to a variable	
Syntax: LoadWaitKey variable	

LoadWaitTIOSKey	Misc
Waits for a key to be pressed with GetKey-style keyboard input (allows 2^{nd}/Alpha), and then loads the key press to a variable	
Syntax: LoadWaitTIOSKey variable	

LoopToNum	Main
A looping mechanism (similar to For loops)	
Syntax: LoopToNum variable, max_value, label[, increment_value]	
Note: Variable must be pre-initialized. When a LoopToNum is encountered, it will add increment_value (default value is 1) to variable, and if variable is less than max_value, it will jump to label.	

LoopToZero	Main
A simpler and faster looping mechanism	
Syntax: LoopToZero variable, label	
Note: Variable must be pre-initialized. When a LoopToZero is encountered, it will subtract one from variable, and if variable is non-zero, it will jump to label.	

MakeAppVar	Appvar
Create appvar for this application	
Syntax: MakeAppVar	
Note: This will create an appvar with size and default data information based on the values at the top of appvar.asm. Open the appvar.asm file in your project to change these values from the defaults.	

Minus	Math
Subtracts two numbers and stores the result to a variable	
Syntax: Minus value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to value1 (and in this case value1	

must be a variable)	
Mult	Math
Multiplies two numbers and stores the result to a variable	
Syntax: Mult value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to value1 (and in this case value1 must be a variable).	
PauseForKey	Misc
Pauses until a certain key is pressed	
Syntax: PauseForKey key_value	
Example: PauseForKey skEnter	
Plus	Math
Adds two numbers and stores the result to a variable	
Syntax: Plus value1, value2[, variable]	
Note: If Variable is omitted, result will be stored to value1 (and in this case value1 must be a variable)	
Point	GFX
Draws a point	
Syntax: Point XCoord, YCoord, color	
Note: Color should be PointOn, PointOff, or PointXor	
Quit	Main
Returns to the operating system	
Syntax: Quit	
Note: This command can be executed at any time, even in a sub-routine.	
Rand	Math
Generates a random number	
Syntax: Rand variable, max_value	
Note: The generated number will be between zero and max_value minus one	
ReadAppVar	Appvar
Read a value from this application's appvar	
Syntax: ReadAppVar offset, variable	
Note: The offset is the position in the appvar to read from. Starts from a position of zero.	
Rect	GFX
Draws a rectangle	
Syntax: Rect X1, Y1, X2, Y2, color	
Note: (X1,Y1) is the upper left corner, and (X2,Y2) is the lower right corner. Color should be RectFilledOn, RectFilledOff, RectFilledXor, RectOutlineOn, RectOutlineOff, or RectOutlineXor	
Return	Main
Returns to the calling sub-routine	
Syntax: Return	
Note: If there is no calling sub-routine, this command will return to the operating system.	

Slide	Main
Calls a SLIDE sub-routine	
Syntax: Slide label	

SlowSpeed	Silver
Sets CPU speed to 6 MHZ	
Syntax: SlowSpeed	

SmallChar	Text
Displays a character in small font	
Syntax: SmallChar XCoord, YCoord, character	
Note: Characters are limited to characters with values between 0 and 127	

SmallNum	Text
Display a number in small font	
Syntax: SmallNum XCoord, YCoord, value	

SmallText	Text
Display text in small font	
Syntax: SmallText XCoord, YCoord, text[, color]	
Note: Text must be enclosed in quotation marks. Default is black-on-white text. If color is TextWhite, text will be displayed as white-on-black.	

Sprite	GFX
Displays a graphic onto the screen	
Syntax: Sprite XCoord, YCoord, graphic	
Note: Graphic should be in the following format: db height_in_pixels, width_in_pixels db graphic data	

WaitKey	Misc
Waits for any key to be pressed	
Syntax: WaitKey	

WriteAppVar	Appvar
Writes a value to this applications' appvar	
Syntax: WriteAppVar offset, value	
Note: The offset is the position in the appvar to read from. Starts from a position of zero.	

8) Contact Information

dan@detachedsolutions.com