DESIGN AND IMPLEMENTATION OF TEXAS INSTRUMENTS CALCULATOR OPERATING SYSTEMS

T.J. Liggett

Computer Science Department, Augustana University
2001 S Summit Ave, Sioux Falls, SD 57197 USA
tjliggett17@ole.augie.edu
Prepared for COSC 310 with Dan Steinwand

***Abstract*— The purpose of this project was to study the design and implementation of third-party operating systems on Texas Instruments calculators. The project consisted of both research and experimentation components. Texas Instruments calculators, their hardware specification, and independent operating systems designed for these devices were diligently researched . Experiments were then conducted to run these operating systems on both emulators and live devices. This paper gives a thorough summary of independent operating systems for Texas Instruments calculators.**

***Keywords*— open-source, operating systems, Texas Instruments calculators, Zilog Z80 processors**

I. Background

Texas Instruments calculators are the dominant calculator in both mathematics education and standardized testing. Research was done into Texas Instruments Calculators as well as independent operating systems for these devices.

A. TI Calculator Specifications

First introduced in 1999 and 2004 [1], the Texas Instruments 83 Plus and 84 Plus Graphing Calculators (TI-83+, TI-84+) are still a mainstay for mathematics education in schools across the country. A brief visit to Office Max will reveal that both still cost north of $100 in 2019, twenty years after the $125 release price of the TI-83+ [2]. The TI-83+ and its successor the TI-84+ have dominated the standardized testing and mathematics education markets for years. This dominance is result of a variety of factors, according to [3]. Teachers are typically hesitant to transition into new software as they must reformat lesson plans and reteach students. Another factor is cheating paranoia; by limiting the number of devices that are permitted, standardized tests like the ACT and SAT can better prevent cheating. This role in academia has left the hardware and software in relatively the same place it was twenty years ago.

Texas Instruments' website [1] provides specifications on individual calculators. The TI-83+ and TI-84+ run on 4 AAA batteries, with a lithium battery backup to protect RAM during the battery change. According to Nash and Olsen [4], the TI-83+ has 512 kilobytes of ROM and 32 kilobytes of RAM on the device;however, the calculator's default operating system provides the user access to only 160 kilobytes and 24 kilobytes of ROM and RAM, respectively. This speaks to the bulk of the default operating system written by Texas Instruments, which has implemented a heavier kernel design. Texas Instruments has protected the source code of their default operating system, providing little insight into its design.

TABLE I
Texas Instruments Calculator Specifications

Model	**Year Released**	**User Accessible**		**Total**	
		RAM	***ROM***	***RAM***	***ROM***
TI-83+	1999	24 KB	160 KB	32 KB	512 KB
TI-84+	2004	24 KB	480 KB	48 KB	1 MB
TI-84+SE	2004	24 KB	1.5 MB	48 KB	2 MB

The TI-83+ and TI-84+ have Z80-equivalent processors, similar to some of the first microcomputer systems according to Nash & Olsen [4]. This processor

is relatively powerful for what is required of the device. The first fully-functioning Zilog-Z80 chips were rolled out in March 1976, but the device is still used today in many embedded systems according to [5]. A Z80 equivalent chip is used in most Texas Instruments calculators. These calculators are programmable in Z80 assembly code. Since the Zilog 80 was modeled after the Intel 8080, the language is similar to that of the 8080 as well as x86 assembly.

B. KnightOS

KnightOS is an open-source Texas Instruments calculator operating system project that was started by Drew Devault in 2010, as listed on their website [6]. Available for free under in both source and binary format under the MIT license, KnightOS is incredibly developer-friendly. Many different developers have contributed to the project, with the newest contributions occurring in 2017 with decimal math support. Despite approaching a decade of development, the project has experienced setbacks due to the availability and motivation of contributors. The project still has little mathematical functionality, and the current version is of little use to non-developers.

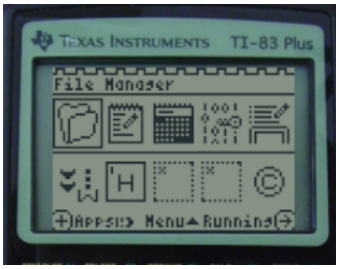

Fig. 1 Screenshot of the default KnightOS interface

Due to the open-source nature of KnightOS, it is possible to dive deeper into its structure than its Texas Instruments counterpart. KnightOS features a microkernel implementation and a UNIX-esque file system. The condensed nature of the kernel allows users much more memory than the default operating system. KnightOS is significantly smaller than the default operating system. The upgrade files for KnightOS with default packages installed are 473 KB, compared to 553 KB and 868 KB for the most recent TI-83+ and TI-84+ operating systems, respectively [7].

The KnightOS kernel handles system calls and hardware interaction, but most of the work is done in userspace. The figure below lists the dependencies of the default KnightOS system. Some of these packages include the file manager, init program, program launcher, core library, and application switcher. The Github repository [8] claims the application switcher allows users to run up to 20 programs at once. Thanks to KnightOS being open source, developers can edit any one of these packages and reinstall it using the SDK. This allows for easy customization of the userspace, as well as the kernel.

TABLE II
DEFAULT KNIGHTOS PACKAGES

Package	Description from Github [8]	Repository
bed	Basic file editor	extra
calcsys	Power user app	extra
calendar	Calendar app	community
castle	Program launcher	core
configlib	Unix-style config file management library	core
corelib	Core system library	core
fileman	File manager	extra
init	Init program	core
kernel-headers	Contains C-header files for kernel	core
periodic	Periodic table application	extra
phoenix	Classic space shooter game	ports
settings	Settings editor	core
textview	Simple text viewer	core
threadlist	Application switcher	core

C. Other Third-Party Operating Systems

While KnightOS is the clear favorite when it comes to Texas Instruments third-party operating systems, other alternatives exist. In this section, I will highlight other third-party operating systems that have made some noise. A variety of operating systems are available online at ticalc.org [9]. KnightOS appears to be the only independent OS with powerful GUI capabilities.

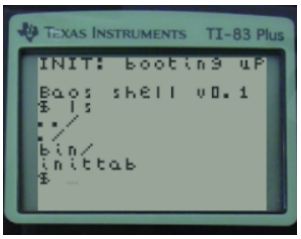

Fig. 2 Screenshot of the default BAOS interface

Among the second-tier operating systems resides Basic Assembly Operating System (BAOS) and CSX by Sean McLaughlin [9]. These operating systems provide the users with a command line similar to the Linux terminal. BAOS is a clear second to KnightOS, providing users with basic commands such as ls, cd, mkdir, cp, and rm, according to the user manual [10]. BAOS allows users to run their own Z80 assembly programs, as well as providing a basic file editor. Both BAOS and CSX are outdated projects last updated in 2008 and 2004, respectively. Many third-tier operating systems also exist as 'one-trick ponies,' where the user can play a single game or perform a single operation. A listing of these operating systems is available in Appendix F.

II. Project Details

The focus of the experimental phase was to implement and modify an independent operating system on a Texas Instruments calculator. In order to effectively achieve this goal, the experimentation process was clustered into four distinct goals for the project.

A. Update/replace the OS on TI-83+, TI-84+ calculator

Texas Instruments provides updates to the default operating system through a binary .8xu file containing the entire operating system. The operating system is small enough to justify this update process. The current version of the TI-83+ operating system upon the writing of this paper was version 1.19, and that is what was installed during the project. This updated operating system is available on their website [7] under their own end user license agreement. This license restricts users from running the operating system on an emulator, as well as reverse-assembling the binary to obtain the source code. Unfortunately, this hindered me from studying the design of the default operating system. Texas Instruments keeps this design private from the outside world.

The operating system binary can be installed on a calculator using the TI OS Downloader available for free in the TI Connect software package. While TI provides an I/O jack with the calculator, this jack only allows the user to transfer information from calculator to calculator. TI forces users to purchase the Texas Instruments Graph Link Connectivity Kit to connect the device to a computer. This kit allows USB connectivity to the calculator. To prevent software bugs from disabling the calculator entirely, TI encoded a boot independent of the operating system in the firmware. To install a software update, hold the DEL key while inserting the battery to trigger this boot. The user can then install the updated software from their system.

The binary is installed as a .8xu file. In more recently developed calculators, TI has put in security restrictions in place to prevent users from installing unverified software on their machines. However, these restrictions were not in place for calculators with a boot code of 1.02 or older, according to KnightOS' website [6]. The TI-83+ being sold in 2019 still has a boot code of 1.01 with no such security restrictions in place. Without these measures, a user can install any .8xu Texas Instruments upgrade file on their machine and provided it runs without error, run it on their machine.

I was able to successfully update my TI-83+ calculator without any hitches. However, even some of the older TI-84+ models I came across — some of which were purchased almost 10 years ago — carried a boot code of 1.03. These devices required me to install the latest operating system available from TI [7] — Operating System 255.8. While I was able to successfully update the operating system on both the TI-84+ and TI-84+ Silver Edition (SE) models, each system went through an additional validation step when updating the model. This validation step became an additional hurdle later on in the process when attempting to install an unsigned operating system.

B. Assemble KnightOS and run on emulator

There was no downloadable version of KnightOS on the project's website. I found one downloadable upgrade file on ticalc.org, but the file was corrupted when loaded onto my TI-83+ calculator. In order to create a KnightOS TI-upgrade file, the file must be

compiled using the KnightOS SDK. In addition to the SDK, the packages mktiupgrade, sass, and kimg are all required. The descriptions, dependencies, and sources for these packages are listed in Table 2 and Appendix C.

TABLE III
PACKAGES NECESSARY FOR DEVELOPING AND COMPILING KNIGHTOS

KnightOS Package	Description from [8]	Dependencies
Standard Development Kit (SDK)	Standard Development Kit for KnightOS developers	Python 3, genkfs*, kpack*
mktiupgrade	Builds Texas Instruments calculator OS upgrades from ROM dumps	cmake, asciidoc
sass	Multiplatform, Two-Pass Assembler	mono-complete
kimg	A tool for converting images to the KIMG format.	none

**Other KnightOS package. Full list of KnightOS packages used listed in Appendix C*

The entire installation process took about five hours (results may vary based on internet connection). I sped up the process after installing the first few packages by utilizing a wired internet connection. Once the installation was finished, I created both a .rom file and a .8xu file for the TI-83+, TI-84+, and TI-84+ SE models. The process I went through to assemble these files is listed in Appendix D.

To test the .rom files, KnightOS provides an emulator but I decided to use a third-party emulator on Windows for a few reasons. First, I wanted to make sure the .rom file worked independently of the KnightOS SDK, and a third-party emulator accomplished this. Second, since the TI-Connect and TI OS Downloader software were installed and running on Windows, I wanted to make sure the rom was being transferred from Bash to Windows without any corruption. I used the TiLEm Emulator [11] on Windows to accomplish this task, a Texas Instruments Z80-based graphing calculator emulator and debugger. When testing the .rom files I had created, I found no issues with the files on the emulator. This showed that any corruption with installing the upgrade files onto the live calculators would come from a problem in the assembly of the upgrade files

C. Replace TI-83+, TI-84+ operating system with KnightOS

Many obstacles occurred on my journey to install KnightOS on an actual Texas Instruments calculator. Because of these obstacles, it took countless hours of both research and experimentation to successfully install this alien operating system on a calculator. Albeit frustrating, this process allowed me to learn a great deal about both operating systems and hardware. I was able to install KnightOS on the TI-83+, TI-84+, and TI-84+ SE models. Through success and failure, I grew more during this process than during the entirety of the rest of the project.

I first attempted to install my compiled .8xu file of KnightOS onto my TI-83+ calculator. While KnightOS is default compiled onto a TI-84+, most of these calculators hold a boot code of 1.03 and provide additional protection against installing unsigned operating systems. I felt it would be easier to attack an earlier model rather than deal with these security concerns. When I transferred the KnightOS .8xu onto the TI-83+ using the TI OS Downloader Application, it appeared to successfully transfer the software. However, turning on the calculator revealed that something was not right; the operating system was installed but something was severely wrong with the display. The device seemed functional, but the screen was so distorted that it was impossible to use the calculator.

Multiple culprits for this error were identified. Since the ROM version of KnightOS seemed to function well on a third-party emulator, it was likely that nothing was wrong with the KnightOS code. The error was occurring during either the creation of the TI-upgrade file (.8xu) or the process of installing the operating system onto the machine. I compiled KnightOS in the Bash Ubuntu on Windows environment. However, the TI OS Downloader Application only has Windows and Mac versions. This created an additional process of moving the files over to Windows before transferring them to the calculator. While this process could have created the error, it seemed unlikely given the emulator was also being run on Windows and the ROM appeared to have transferred over fine.

After hours of troubleshooting, I decided to shift focus to assembling KnightOS on a TI-84+, as this is the default version that is compiled. I reasoned that eliminating the platform variable might help the

mktiupgrade KnightOS package perform correctly. My first attempt to install KnightOS on a TI-84+ was on a model with a boot code of 1.03, and I suffered the consequences. My friend and teammate Tanner Kippes provided me the model for experimentation. When attempting to load KnightOS, the calculator froze validating the operating system. When I tried to reinstall the old software, it continued to freeze in the same place. It is likely that I messed up something with the RAM with all of the operating system switching going on. After three hours of troubleshooting, I was able to clear the RAM manually by removing both the backup lithium battery and the AAA batteries at the same time. This process could have been expedited if I would have found a screwdriver faster; I spent over an hour searching for a screwdriver small enough for the device. After this reset, I loaded the original operating system back onto the device. I was able to return the device to Tanner rather than providing him the $100+ replacement.

My first breakthrough occurred when I discovered my friend Logan Swanson possessed a TI-84+ model with a boot code of 1.00. After promising him the use of my TI-83+ for his Calculus III exam and showing him physical proof that Tanner's calculator was intact, Logan agreed to loan me his calculator. With the perfect model for experimentation, I was shocked when I was able to successfully send KnightOS to the device. Everything worked exactly as it did in the emulator.

I was also able to install KnightOS onto a TI-84+ SE with a boot code of 1.03, albeit with another obstacle in the way. After hearing of my exploits, my friend Izzy Sommers donated her old TI-84+ SE model to me for this project, giving me the important clearance that I could brick the device. The freedom attached to this donation allowed me to experiment more aggressively with the device. With the default operating system still installed, I installed the application Unsigned, created by Brian Coventry, onto the device. This application patches the boot code on the calculator, removing the extra validation security. This process is dangerous as it permanently modifies the device, so any errors to this procedure could seriously harm the calculator. However, Izzy's clearance provided me a calculator I was willing to take that risk on. After patching the boot code, I installed KnightOS onto this device.

The day of my presentation for this project, a miracle occurred. I decided it would be helpful to show the failed installation of KnightOS on the TI-83+ model. I had already put the default operating system back onto the device, so I reinstalled KnightOS. To my surprise, KnightOS successfully installed onto the device. I am still unsure of what caused this successful installation. In the words of my professor Dan Steinwand, "the presentation gods must have just decided, 'let's just let him have this one!'" A few factors could have changed my fortunes. Since my last attempted installation on the TI-83+, both my Bash and Windows environments received updates. The mktiupgrade package of KnightOS uses both cmake and asciidoc, and an update to one of these packages could have helped. Through blood, sweat, and tears, I was able to install KnightOS on the TI-83+, TI-84+, and TI-84+ SE calculators.

D. Modify KnightOS

It took much longer than expected to assemble and install KnightOS onto the live devices. Thus, I was not able to spend as much time as I wanted to delve into Z80 assembly and modifying KnightOS. As it stands, I have limited knowledge when it comes to low-level programming, so the task of writing my own software in this environment was daunting given the limited scope of the project. Given more time, I would have loved to improve the mathematics operations offered on KnightOS. It is a project I might still consider in future endeavors.

KnightOS provides developers with a relatively straightforward process to develop their own packages and applications utilizing the SDK. The KnightOS SDK also provides limited C development support. I was able to design a basic hello world program using the C template provided by the SDK. This source code for this file exists in Appendix E. After writing the file, KnightOS will automatically include packages such as the core library into the package.config file if these packages are installed using a command. A more in-depth tutorial of this process is located on KnightOS.org/documentation.

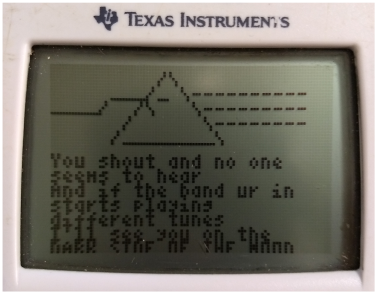

Fig. 3 floyd/main.c executed on KnightOS on TI-84+

To modify existing packages, users follow a similar process to designing their own apps, just initializing the project within a folder that contains the source code of the package found on the KnightOS Github page. Before compiling, files can be modified. During this process, I was able to modify the castle/graphics.asm file and recompile the package. Changes I made are shown in Appendix E.

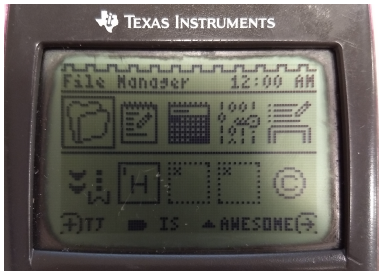

Fig. 4 KnightOS with modified core/castle package on TI-84+ SE

A calculator app written in assembly is located on the KnightOS Github but is not on the default KnightOS userspace. Developers must install this package onto their build using the SDK. The app features floating point addition and subtraction. It was a great place to start toward expanding mathematical operations for KnightOS. However, because the calculator app is not in the default userspace, I had issues compiling it. The errors I received are shown in the Figures 5 and 6.

```
tliggett@TJ-LIGGETT-AUGUSTANA:~/KnightOS/calculator/calctest$ sudo make package
[sudo] password for tliggett:
mkdir -p bin/root/bin/
sass --encoding "Windows-1252" --include ".knightos/include/;.knightos/" --listing
 bin/root/bin/main.list main.asm bin/root/bin/calculator
main.asm:15 Error: UnknownSymbol
main.asm:20 Error: UnknownSymbol
main.asm:22 Error: UnknownSymbol
main.asm:23 Error: UnknownSymbol
main.asm:25 Error: UnknownSymbol
main.asm:28 Error: UnknownSymbol
main.asm:29 Error: UnknownSymbol
main.asm:32 Error: UnknownSymbol
main.asm:33 Error: UnknownSymbol
main.asm:34 Error: UnknownSymbol
main.asm:34 Error: UnknownSymbol
main.asm:34 Error: UncoupledStatement
Assembly done: 388 ms
Makefile:7: recipe for target 'bin/root/bin/calculator' failed
make: *** [bin/root/bin/calculator] Error 12
```

Fig. 5 Errors that occurred when attempting to compile the calculator application.

```
main.asm
18      pcall(getKeypadLock)
19
20      kcall(parser_init)
21
22      kcall(init_ui)
23      kcall(redraw_ui)
24
25      kld(hl, .test_expr)
26      ld de, 0x0808
27      pcall(drawStr)
28      kcall(parse_expr)
29      kcall(eval_expr)
30
31      ld a, 0xF
32      kld(ix, result)
33      kld(hl, .output_str)
34      pcall(fptostr)
35
36      kld(iy, (screen_buffer))
37      ld de, 0x080F
38      pcall(drawStr)
39      jr main_loop
```

Fig. 6 The code causing this error. Highlighted lines caused errors.

For some reason, the sass compiler was not picking up many of the system calls being sent to the kernel. I was not able to figure out what exactly was wrong, and had to focus my energy on other areas of the project.

III. Conclusion

This paper provides insight into the design and implementation of calculator operating systems. Because Texas Instruments does not provide the source code for its operating systems, independent operating systems provide another perspective on the calculator. One can learn much about both hardware and operating systems through the design and implementation of these independent systems. The purpose of this paper is to provide a summary of my endeavors as well as a comprehensive guide to the process of installing independent operating systems on Texas Instruments calculators.

Further research could be done into the coding of the specific components of KnightOS. A code analysis of each repository would provide deeper understanding of the operating system and for calculator operating systems in general. As previously stated, the biggest thing I wish I could have done was the design of a stronger calculator application for KnightOS. In order for KnightOS to become a viable Texas Instruments

calculator operating systems, monumental steps must be made in its computational function.

Through my own blood, sweat, and tears, I was able to install an independent operating system onto my calculator and modify it. This process sucked massive amounts of mental, physical, and emotional energy from me, but through the process, I have grown as a student and programmer. For anyone looking into becoming a software engineer who would like to expand their knowledge of operating systems, I would recommend reading this paper and attempting to duplicate its results.

Appendix A: Hardware

Throughout the project, various hardware devices and resources were used. Many of the calculators used in experimentation were owned by Augustana students that were unlucky enough to be at the Froiland Science Complex at the same time as me. To show appreciation to the contributors, I have compiled a list of devices used during the project.

TABLE IV
Hardware Used During Project

Resource*	**Additional Description**	**Owner**
Dell Latitude e7470	Computer used in development	TJ Liggett
TI-83+ Calculator	Boot code: 1.01	TJ Liggett
TI-84+ Calculator	Boot code: 1.00	Logan Swanson
TI-84+ Calculator	Boot code: 1.03	Tanner Kippes
TI-84+ SE Calculator	Boot code: 1.03 (now patched)	Isabel Sommers

**No devices were harmed in the duration of the project*

Appendix B: Software

I felt it necessary to list the different software used during the project. Some of the software used for this project was not listed earlier in the paper. This appendix also provides a single place to look for software necessary for KnightOS development.

As KnightOS packages, repositories, and software are covered thoroughly throughout this paper and the other appendices, this appendix will not list everything KnightOS related. All KnightOS-related software and tutorials can be found on the KnightOS website [6] and Github [8]. KnightOS provides a standard emulator for z80 calculators, which works well with the development environment for testing and debugging. For testing and demonstration, the independent emulator TiLEm [11] was used for the project. This emulator worked very well for both TI-83+ and TI-84+ calculator .rom files.

Bash Ubuntu on Windows was used as the development environment. I installed it using the How to Geek guide [12]. This software allowed me to develop on a Windows computer using the default Linux process provided by KnightOS. Per the KnightOS website [6], listed below are the required dependencies for KnightOS development in a Linux environment:

- mono-complete
- cmake
- git
- asciidoc
- gcc
- python3-pip
- libsdl1.2-dev
- libmagickwand-dev
- libreadline-dev
- libboost-dev
- flex
- bison

These dependencies can be installed with a single sudo apt install command. More information about setting up a development environment for KnightOS can be found on the project's website.

Texas Instruments provides software for managing calculators called TI Connect [13]. This software package includes multiple programs such as a data editor, device explorer, screen capture, and OS downloader. The OS downloader was instrumental in installing KnightOS onto the calculators during this project. Another linking program, TiLP [14], allows users to transfer files through a Linux environment. TiLP is independent software and would be helpful for

those looking to emulate this project. In order to install KnightOS and other independent operating systems on Texas Instruments calculators with a boot code of 1.03 or higher, it is necessary to patch the boot code on the device. The TI calculator application Unsigned by Brian Coventry [15] was used for this process. The program allows users to patch the boot code on their device, as well as modify much of the development information (I changed the designer of my device to TJ IS AWESOME).

Appendix C: KnightOS Repositories

The table below provides descriptions of the KnightOS repositories used for this project as well as their classifications and dependencies. The table is intended to enhance the reader's understanding of KnightOS. More information about these repositories can be found on the KnightOS Github, as well as knightos.org.

TABLE V
Independent Texas Instruments Calculator Operating Systems

Repository	**Classification/Usage**	**Description from [8]**
KnightOS	OS package	The base KnightOS package. Can be compiled into default system.
bed	User space application	Basic file editor
calcsys	User space application	A power user app for KnightOS
calculator	User space application	A scientific calculator application
calendar	User space application	Calendar application
castle	User space process	Program Launcher
configlib	User space process	Unix-style config file management library
corelib	User space library	Core system library
fileman	User space application	File manager
genkfs	Development Tool	Writes a KFS filesystem into a ROM file
init	User space process	Init program
kcc	Development Tool	KnightOS C Compiler
kernel	Kernel space software	Kernel for z80 calculators
kimg	Development Tool	Tool for converting images to KIMG format
kpack	Development Tool	Manipulates KnightOS packages on POSIX systems
kpm	User space process	Package manager
libc	User space library	C Library
mkrom	Development Tool	Combines page dumps into ROM files
mktiupgrade	Development Tool	Makes TI calculator upgrade files from ROM dumps
patchrom	Development Tool	Patches jump tables into ROM files and generates an include file
periodic	User space application	Periodic Table application
phoenix	User space application	Class space shooter game
sass	Development Tool	Assembler
scas	Development Tool	Assembler and linker
sdk	Development Tool	KnightOS Standard Development Kit
settings	User space application	Settings editor
textview	User space application	Text viewer
threadlist	User space process	Application switcher
z80e	Development Tool	A z80 calculator emulator and debugger

Appendix D: Development Processes on Linux

The Linux development processes for this project are located at https://www.codetj.com/calculatorOS/ldp.pdf. These processes show the full development process of KnightOS on Linux.

Appendix E: Code Written

Code written for the project is displayed in this appendix. The complete files are included at https://www.codetj.com/calculatorOS/source.html.
I have included the modifications made to the default KnightOS software as well as program files I wrote for the project in this appendix.

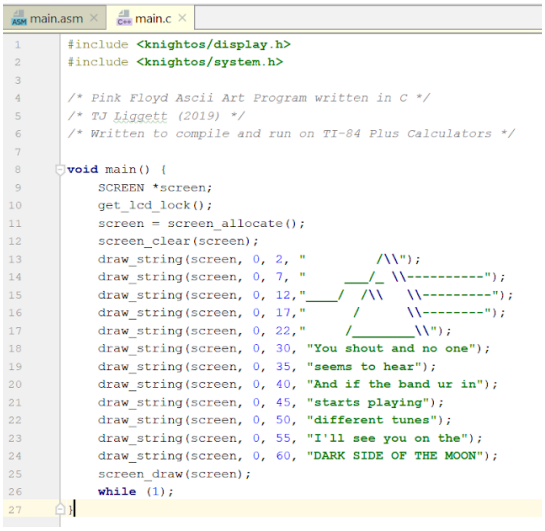

Fig. 7 An example c program written for KnightOS. Was packaged into community/floyd. For the full process of compiling and installing this package into KnightOS and onto machine, see Appendix D

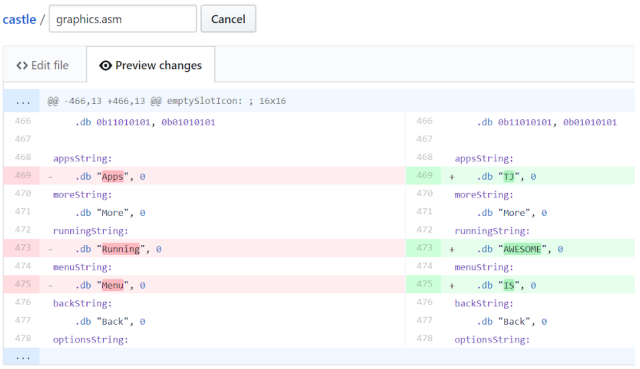

Fig. 8 Modifications made to the core/castle package, the default program launcher for KnightOS. For the full process of compiling and installing this package into KnightOS and onto the machine, see Appendix D

Appendix F: Independent Calculator Operating Systems

Independent operating systems for Texas Instruments z80 calculators as well as their descriptions and sources are listed in this appendix. Minimal research will show KnightOS is far and away from the front-runner when it comes to independent operating systems. BAOS has some merit as a command line operating system. Other operating systems carried little value outside of learning the structure of TI calculator hardware and operating systems.

TABLE VI
Independent Texas Instruments Calculator Operating Systems

Operating System	Description	Source/Website
KnightOS	Open-source, Unix-esque operating system for TI calculators	https://knightos.org/
BAOS	Basic command line operating system for z80 calculators	http://bzc.sourceforge.net/baos.html
CSX	The “third third-party OS for the TI-83 Plus.” A command line OS	https://www.ticalc.org/pub/83plus/os/
PongOS	Allows users to play pong (and only play pong) on their z80 calculators	https://www.ticalc.org/pub/83plus/os/
SmileyOS	Displays smiley face on calculator then turns off	https://www.ticalc.org/pub/83plus/os/

Acknowledgements

This research paper was completed for the Computer Science 310: Operating Systems with Professor Dan Steinwand. Professor Steinwand provided assistance when I struggled with the Linux kernel, as well as with properly compiling KnightOS. I would also like to thank KnightOS creator Drew DeVault. Drew provided assistance when I reached out to him with problems compiling KnightOS.

A special thanks goes out to the hardware contributors for this project: Tanner Kippes, Logan Swanson, and Isabel Sommers of Augustana University. Their contributions are described in Appendix A. Additional gratitude is extended to Benedict Donnay of the University of Wisconsin and Sarah Westerman of Augustana University for helping to revise this paper.

References

[1] "TI Products | Graphing Calculators | Scientific Calculators," Texas Instruments. [Online]. Available: https://education.ti.com/en/products?category=graphing-calculators. [Accessed: 21-May-2019].

[2] J. Powers, "January 10, 1996: TI-83 Graphing Calculator," Day in Tech History, 01-Jan-2019. [Online]. Available: https://dayintechhistory.com/dith/january-10-1996-ti-83-graphing-calculator-2/. [Accessed: 21-May-2019].

[3] R. O'Connell, "Why Are Graphing Calculators So Expensive?," Mental Floss, 10-Mar-2016. [Online]. Available: http://mentalfloss.com/article/73831/why-are-graphing-calculators-so-expensive. [Accessed: 21-May-2019].

[4] J. C. Nash and C. Olsen, "The Texas Instruments TI-83 Graphing Calculator," The American Statistician, vol. 52, no. 3, p. 285, Aug. 1998.

[5] R. Holland, "Chip Hall of Fame: Zilog Z80 Microprocessor," IEEE Spectrum, pp. 29–40, Jun. 2017.

[6] D. DeVault,"KnightOS," *KnightOS*. [Online]. Available: https://knightos.org/. [Accessed: 20-May-2019].

[7] Software, OS Updates and Apps. [Online]. Available: https://education.ti.com/en/software/search. [Accessed: 21-May-2019].

[8] D. DeVault, "KnightOS," GitHub. [Online]. Available: https://github.com/KnightOS. [Accessed: 21-May-2019].

[9] "TI-83/84 Plus Operating Systems," ticalc.org, 13-Apr-2015. [Online]. Available: https://www.ticalc.org/pub/83plus/os/. [Accessed: 21-May-2019].

[10] "BAOS Manual," BZC. [Online]. Available: http://bzc.sourceforge.net/baos/Manual.pdf. [Accessed: 20-May-2019].

[11] B. Moody, T. Duponchelle, and L. Bruant, "TilEm: TI Linux Emulator," Linux (TI) Programmer Group. [Online]. Available: http://lpg.ticalc.org/prj_tilem/. [Accessed: 21-May-2019].

[12] C. Hoffman, "How to Install and Use the Linux Bash Shell on Windows 10," How-To Geek, 07-Mar-2018. [Online]. Available: https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/. [Accessed: 21-May-2019].

[13] "TI Connect™ Software," Texas Instruments. [Online]. Available: https://education.ti.com/en/products/computer-software/ti-connect-sw. [Accessed: 21-May-2019].

[14] R. Liévin, "TiLP," TiCalc.org. [Online]. Available: http://lpg.ticalc.org/prj_tilp/. [Accessed: 21-May-2019].

[15] B. Coventry, "Unsigned - Downgrade from OS 2.55," ticalc.org. [Online]. Available: https://www.ticalc.org/archives/files/fileinfo/441/44190.html. [Accessed: 21-May-2019].