Propeller programming
Assembler, Spin, and Test Driven Development

Sridhar Anandakrishnan

November 10, 2016
## Contents

8.6. Testing decompression of two samples .................................. 93  
8.7. Testing decompression of 128 samples .................................. 94

### 9. Debugging PASM code  
9.1. Logging to a HUB array ............................................... 97  
9.2. Spin code ............................................................... 99  
9.3. PASM code .............................................................. 99  
9.4. Bug fix ................................................................. 100

### 10. Interacting with the world  
10.1. Outline ................................................................. 105  
10.2. Spin ................................................................. 107  
10.3. PASM ................................................................. 108  
   10.3.1. Toggle a pin in PASM ........................................... 108  
   10.3.2. Monitor a switch ............................................. 109  
10.4. Communication Protocols ........................................... 110  
10.5. SPI logging ......................................................... 110  
   10.5.1. Spin SPI read .................................................. 114  
   10.5.2. PASM SPI write ............................................... 114  
   10.5.3. Logging deadlock ............................................. 116  
10.6. Locks ................................................................. 116  
   10.6.1. Introduction to locks ....................................... 118  
   10.6.2. Using locks for logging .................................... 119  
10.7. Some common tasks .................................................. 120  
   10.7.1. Assignment ................................................... 120  
   10.7.2. Multiplication ................................................ 120  
   10.7.3. Division ....................................................... 122  
   10.7.4. Loops .......................................................... 122  
   10.7.5. Conditionals .................................................. 123  
10.8. Maintenance ........................................................... 123

### III. C Language  

#### 11. C Programming for the Propeller  
11.1. The C language ....................................................... 127  
11.2. Programming the Propeller in C .................................... 131  
   11.2.1. SimpleIDE .................................................... 132  
   11.2.2. Hello, World .................................................. 132  
   11.2.3. Launching a new Cog ....................................... 134  
   11.2.4. Compression code in C ..................................... 138
## Contents

12. Programming in Cog-C mode 143
   12.1. Cog-C Mixed Mode Programming ........................................... 143
      12.1.1. Main cog code .................................................. 143
      12.1.2. Compression Cog-C Code ........................................ 145
      12.1.3. Header file compr_cogc.h .................................... 147
      12.1.4. Running the Cog-C code ....................................... 147

13. Programming With C and PASM 149
   13.1. Compression with C and PASM ......................................... 149
      13.1.1. C code for main cog .......................................... 149
      13.1.2. PASM code ..................................................... 152

14. Hardware I/O with C 153
   14.1. Referencing hardware in C ........................................ 153
   14.2. simpletools library ............................................... 154
   14.3. Using registers ................................................... 154
      14.3.1. Set a pin .................................................... 155
      14.3.2. Read a pin .................................................. 155
   14.4. Implementing SPI in Cog C ....................................... 156
      14.4.1. Main cog (controller) ...................................... 157
      14.4.2. SPI Master .................................................. 159
      14.4.3. SPI Slave (simulated data producing device) ........... 161
      14.4.4. Running the SPI code ..................................... 162

15. Using Inline Assembly Instructions in C code 163
   15.1. Inline Assembler .................................................. 164
   15.2. spiSlave.cogc inline assembly ................................. 165
   15.3. Timing ............................................................ 165

16. Concluding thoughts 167
### List of Figures

1.1. A railroad turntable ............................................. 5
1.2. Propeller current consumption ................................. 7
1.3. Hub and cog memory layout ................................. 10

2.1. Sketch of delta compression ................................. 14
2.2. Seismic record of M9 Tohoku earthquake ................. 16

3.1. Track maintenance crew ........................................ 18
3.2. PropellerIDE windows ........................................... 22
3.3. Connecting the transcontinental railroad .................. 27

4.1. Train in snowdrift ............................................ 31

5.1. The Loop, Darjeeling Hill Railway .......................... 34
5.2. Bullock cart, Victoria Terminus, Bombay .................. 44

6.1. Manual page for rdlong ....................................... 54
6.2. Information desk at Penn Station ......................... 65

7.1. Compression flowchart ........................................ 69
7.2. Layout of memory to be passed to the PASM cog ........ 73
7.3. Civil war railway bridge ..................................... 84

8.1. Decompression flowchart ..................................... 86
8.2. Riding the cog railway down ............................... 88
8.3. Train derailment ............................................. 96

9.1. Redwood logs on a train .................................... 98
9.2. Compression flowchart with bug fix ..................... 102
9.3. Locomotive crash ........................................... 103

10.1. Switchman throwing a switch .......................... 106
10.2. Electric telegraph .......................................... 111
10.3. Logic analyzer display .................................. 113
10.4. Railway semaphores ....................................... 117
10.5. Locomotive lubrication chart ......................... 124
11.1. SimpleIDE window with Project Manager button, Build Window button (at bottom of picture) and Build button (the hammer at the top of the picture) highlighted. .................. 132
11.2. Settings for the Project Options, Compiler, and Linker tabs. ............ 133
Listings

1.1. Variable declarations in Spin ........................................ 8

3.1. Spin program template. “Hello World” .............................. 19
3.2. Changes to template.spin to include the HELLO PASM cog code. .... 23
3.3. Changes to template.spin when HELLO is in a separate file. ......... 25
3.4. Contents of hello.pasm. .................................................. 25

5.1. steim_spin_Demo: first iteration of implementing the compression in Spin. 33
5.2. steim_spin.spin: First iteration of the compression code .............. 36
5.3. steim_spin.spin: complete compressor code listing .................. 39
5.4. steim_spin.spin: Spin decompression method .......................... 40
5.5. Measuring timing of compression ....................................... 42
5.6. Clock mode settings ...................................................... 43

6.1. steim_pasm_Demo: Spin side of the first iteration of the PASM compression 55
6.2. First version of steim_pasm.spin showing the Spin methods and the beginnings of the PASM code .............................. 55
6.3. PASM fragment for passing parameters using par ...................... 59
6.4. PASM fragment showing cog memory layout ........................... 62
6.5. PASM fragment showing memory layout for passing array address in a register ........................................ 63

8.1. Decompression method in spin that triggers the PASM decompression cog 87
8.2. Decompression code in PASM; initialization ............................ 87
8.3. Decompression code in PASM; details .................................. 89
8.4. Decompression code in PASM; subroutine to save a sample to HUB after reconstruction ........................................ 90
8.5. Decompression code in PASM; subroutine to reconstruct the first sample 90
8.6. Decompression code in PASM; subroutine to reconstruct a sample .... 91
8.7. Testing the decompression code; zero and one sample ................. 93
8.8. Testing the decompression code; 128 samples initialized ............. 94
8.9. Testing the decompression code; stub showing test results ........... 94
8.10. Testing the decompression code; complete code ........................ 95

9.1. Debugging Spin code with print commands ............................ 97
9.2. Debugging PASM code by passing the address of the log buffer in a register. 97
9.3. Debugging PASM code by sharing a log buffer; Spin code ............ 99
9.4. Debugging PASM code by sharing a log buffer; PASM code

10.1. Toggle a pin in Spin

10.2. Read a switch and pin

10.3. SPI Logging, Spin side

10.4. SPI Logging, PASM side

11.1. define statements in C

11.2. include statements in C

11.3. Hello World program in C

11.4. Part 1: Front matter for file compr_cog0.c

11.5. Part 2: Compression cog code in file compr_cog0.c

11.6. Part 3: Main code in file compr_cog0.c

12.1. Contents of compr_cogc.c showing modifications from previous chapter

12.2. Contents of compr_cogc

12.3. Contents of compr_cogc.h

13.1. Contents of compr_c_pasm.c

13.2. Contents of compr.spin

14.1. Hello Blinky: using simpletools.h to toggle an LED

14.2. Setting a pin in C using OUTA register

14.3. Toggle an LED using registers in C

14.4. Contents of main file spi_c.c

14.5. Contents of header file spi_c.h

14.6. Contents of spiMaster.cogc

14.7. Contents of file spiSlave.cogc

15.1. Modified version of spiMaster.cogc that uses inline assembly

15.2. Inline assembly format

15.3. Modified version of spiSlave.cogc to use inline assembly
Preamble

This book is intended for those who are familiar with Spin programming for the Parallax Propeller but have an interest in learning Propeller C and Propeller Assembler (PASM) programming. The overall task we will pursue in the book is to implement a Delta Compression algorithm: first in Spin, then in PASM, then in C. Along the way, I talk about Test Driven Development and will end with a chapter on hardware manipulations.

Intended audience

It will be helpful to have some knowledge of the Spin programming language. The intent is to help you extend the capabilities of the Propeller processor by using the Assembler language. If you don’t know Spin, but do know another programming language (C or Python, for example), you will still be able to follow along effectively.

You can only learn by doing, so you must purchase a Propeller board such as the Quickstart board\(^1\) so that you can run the code.

Formatting

Code listings are typeset in a typewriter font: \texttt{nSamps} := 1.

In order to keep the bloat down, I often elide lines that have been explained earlier. I will insert an ellipsis to indicate that: (\ldots\).

Lines of code in Spin may not have a line break. When a long line listing is broken into two because of the page width, this is indicated with a red arrow: (\rightarrow).

The code examples can be downloaded from github (https://github.com/sanandak/propbook-code). There are two libraries (FullDuplexSerial4PortPlus_0v3 and Numbers) used in the code. Both can be downloaded from the Parallax Object Exchange (obex.parallax.com) and for convenience I include them in the github repository.

The book is typeset using the \LaTeX\ text processing macros. I encourage you to take a look at the \texttt{tikz} package for drawing, the \texttt{bytefield} package for showing bits and bytes, and the \texttt{listings} package for formatting code. The start of my latex file looks like this:

\begin{verbatim}
\documentclass[usenames,dvipsnames]{scrbook}
\usepackage{bytefield}
\usepackage{tikz}
\usetikzlibrary{matrix,positioning,trees}
\end{verbatim}

\(^1\)https://www.parallax.com/product/40000
Trains

As you will discover, I like trains! But my choice of trains as an analogue for the Propeller isn’t entirely arbitrary: like the Propeller, trains stations have many parallel tracks with trains moving at different speeds and performing different tasks... but with the need to somehow communicate between each other and with the outside world.

And there is always the possibility of a crash!

Gentle Programmer, as in any train station, you will experience tears and heartache enroute to your destination, but if you persist, there is a great reward at the other end!²

² Oh, if only Rick and Ilse had used github! (What? You don’t know who they are? Put down this book and go rent Casablanca!)