AN3553

Core Independent Computer Numerical Control (CNC)

Introduction

Author: Alec Miller, Microchip Technology Inc.

Computer Numerical Control (CNC) machines are any machines in which programmed instructions are followed by a device to control machining tools, such as lathes, routers or drills. These types of machines have applications ranging from industrial manufacturing to hobbyist 3D printing. This application note describes the implementation of a two-axis CNC gantry that makes use of the Direct Memory Address (DMA), Numerically Controlled Oscillator (NCO) and other Core Independent Peripherals (CIPs) to handle acceleration and movement of the gantry with minimal input from the microcontroller core.

Table of Contents

1. Theory of Operation

1.1 Background

There are a wide variety of applications where CNC machines are used, including CNC routers, 3D printing, pick and place machines, and laser cutters. The motor control principles used in CNC machines can be used in other applications that require precise position control such as camera motion rigs. An actuator is any electromechanical device that can convert electrical signals to physical movement. Actuators used in CNC machines include stepper motors, hydraulic actuators and servo motors.

CNC machines are normally controlled in a format known as G-code. G-code describes, in an abstracted form, how the machine should move, how fast it should move and other operation details. For example, the G-code instruction `X10 Y20` would instruct the machine to position the end effector (e.g., a 3D printer nozzle) 10 units in the x axis and 20 units in the y axis from the origin point.

A complete listing and explanation of the G-code commands implemented in this application can be found in the G-code section of this application note. Since G-code is provided in a generic form, and the exact details of the motion (i.e., how to move the actuators) are left up to the device, the same instruction set can be used in a wide variety of devices.

1.2 Peripherals Used

The following Core Independent Peripherals (CIPs) are used in this application. Each CIP used in this application is explained in detail in the following sections of this document.

- Numerically Controlled Oscillator (NCO): Generates output frequencies that are a fraction of the input frequency and are used to control the speed of motion in this application.
- Universal Asynchronous Receiver/Transmitter (UART): The communication method between the computer and PIC® microcontroller.
- Direct Memory Access (DMA): Copies data from one memory location to another and is used to handle acceleration curves and incoming data from the UART.
- Timer 2: Used to trigger the DMA modules associated with acceleration.
- Signal Measurement Timer (SMT): Used to measure travel distance for the CNC machine.
- Configurable Logic Cell (CLC): Performs various logic operations based on their inputs. They are used in this application to allow both manual and automatic control of the stepper motors on the same pin.

1.3 Application Overview

For this application, the CNC machine will be a two-axis stepper-driven machine with a retractable pen as the end effector. G-code commands are initially processed on a PC with a program written in Python. This G-code is converted by the Python program into instructions specific to the device, which is transferred over serial communication to the PIC® microcontroller. Motion is controlled by two NCO modules slaved to a master NCO to ensure synchronized consistent movement. For instance, if the x-axis needs to travel twice as far as the y-axis, the x-axis NCO is configured with twice the output frequency as the y-axis NCO. By using multiple CIPs, much of the operation is executed with little to no work by the main processor, which will be discussed in this document.

2. Hardware Configuration

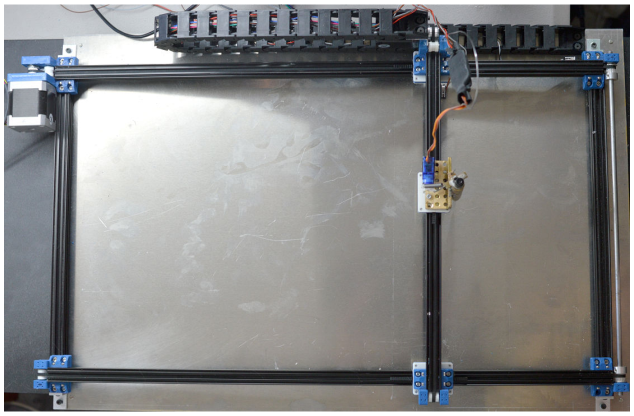
2.1 Microcontroller

The PIC18F57Q84 microcontroller is a high performance device suitable for many industrial and automotive applications. This device family features several advanced digital peripherals making it well suited for this application, including three Numerically Controlled Oscillator (NCOs), a Signal Measurement Timer (SMT) and eight Direct Memory Access (DMA) modules. Please refer to PIC18F57Q84 for more information about the PIC18-Q84 device family. For the board, the PIC18F57Q84 used was in a Curiosity Nano form factor with a Curiosity Nano Adapter expansion board.

2.2 CNC Gantry

The gantry is the frame that supports the end effector and has the components moved by the stepper motors to create motion of the end effector. The basis of the CNC in this application is a two-axis gantry made of extruded aluminum rails. A cable track is used to route the wires to prevent tangling in the machine, as shown in Figure 2-1.

Figure 2-1. CNC Gantry

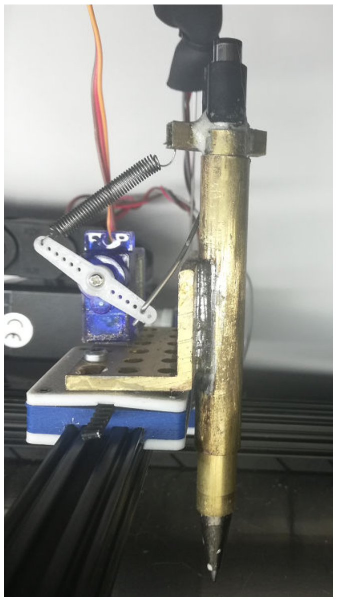


2.3 End Effector

The end effector is the component of the CNC machine designed to interact with the world. In this application, the end effector actuator consists of a pen pulled towards the work area by a spring (or pulled away from the work area by a motor), as shown in Figure 2-2. The spring ensures there is sufficient force for the pen to draw when pressed down while preventing so much force that it catches on and rips the paper.

When the actuator pin is logic high, the pen can be pulled towards the work area by the spring. When the pin is logic low, the motor is activated and causes the pen to be pulled away from the work area.

Figure 2-2. Pen Actuator



2.4 Stepper Motors

The motors used are two SY42STH47-1206A stepper motors, one for each axis. Since the stepper motors require higher voltage and current than what is supplied by the pins of the microcontroller, they are driven using the *A3967* stepper motor drivers, with one driver per motor. The drivers are controlled by enable, direction and step pins connected to the microcontroller, as described in the Pinout section of this document. When the enable pin is high, each pulse on the step pin will cause the stepper to rotate one step and in the direction determined by the direction pin.

2.5 Limit Switches

The Home section of this document describes how a limit switch is placed on both axes. The limit switches are used to help home (reset to the origin position) the platform before operation by advancing the platform in their direction until they are pressed. The switch is wired in the normally closed configuration and pulled low. This ensures the pin will be high even if the limit switch is tripped or if the limit switch is disconnected.

2.6 UART to USB Bridge

The Communication section of this document describes how a UART to USB bridge is built into the Curiosity Nano, allowing the microcontroller to communicate with the computer.

2.7 Pinout

The following tables show the pin connections used in this application. Table 2-1 shows the pin connections needed to control movement of the CNC machine on the x-axis while Table 2-2 shows the pin connections used to control movement on the y-axis. Table 2-3 shows the pin connections used for serial communication using the UART module.

Table 2-1. X Axis

Function	Pin
X Enable	RA4

...........continued

Function	Pin
Step	RA5
Direction	RD7
Limit	RC7

Table 2-2. Y Axis

Function	Pin
Y Enable	RA6
Y Step	RB4
Y Direction	RD4
Y Limit	RA1

Table 2-3. Other Connections

Function	Pin
UART TX	RF1
UART RX	RF0
End Effector	RF2

3. Motion Control Overview

3.1 Common Software-Only Method

One of the biggest challenges associated with CNC machines is ensuring smooth movement in two or more axes at a time. Consider an example where the machine has received a command to move from the start position 0,0 to end position 10,15 at a speed of 10 cm/s.

First, use the Pythagorean Theorem to calculate the total distance the actuator will travel, as shown in Equation 3-1.

Equation 3-1. Total Travel Distance

$$Total\ Distan\,ce = \sqrt{X^2 + Y^2} = \sqrt{10^2 + 15^2} = 18.03\,cm$$

Next, calculate the percent of total distance (and correspondingly, the percent of the total speed) of each axis, as shown in Equation 3-2.

Equation 3-2. Travel Percentages

$$X_{PERCENT} = \frac{X}{Total\,Distance} = \frac{10}{18.03} = 55.48\,\%$$
$$Y_{PERCENT} = \frac{Y}{Total\,Distance} = \frac{15}{18.03} = 83.21\,\%$$

After calculating the percentage of the total distance for each axis, multiply by the desired overall speed to get the speed of each axis, as shown in Equation 3-3.

Equation 3-3. Travel Speeds

$$X_{SPEED} = Desired\,Speed \times X\ Percent = 10 \times 55.48\,\% = 5.55\,cm/s$$
$$Y_{SPEED} = Desired\,Speed \times Y\ Percent = 10 \times 83.21\,\% = 8.32\,cm/s$$

The controller must then calculate how often to advance each stepper motor to achieve the two desired speeds and then use a timer to advance the axis that often. This is more challenging if, rather than moving at a constant speed, the machine must ramp up to the desired speed or slow down to stop. Therefore, the speed in the two axes must constantly recalculate as acceleration is ongoing, placing further load on the controller.

To optimize this process, CIPs can be used to limit the number of calculations and the remaining calculations can be batch-processed in advance by a computer. An inexpensive 8-bit microcontroller can handle the remaining real-time demands of operating the CNC machine, as will be discussed in the following sections.

3.2 Numerically Controlled Oscillator (NCO) Divisor

NCOs are peripherals that produce an output frequency that is a fraction of the input frequency as defined by Equation 3-4.

Equation 3-4. NCO Output Frequency

$$F_{OUT} = F_{IN} \times \frac{Increment\ Value}{2^{20}}$$

This peripheral works by incrementing a 20-bit accumulator register by the value of the increment register on every pulse of the input clock. Every time the accumulator register overflows, an output pulse is generated.

Three NCOs are used in this application: a master NCO and two slave NCOs, one for each axis. Each slave NCO toggles the step pin of the stepper drivers for the corresponding axis to produce movement. By configuring a master NCO, such that its output frequency corresponds with the desired overall speed of movement, the two slave NCOs can be configured to produce the correct proportional speed.

Given that F_{OUT}/F_{IN} is analogous to the x/y percent (axis percent) calculated using Equation 3-3, the NCO equation can be arranged to solve for the increment value that will give the desired speed proportion, as shown in Equation 3-5.

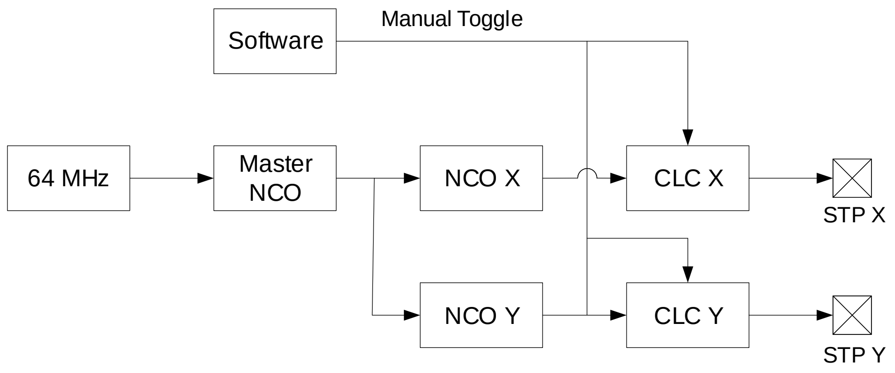
Equation 3-5. NCO Increment Proportions

$$\frac{F_{OUT}}{F_{IN}} = Axis\ Percent\ = \frac{Increment\ Value}{2^{20}}$$

$$Increment\ Value\ = Axis\ Percent\ \times 2^{20}$$

While this still requires the use of the equations in the previous section, these can be calculated in advance by a computer with the resulting increment values provided during operation, following the communication protocol described in the Communication section. Once the increment values have been loaded and the master NCO is enabled, no further action is required by the processor to control the motion. The exact timing of each step in each axis does not need to be calculated in real time.

Figure 3-1. NCO Configuration



The block diagram of the NCOs necessary for a two-axis system can be seen in Figure 3-1. The master NCO provides the desired speed while two slave NCOs reduce that to the speed required in each axis. The two output frequencies are then sent to Configurable Logic Cells (CLCs) that drive the X and Y step pins, as described in the Stepper Motors section.

CLCs are a highly flexible peripheral that allow for logic such as AND and OR gates to be applied to a wide variety of inputs. In this case, the CLCs are configured to pass the output of the corresponding NCO, with the Output Polarity bit of the CLCnPOL register used to allow the pin to be toggled by software, if needed. This is necessary in the homing process described in the Home section.

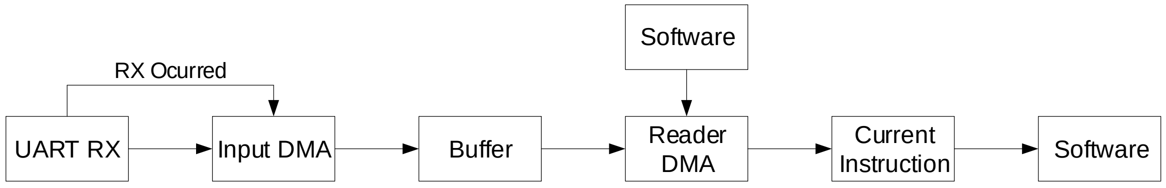
3.3 CNC Acceleration Operation

Since the speeds of the two axes are proportional to the frequency of the master NCO, a change to the frequency of the master NCO results in proportional changes to the frequencies of the two slave NCOs. This means that increasing the master NCO increment value will increase the speed of both axes while decreasing it will decrease the speed of both axes, all without the need for additional calculations.

Some of the newer PIC18 microcontrollers have the Direct Memory Access (DMA). This module helps transfer chunks of data between different memory locations without any CPU intervention. By using the DMA, this acceleration can occur without the need for the processor to copy new values into the increment register. The DMA can, when triggered, copy data from one memory location to another.

For instance, the DMA can be configured to copy data from a buffer array written to by the user to the UART TX register each time it is triggered by the UART completing transmission. In this case, the DMA can be used to copy data from an acceleration Look-Up Table (LUT) to the increment value of the master NCO to cause the machine to accelerate.

Figure 3-2. Acceleration DMA Configuration



The layout of the two DMA modules used for this application is shown in Figure 3-2. The primary DMA module feeds values from a LUT of increment values to the increment register of the NCO, which changes the speed of movement proportionally. A timer, fed by the output of the NCO, is configured to control the rate of acceleration by causing the DMA to change the increment value at set intervals.

Since the input frequency of the timer is the frequency of the NCO (as the speed is increased), the update rate from the LUT also increases. To help compensate for this, a second DMA module is used, which feeds timer period values from a LUT into the period register of the timer. By increasing the period as the frequency increases, the actual time between NCO increment increases (in terms of actual seconds) can be kept constant. This prevents any need for a higher number of samples at the faster end to compensate.

This method of acceleration requires very little work on the part of the processor, outside of initial configuration. To achieve a target speed, the DMA 'n' Source Size (DMAnSSZ) register (the number of bytes of the source data) is set to the acceleration step of the desired speed for the two DMA modules.

Example 3-1. DMA Ramp Up Configuration

```
// Source starts at location of NCO lookup table
DMAnSSA = NCO_INC_LUT_LOC;
// Destination starts at location of NCO Increment Address
DMAnDSA = NCO_INC_LOC;

// Increment source/destination location each time
DMAnCON1bits.DMODE = INCREMENT_POINTER;
DMAnCON1bits.SMODE = INCREMENT_POINTER;

// Copy bytes from source until desired location is reached
DMAnSSZ = desired_lut_index * NCO_INC_SIZE;
```

The DMA will continue transferring data until the target speed has been reached. At this point, the DMA Source Interrupt (DMAxSCTIF) will fire, indicating that acceleration is complete.

To slow down the machine at the end of travel, DMA 'n' Source Start (DMAnSSA) register (the address to start copying at) is set to the end of the location in the LUT associated with the current speed. As DMAnSSZ remains the same, the DMA is set to decrease the memory pointer copying from each transaction, rather than increment.

Example 3-2. DMA Ramp Down Configuration

```
// First byte in LUT
void * start_of_lut = NCO_INC_LUT_LOC;
// Location to start reversing from
void * location_to_copy = start_of_lut + source_size - 1;
DMAnSSA = location_to_copy;

// Start/End of NCO Inc Register
void * start_of_nco_inc = NCO_INC_LOC;
void * end_of_nco_inc = start_of_nco_inc + NCO_INC_SIZE - 1;

// Number of bytes to return to start
uint16_t source_size = current_lut_index * NCO_INC_SIZE;

// Start copying to end of location (backwards)
DMAnDSA = end_of_nco_inc;
// Copy until it gets back to start
DMAnSSZ = source_size;

// Decrement source/destination each time
DMAnCON1bits.DMODE = DECREMENT_POINTER;
DMAnCON1bits.SMODE = DECREMENT_POINTER;
```

3.4 CNC Head Travel Logic

As discussed in the Common Software-Only Method section, total distance traveled can be determined using Equation 3-1. Therefore, to ensure the distance traveled in each axis is correct, it is not necessary to measure the distance traveled in either axis. Instead, once the number of output ticks from the master NCO is equal to the total distance to travel, both axes will have traveled the correct distance. The number of output ticks is measured by connecting a Signal Measurement Timer (SMT) to the output of the master NCO.

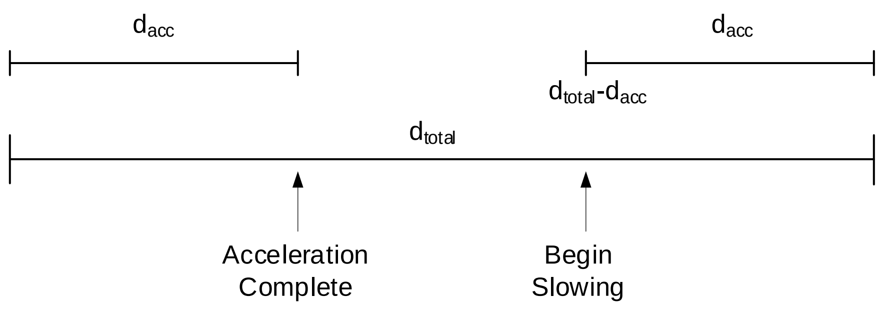
For travel with acceleration, there are two possibilities:

1. The travel distance is sufficiently long enough for the machine to ramp up to full speed, travel for a period of time and then slow back down to a speed slow enough to stop.
2. There is insufficient distance to accelerate, therefore it will need to begin slowing down before it has reached full speed.

To determine which is the case, the SMT is configured to generate an interrupt once the SMT reaches half the total travel distance. At the same time, the acceleration DMA is configured to generate an interrupt once it has finished transferring data (i.e., acceleration is complete).

In the case where the DMA interrupt fires first, this indicates that there was enough time for full acceleration. In this case, the current value of the SMT is read and this value is the distance it takes for acceleration. Therefore, it also the amount of distance it will take for the machine to slow down.

Figure 3-3. Acceleration Timing



The SMT is reconfigured to generate an interrupt when it reaches a value of the total travel distance less the acceleration distance (i.e., the interrupt will fire to start the slow down process when there is enough room left in the travel for the machine to slow to a stop), as shown in Figure 3-3. Once this interrupt is fired, the DMA is reconfigured to ramp the speed down, as described in CNC Acceleration Operation. The SMT is also reconfigured to generate an interrupt once it reaches the value of the total travel distance. When this interrupt fires, travel is complete and the master NCO is disabled.

Where the SMT interrupt set for the halfway point fires before acceleration is complete, this indicates that there was not enough time for full acceleration. In this case, the machine will need to begin to slow down immediately. The DMA is configured to begin to slow down from whatever point in the LUT it managed to reach and the SMT is reconfigured to generate an interrupt once it reaches the value of the total travel distance. When this SMT interrupt fires, travel is complete and the master NCO is disabled.

4. Control Logic

4.1 G-Code

G-code is pre-parsed by the computer and converted to values that are sent to the PIC, as described in the Communication section. The limited set of G-code commands listed in Table 4-1 are supported.

Table 4-1. G-Code Commands

Command	Argument	Description
G0	—	Go Full Speed
G1	—	Use Linear Movement
G17	—	Use XY Plane
G20	—	Use inches
G21	—	Use millimeters
G28	—	Home command
G90	—	Use absolute positions
G91	—	Use relative positions
F	Units/second	Set federate (speed)
X	Units	Travel in x axis
Y	Units	Travel in y axis
Z	`0/1`	Raise/Lower tool

4.2 Configuration Communication

There are two stages of communication: configuration and standard communication. Configuration communication occurs once when the computer requests metadata from the PIC microcontroller by sending the Initial Connection Byte (0x01). At this point, the PIC returns the metadata described in Table 4-2.

Table 4-2. Communication Metadata

Name	Size
Info Size	1 byte
Buffer Size	1 byte
Ticks per Meter	4 bytes
Max X	2 bytes
Max Y	2 bytes
Num Accel Points	1 byte
Accel Points	3 * n bytes

4.2.1 Info Size

The info size identifies how many more bytes of metadata to expect (not counting this byte).

4.2.2 Buffer Size

The buffer size identifies how many commands should be sent when requested to fill the buffer. The functionality of the buffer is discussed in Buffer.

4.2.3 Ticks Per Meter

Ticks per meter defines how ticks are required on the stepper in order to move one meter.

4.2.4 Max X and Y

Max X and Y defines the maximum ranges in ticks of the x-axis and y-axis, respectively.

4.2.5 Num Accel Points

Num accel points defines the number of points in the acceleration LUT.

4.2.6 Accel Points

Accel points is an array that contains the entire LUT of increment values. The size of this array is the value of num accel points times the size of a single value (3 bytes). By converting those increment values to ticks per second, the Python program can determine which value in the LUT to accelerate to in order to achieve the desired speed.

4.3 Standard Communication

The second stage is Standard Communication, when the actual CNC movements begin and is initiated by sending the start program byte (0x02). At this point, the PIC will respond with 0x55 to acknowledge it is ready to begin receiving data. This communication consists of a mode byte, followed by additional data (if necessary) or padding if additional data is not required. The mode byte will be one of the values in Table 4-3 and determines how the remainder of the packet is processed.

Table 4-3. Communication Types

Code	Function
0x00 – 0x03	Standard Movement
0x06	Home
0x07/0x08	Raise or Lower Pen
0xFF	End Of Transmission

4.3.1 Standard Movement

If the mode byte is 0-3, the mode is standard movement. In this state, the contents of the packet are as shown in Table 4-4.

Table 4-4. Standard Movement

Name	Size
Mode	1 byte
Distance	2 bytes
Target Speed	1 byte
X Speed	3 bytes
Y Speed	3 bytes

The mode byte of this transmission is in the range 0x00 to 0x03, or `0b00` to `0b11`. The two bits are used to indicate if travel in each axis is positive (`1`) or negative (`0`), with x taking the least significant bit. For example, if the command is to go in the negative direction in x and the positive direction in y, the command will be `0b10` or 0x02.

The distance byte indicates how long to travel for, from the perspective of the master NCO.

Target speed is the index of the target speed in the acceleration LUT.

X and Y speed are the increment values of the X and Y slave NCOs, calculated as described in NCO Divisor.

4.3.2 Home

This command homes the CNC. The remaining packet is padded to be the same size as the standard movement packet. Homing is achieved by manually toggling the CLC and driving the step pin and causing the motor to move backwards in that axis. This continues until the limit switch for that axis is pressed, advancing slightly away from the limit switch on both axes.

4.3.3 Raise or Lower Pin

If the command is 0x08, the pen is lowered. If the command is 0x07, the pen is raised. The remaining packet is padded to be the same size as the standard movement packet. However, there is a slight delay after raising or lowering the pen to give the actuator enough time to reach its target position before moving on to the next command.

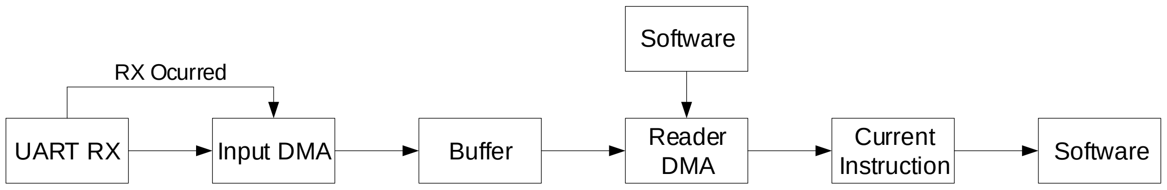
4.3.4 End of Transmission

This indicates that the routine is at its end and that the PIC should stop expecting additional data and return to Idle mode to await another routine. The remaining packet is padded to be the same size as the standard movement packet.

4.4 Buffer

A First-In First-Out (FIFO) buffer system is employed for commands to ensure that the PIC does not need to wait on the computer for additional commands and always has a supply of commands to execute. Two DMA modules are employed in the buffer system, as shown in Figure 4-1. The first block (the input DMA) copies data from the UART to the FIFO each time the UART reports receipt of data. The second block (the reader DMA) copies data from the FIFO to a dedicated memory location, as it is requested by the program.

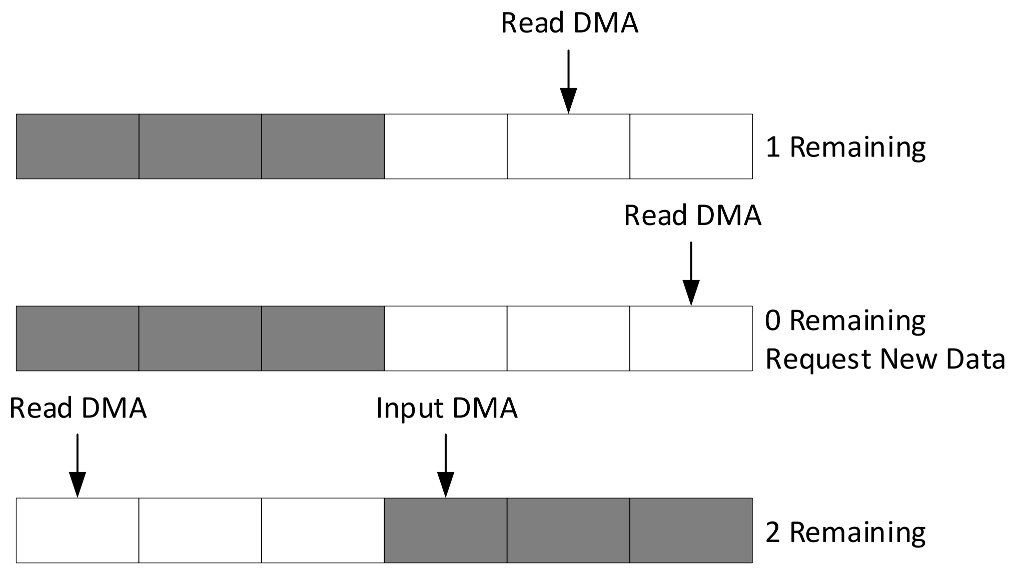
Figure 4-1. Buffer Configuration



The UART is configured using MPLAB® Code Configurator (MCC) as a standard 8-bit transmission, with no parity bit and a baud rate of 115200. Communication to the Python program was handled using the standard `UART1_Write()` function generated by MCC, which transmits the byte passed to it over UART.

While a UART is used in this application, virtually any CIP-based communication protocol could be used to feed the FIFO. The only requirement is that data is transferred a byte at a time and that the peripheral can trigger the DMA on receipt of data. For example, CAN, I^2C or reading from an SD card could all replace the UART in this application, with no changes to other program functionality.

Figure 4-2. Buffer Operation



Each transmission from the Python program sends enough data to fill half of the FIFO. This means that at any time, half of the FIFO can be safely read from by the reader DMA while the other half is written to by the input DMA. For example, if 10 instruction spots are allocated for the buffer, five instructions at a time will be transmitted by the Python program.

Following the logic shown in Figure 4-2, each time new data is requested the number of remaining instructions is decremented. Once there are no remaining instructions in that half of the FIFO, the reader DMA moves on to the other section and 0xAA is sent via UART to the Python program, requesting the next instructions. As the reader DMA continues to copy from the FIFO, the Python program sends the new instructions via DMA. The input DMA then copies this new data to the old section of the FIFO buffer.

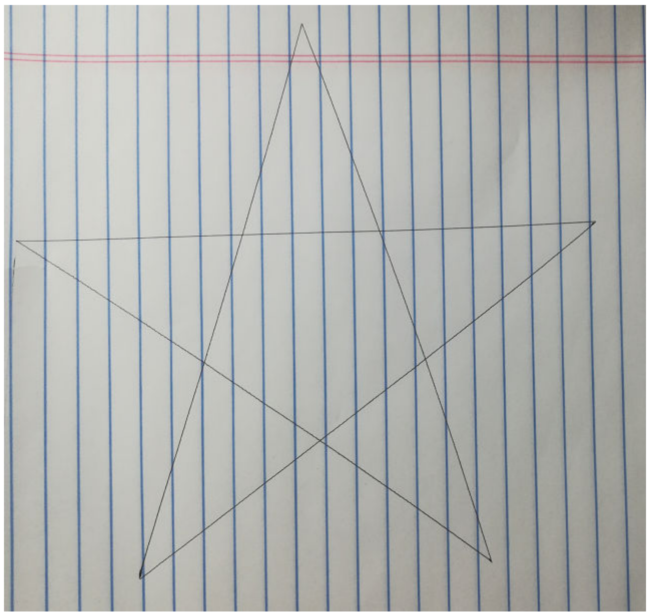
5. Conclusion

Traditional CNC control systems require extensive real-time calculations and careful timing to allow for coordinated movement in multiple axes. This in turn requires more expensive chips with floating point math capability. By making use of Core Independent Peripherals (CIPs) to handle the majority of real-time control, and pre-calculating the necessary math on an attached Python program, an inexpensive 8-bit microcontroller can achieve the same results.

When running the test G-code (as shown in Appendix C: Star G-Code), the following output (Figure 5-1) was drawn on a piece of paper by the machine.

The machine is able to draw any pattern defined in a series of straight lines. The speed of motion is variable and acceleration occurs at the start and end of motion to prevent any skipping. The ability to handle curves could be added as an improvement, possibly by providing a Look-Up Table of X and Y increment values that will achieve the desired curve from the Python GUI. At this time, the end effector is only capable of raising and lowering. An alternate system that allows for variation in intensity (e.g., a laser) could be considered.

Figure 5-1. G-Code Output



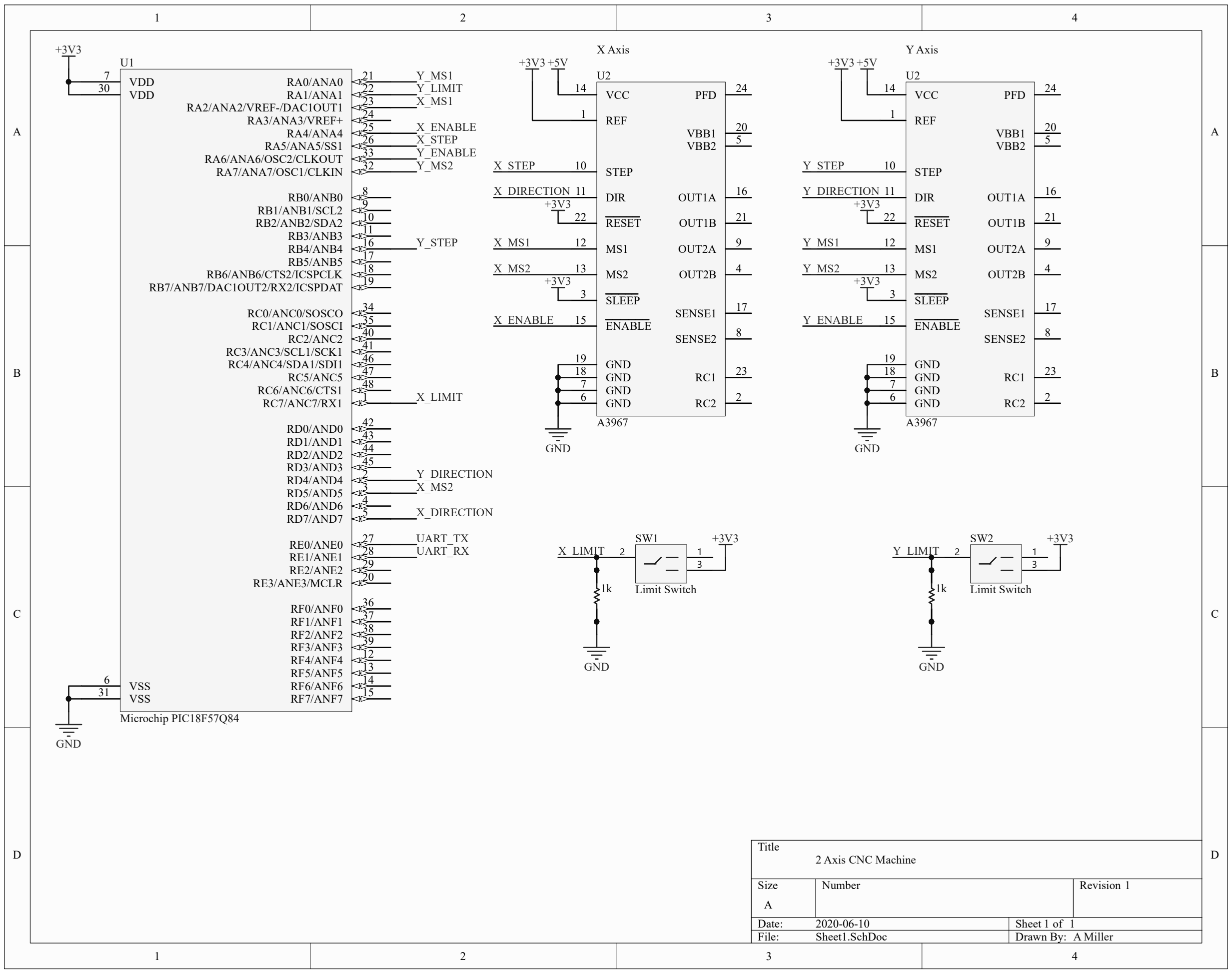
6. Appendix A: Application Code

The complete C and Python code for this project is available at:

View Code Example on GitHub
Click to browse repository

7. Appendix B: Application Schematic

Figure 7-1. Application Schematic



8. Appendix C: Star G-Code

```
G0 G21 G17 G90
Z1
G28
X9 Y122
G1 F100
Z0
X180 Y122
X42 Y22
X95 Y185
X147 Y22
X9 Y122
Z1
G0
X0 Y0
Z0
```

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Legal Notice

Trademarks

ISBN: 978-1-5224-6367-2

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra’anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820