Propeller CNC Project

Started 2011 @ Don Starkey
Don@StarkeyMail.com
1/7/2013

My original design goal was to write a complete 3-axis CNC controller based on the Parallax Propeller chip.

Having some background experience in CNC retrofitting, programming and operation, I wanted to make it as easy to use as possible while including real world features. The machine I cut my teeth on was an old Kearney & Trecker VMC with an Allen Bradley controller. The control died soon after I got the beast running and I then replaced the control with a program called Camsoft. This was a PC based control that allowed you to create the user interface, interpret all the G-codes and otherwise control the machine any way you wanted. We used that machine for a number of years, later being replaced with a new Haas VF5 VMC. I never ran the Haas control but with the help of VISI-CAM, I still program more complex parts to be run on both the VF5 and a Haas SL-40 lathe.

I started the project after I wrote a high speed stepper driver for a different project based on the propeller chip. While researching the stepper driver, I came across the Bresenham's line and circle algorithms. My requirements for the stepper driver were that it was to operate fast, low latency and include acceleration and deceleration in the motion profile. Once that project was completed, I decided to start the 3-axis milling machine control based on what I had learned along the way.

The project has been stalled for quite some time due to other projects taking precedence. I have decided to post it to the Parallax community to see if others would be interested in taking it up and seeing it to completion.

As the project stands now, it is a working system but it is far from finished. See known bugs at bottom.

Some of the features are listed below.

- 3-Axis stepper driven control for a small milling machine.
- Analog feed rate override pot to adjust the feed on the fly. I like the intuitive feel of a knob over push buttons. I use 0-200% over ride but it can be from 0 to any percentage you choose.
- Over travel / home limit switches on both ends of all 3-axis.
- Runs completely on 1 Parallax Propeller.
- Uses an external VT-100 terminal (free software running on an attached notebook computer) for the user interface.
- 3-Axis linear interpolation with acceleration / deceleration.
- 2-Axis circular interpolation (with acceleration / deceleration) in XY plane.
- Limited amount of Z-axis movement during circular interpolation mode allowing for helical interpolation.
- User configurable logic states for over travel/home switches for each axis.
- User configurable logic states for Step & Direction.
- User configurable Step pulse state & width.
- Tool length offsets, Fixture offsets.
- Setup screen for Steps/Inch, Rapid Feed rate Amount, Feed rate Override Pot Amount.
- Manual JOG mode.
- MDI editor.
- CNC Program editor.
- Single step mode.
- Program storage on attached SD card.
- Program overlay stored on SD card to allow very large SPIN programs to be loaded as modules.

System Architecture

Hardware: See attached drawing: CU-3246

Software:

Due to the size of the SPIN code, the main user interface program is currently broken up into two smaller spin programs. This is in addition to 7 PASM programs that are automatically loaded for the CNC control. The main SPIN program (named CNC6.spin) is responsible for loading and running the entire control and should be saved to the EEProm using <F11>. The 2nd SPIN program (named Config.ovl.spin) is compiled separately and saved as a binary file (renamed Config.ovl) and stored on the SD memory card. This sub-program is automatically opened from the SD card and run when tool length offsets are edited or the system configuration is changed. Because of chaining back and forth, there is a noticeable time delay but allows for larger programs.

When the main program (CNC6.spin) is started, it loads all the required PASM code and displays the main menu on the attached VT-100 serial terminal. I use Teraterm, version 4.72 as the terminal emulation software on an attached notebook computer. All GUI is done using the serial terminal, all CNC code execution is done on the propeller and is not limited in any way by the speed of the terminal program or the laptop computer. The step calculation is very fast.

I use text-based menus for the navigating the different modes within the control. When necessary, sub menus are displayed. I tried to make navigating around the screen intuitive by using arrow keys on the keypad. This might be an issue since notebook computers often lack numeric keypads.

The main SPIN program allows for:

- CNC program loading, editing & execution of CNC programs.
- Editing & running MDI programs.
- Machine jogging, homing.
- Editing Offset.
- System Configuration.

There are 4 PASM programs that are the core of the CNC control. The way these programs operate is as follows. The main SPIN program parses the CNC program and passes the difference between where the machine currently is and the target location to shared memory locations along with the commanded feed rate. The movements are then calculated by a linear interpolation cog (Int_line8.spin) and a circular interpolation cog (Hybrid_circle8.spin) as required for the move. Both these cogs are loaded and sit waiting to be triggered by the main SPIN program. Once triggered the linear or circular interpolation cogs will calculate the step & direction along with the time delay between steps for the move. These step & direction signals along with the delay times are passed to a ring buffer cog that stores them and releases them to the output driver cog. By splitting the ring buffer away from the output driver, up to 300 steps can be buffered allowing for reduced or eliminated latency between moves. The acceleration/deceleration is based on values found in a table in the main SPIN program. It took too much time to do the proper floating point calculations so I use a table of acceleration coefficients and simply scale the delay for accel/decel accordingly. While a movement is occurring, the value from the feed rate override pot is read and is used to scale the commanded feed rate delay time between steps. This allows the feed rate to be adjusted on the fly. It is conceivable but not likely that changing the override pot too quickly could stall the stepper motors since no consideration is made for accel/decel based on changing override pot values. If this became an issue, it could be fixed with a simple RC filter between the pot and the A/D converter. The output driver cog takes step & direction bit patterns along with delay time values from the ring buffer cog and sets the required bit patterns to the stepper motor drivers and delays accordingly to maintain the correct feed rate. Over travel is monitored by the output driver and is optionally used to shut down all movement or just individual axis to allow for simultaneous homing of the machine. The step pins are pulsed either high or low according to how the output levels are set in the configuration menu. The pulse time is adjustable according to your stepper driver requirements. Movement values are passed to the interpolation cogs as floating point numbers. Actual step counts are passed back to the SPIN program as integer step counts from the output driver. This assures accurate reporting of where the machine actually is.

For the floating point calculations, I use Jonathan 'Lonesock' Dummer routines.
For the SD memory card driver, I use Kwabena W. Agyeman routines.

I am sure that there are more recent versions of both available. I incorporated the floating point math routine into the interpolation cogs so some careful rewriting will be necessary if changed.

I built my own logic level converter to shift from 0-3.3Vdc to 0-5Vdc as require by the stepper driver. I used a 2N2222 transistors and current limiting resistors in series with the base and collectors. I can provide schematics if necessary.

For the home / +limit switches, I use gap type optical sensors with current limiting resistors to protect the inputs of the propeller chip.

For the -limit switches, I use simple limit switches with current limiting resistors to protect the inputs of the propeller chip.

Known issues / bugs:

The major bug that I have been trying to track down is in the ring buffer routine (OutputBuffer.spin). When the ring buffer is used, there are occasional missed steps. Moving 1 inch might move 0.999 inches at 1000 steps per inch. If I disable the ring buffer by setting the buffer size to 2 (line 12 in the PASM program), it works fine, no missed steps.

I have included 6 debug variables in the main SPIN program (DebugVar1 .. DebugVar6). These variables are available in all the PASM programs for debugging purposes. They can be removed entirely.

Because of how I pass variables between programs, DO NOT REARRANGE the order of the variables if they are commented with an offset value like +40 as shown below:

```
long s_K        ' +36 Distance from Starting Z to Center of Radius along Z-Axis (Float)
long s_SPM      ' +40 Speed of movement in Inches/Minutes (Float)
long s_XAt      ' +44 Current location of X Axis (Machine position, Integer Step Counts)
```

These variables are not to be rearranged.

Any code found aligned to the LEFT side of the program is probably debug code that I have left in and could be removed. ex:

```
DecelKnown
            mov    DecelPos,tmp1          ' Where to start decelerating
            mov    Accel,#0                        ' Initialize the Accel table pointer

'wrlong STableAt, Debug1At
'wrlong STableLen, Debug2At    <--------- This is debug code that I left in the program
'mov tmp2,STableAt                        and can be removed.
'rdlong tmp1,tmp2
'wrlong tmp1, Debug3At
'a jmp #a
```

The feed rate override pot works fine but there is a software non-linearity issue that should be investigated down the road.

Wish list:

Add ^F to find within the editor.
Add file I/O error checking.
Eliminate Decel/Accel between tangent moves.
Adjustable accel by tweaking table.
Remove VT-100 and use Video/Keyboard.
Block Delete feature.
Add tool diameter offsets.
"Push Button" Interface to include hardware switches:

- Hold button
- Cycle Start
- Single Block
- E-Stop
- Reset
- Jog joystick
- Handle knob

Add 4th or 5th axis to the linear interpolation cogs to allow for extrusion type rapid prototype machines. Could remove the circular interpolation cog to free up a processor for heat control. This would be great with the feed rate override, add an override knob for the extruders.