FPGA Step and Direction Control of Motors with Ramping

Author(s):
Ryan W. Vallieu, Senior Automation Systems Architect, Data Science Automation, Inc.
Richard M. Brueggman, Founder and CEO, Data Science Automation, Inc.

NI Product(s) Used:
NI sbRIO-9626
LabVIEW FPGA 2012
LabVIEW RT 2012
LabVIEW 2012 SP1
NI RIO 12.1

Category:
Advanced Manufacturing and Control
Advanced Research

The Challenge

Perform hardware controlled Step and Direction output on 3 stepper motors with ramping to velocity set point targets while also validating encoder input feedback based on step output counts.

The Solution

The sbRIO-9626 FPGA is used to perform SPI communication to the Stepper Drive boards for setting the drive currents as well as for outputting the Step and Direction signals for the three motors.

Introduction

Data Science Automation (DSA) is a premier National Instruments (NI) Alliance Partner that specializes in automating and educating the world leading companies. Clients choose DSA because of DSA's deep knowledge of National Instruments products, disciplined process of developing adaptive project solutions, staff of skilled Certified LabVIEW Architects and Certified Professional Instructors, and unique focus on empowerment through education and co-development.

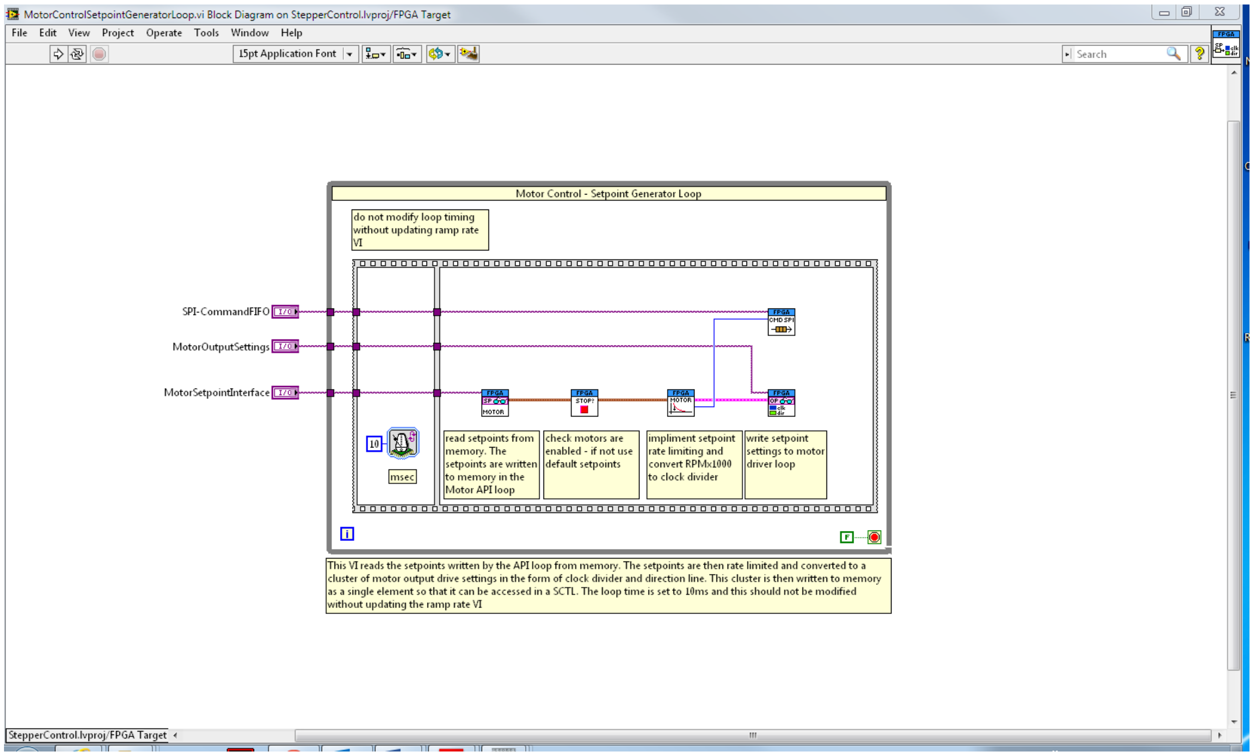
A biomedical device needed to have hardware level control of motion control outputs to stepper motors driving fluid pump heads. Software of Unknown Provenance (SOUP) where documentation could not be easily obtained was a deciding factor in moving away from the original intelligent stepper motors used in a prototype system that performed the same functionality. FMEA indicated that the motor control should be implemented on the FPGA for highest reliability and safety intervention.

The ability to implement custom communication protocols on the FPGA for SPI communication to the stepper drives was an important driver in selecting to control the motors from the FPGA. The motor drive settings are basically only set to 2 levels of power – either low current settings when the motors are not commanded to move or full available current when the motors are commanded to start moving. SPI communication is validated with the query of any parameters that are changed. If the query does not show the same value that was sent to the drive the system is put into safe mode (no motor power) and fluid motion is blocked by solenoid driven clamps.

Direction settings are easily maintained to the motor drivers. Only one motor is allowed to change directions during a tube priming phase of operation. The other two pumps are never allowed to change direction and have the direction bit hard coded into the FPGA as constant values to output on the Step and Direction lines to the motor drives.

Set points are sent from the HMI software to the Real-time executable and the RT code interacts with the FPGA for target RPM set points via front panel programmatic communication. Set points are validated against checksums to prevent corruption of communication between the RT system and the FPGA code.

RPM set points are converted to a number of clock ticks that must elapse between step outputs. The step generation code is running in a Single Cycle Timed Loop on the FPGA and the clock ticks are counted to know when the steps should be generated. Set points are sent to the motor step output clocking loop in a ramp so that the maximum RPM is not set immediately on receiving a new set point. A memory item is used to transmit the RPM clock cycle set points to the Step Generation code from the 10-ms cycle clock cycle target generation loop, MotorControlSetpointGeneratorLoop.vi (Figure 1). This is to allow for smooth ramping. Two part ramping (Figure 2) is implemented where an initial 6-point non-linear ramp is implemented and then transitions to a linear ramp until the target clock ticks (RPM) is achieved.


MotorControlSetpointGeneratorLoop.vi Block Diagram on StepperControl.lvproj/FPGA Target
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Motor Control - Setpoint Generator Loop
SPI-CommandFIFO
MotorOutputSettings
MotorSetpointInterface
I/O
FPGA
SP
MOTOR
STOP?
CMD SPI
OP
clk
dir
10
msec
i
F
StepperControl.lvproj/FPGA Target

do not modify loop timing without updating ramp rate VI

read setpoints from memory. The setpoints are written to memory in the Motor API loop

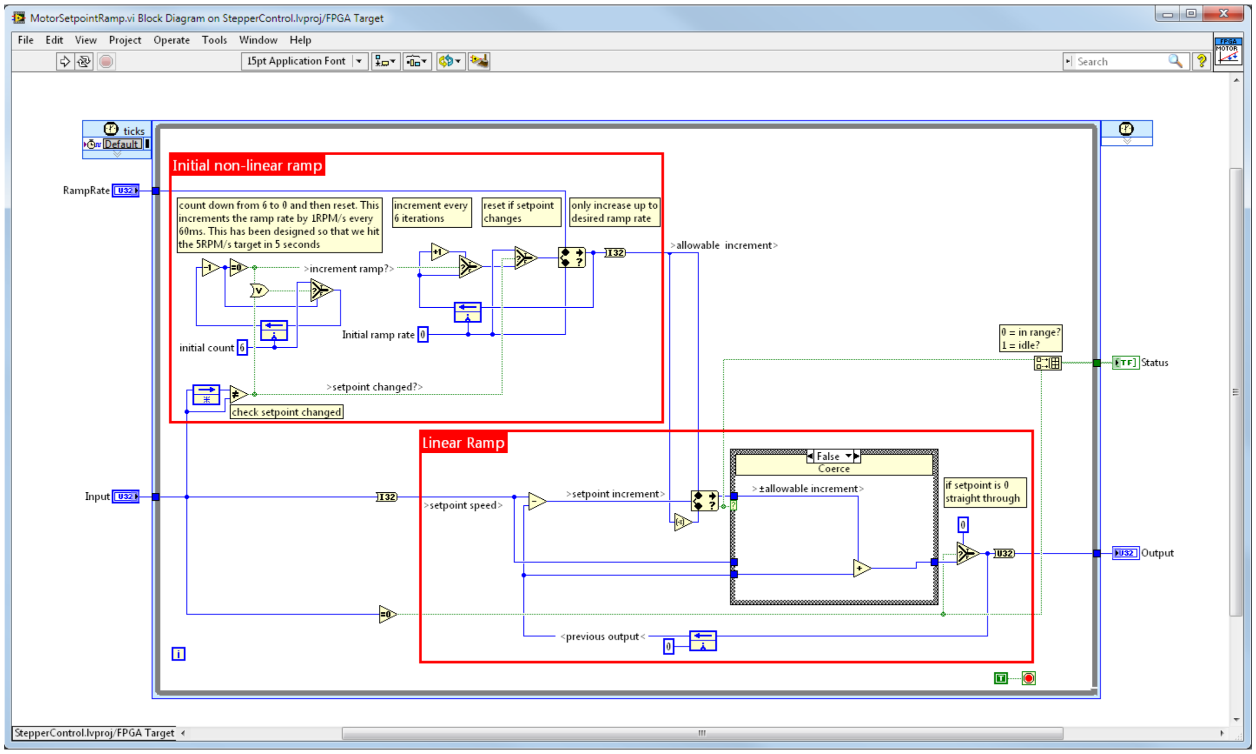
check motors are enabled - if not use default setpoints

impliment setpoint rate limiting and convert RPMx1000 to clock divider

write setpoint settings to motor driver loop

This VI reads the setpoints written by the API loop from memory. The setpoints are then rate limited and converted to a cluster of motor output drive settings in the form of clock divider and direction line. This cluster is then written to memory as a single element so that it can be accessed in a SCTL. The loop time is set to 10ms and this should not be modified without updating the ramp rate VI

Figure 1 - RPM Conversion to Clock Cycle - Ramped Targets



count down from 6 to 0 and then reset. This increments the ramp rate by 1RPM/s every 60ms. This has been designed so that we hit the 5RPM/s target in 5 seconds

increment every 6 iterations

reset if setpoint changes

only increase up to desired ramp rate

check setpoint changed

Figure 2 – Set Point Ramping Control

While the system is performing Step outputs the quadrature encoder inputs are validated against the tracked Step output counts to ensure that the motor is not stalling or falling behind the commanded motion. A known number of quadrature cycles of the 400 CPR encoders allows for checking for the proper number of encoder feedback counts per twelve output Step signals. The system in parallel with the Step output generation is decoding the Quadrature Encoder signals on the A and B channels (Figure 3 and Figure 4).

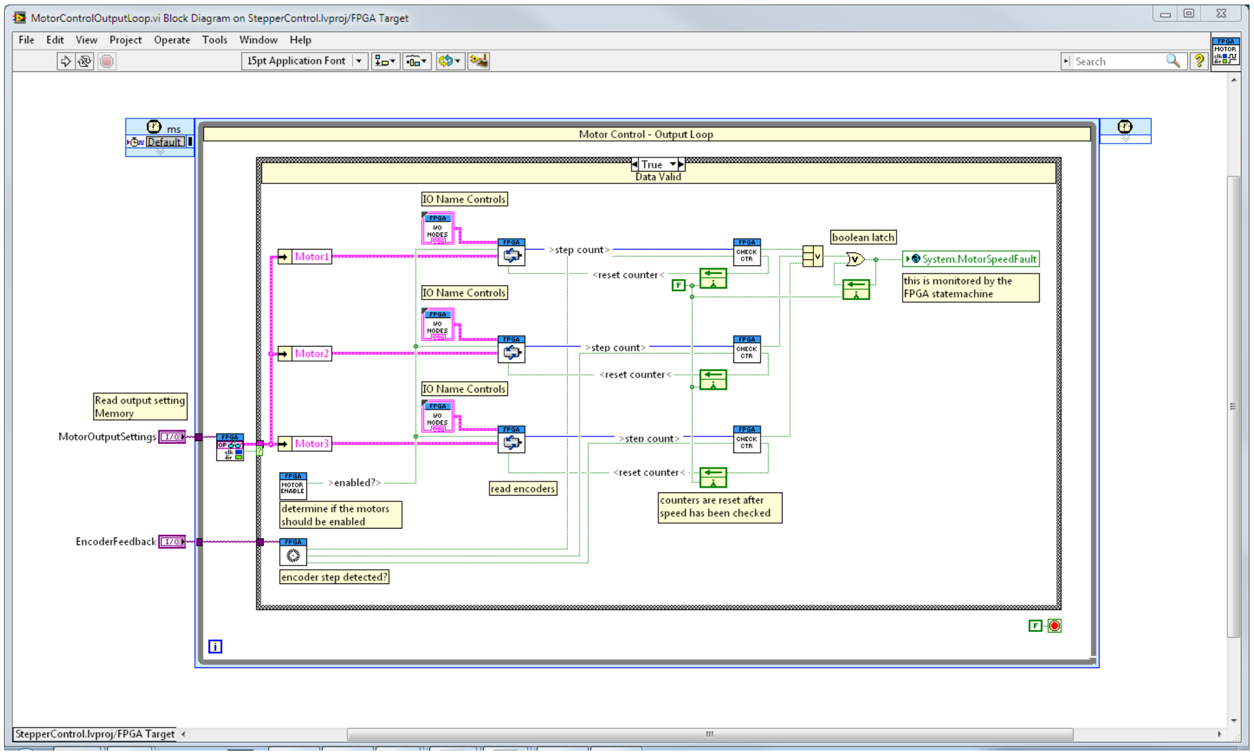
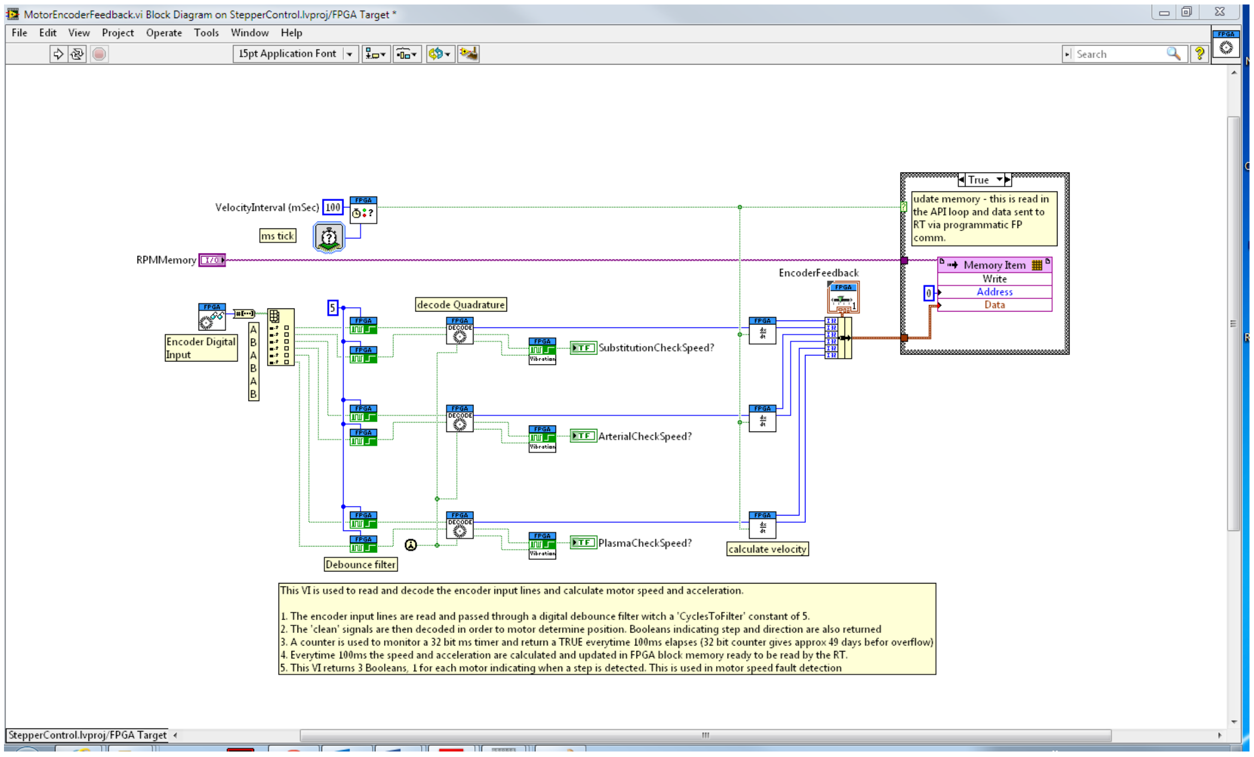

MotorControlOutputLoop.vi Block Diagram on StepperControl.lvproj/FPGA Target
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
ms
Default
Motor Control - Output Loop
True
Data Valid
IO Name Controls
Motor1
Motor2
Motor3
step count
reset counter
boolean latch
System.MotorSpeedFault
this is monitored by the FPGA statemachine
Read output setting Memory
MotorOutputSettings
enabled?
determine if the motors should be enabled
read encoders
counters are reset after speed has been checked
EncoderFeedback
encoder step detected?
StepperControl.lvproj/FPGA Target

Figure 3 - Motor Control Output Loop and Speed Check

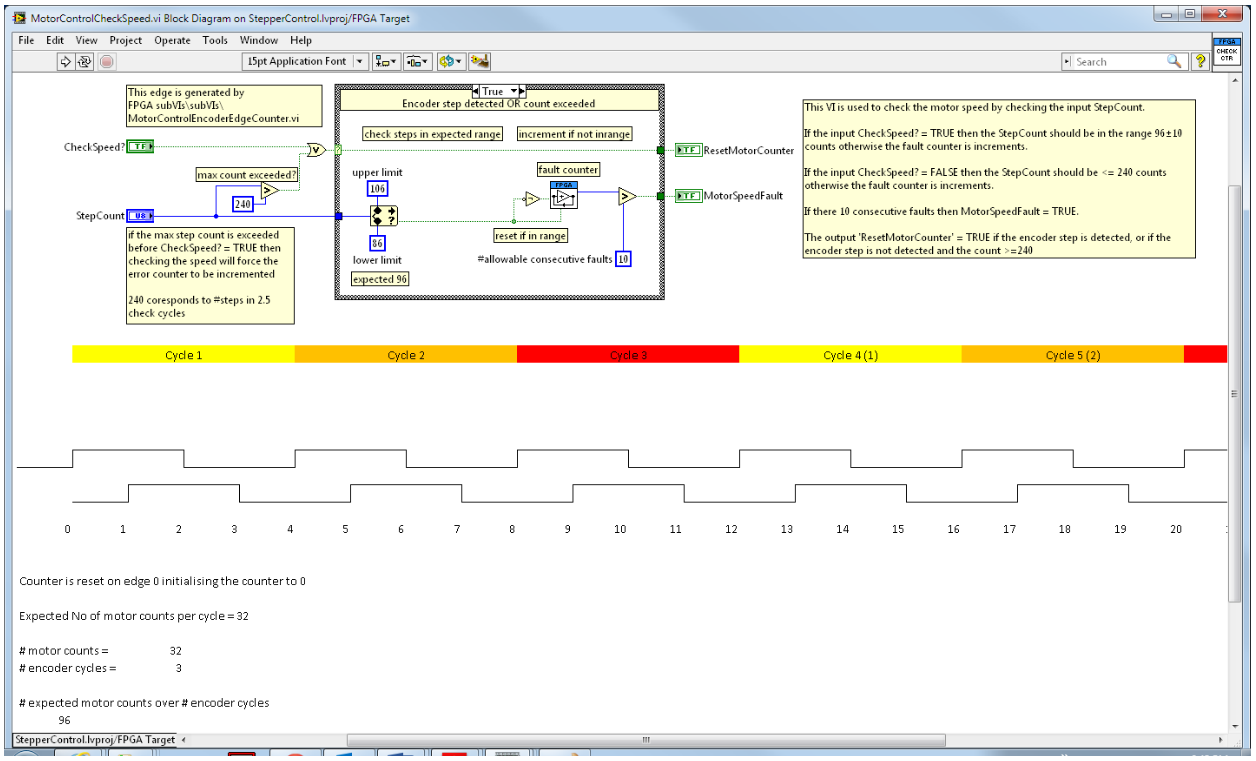

MotorEncoderFeedback.vi Block Diagram on StepperControl.lvproj/FPGA Target *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
VelocityInterval (mSec)
100
ms tick
RPMMemory
True
udate memory - this is read in the API loop and data sent to RT via programmatic FP comm.
EncoderFeedback
Memory Item
Write
Address
Data
Encoder Digital Input
5
decode Quadrature
SubstitutionCheckSpeed?
ArterialCheckSpeed?
PlasmaCheckSpeed?
calculate velocity
Debounce filter
StepperControl.lvproj/FPGA Target

This VI is used to read and decode the encoder input lines and calculate motor speed and acceleration.

1. The encoder input lines are read and passed through a digital debounce filter witch a 'CyclesToFilter' constant of 5.
2. The 'clean' signals are then decoded in order to motor determine position. Booleans indicating step and direction are also returned
3. A counter is used to monitor a 32 bit ms timer and return a TRUE everytime 100ms elapses (32 bit counter gives approx 49 days befor overflow)
4. Everytime 100ms the speed and acceleration are calculated and updated in FPGA block memory ready to be read by the RT.
5. This VI returns 3 Booleans, 1 for each motor indicating when a step is detected. This is used in motor speed fault detection

Figure 4 - Motor Encoder Feedback

In this system with 400 Counts per Revolution (CPR) quadrature encoders there are 12 encoder edge counts expected with 3 encoder cycles since they are Quadrature encoders (1 encoder cycle is 4 edges) and x4 decoding is performed by MotorEncoderFeedbackDecodeEncoderPosition.vi. Once 12 same direction encoder edge detections have been detected the "CheckSpeed?" Output is set to true. A single encoder cycle is 32 motor output steps with the 12,800 step microstepped motors and a 400 CPR encoder. Thus the expected motor output steps for 3 encoder cycles (12 encoder edge counts) is 96 motor output steps (Figure 5). Speed mismatch is allowed up to 10 times before a system fault is triggered that shuts down the motor control and prevents fluid motion.



This edge is generated by FPGA subVIs\subVIs\MotorControlEncoderEdgeCounter.vi

if the max step count is exceeded before CheckSpeed? = TRUE then checking the speed will force the error counter to be incremented

240 coresponds to #steps in 2.5 check cycles

This VI is used to check the motor speed by checking the input StepCount.

If the input CheckSpeed? = TRUE then the StepCount should be in the range 96±10 counts otherwise the fault counter is increments.

If the input CheckSpeed? = FALSE then the StepCount should be <= 240 counts otherwise the fault counter is increments.

If there 10 consecutive faults then MotorSpeedFault = TRUE.

The output 'ResetMotorCounter' = TRUE if the encoder step is detected, or if the encoder step is not detected and the count >=240

Counter is reset on edge 0 initialising the counter to 0

Expected No of motor counts per cycle = 32

motor counts = 32
encoder cycles = 3

expected motor counts over # encoder cycles
96

Figure 5 - Motor Control Check Speed

This final implementation satisfies the requirements for stall detection and smooth ramping. Other hardware options considered initially were intelligent controllers, however control from the Real-Time code did not meet the risk mitigation level required by the client and the certification team.

Rapid Prototyping Benefits of LabVIEW FPGA

The use of the FPGA on the sbRIO-9626 benefited this project by allowing prototyping of code and rapid testing of code interacting with the motion hardware. No custom ASIC hardware was needed to achieve integration of PWM step signal and quadrature encoder measurements. The speed of the encoder detection events with the FPGA Single Cycle Timed Loop was well above the maximum speed required of the system.

An added benefit of utilizing the LabVIEW Development Environment with the sbRIO-9626 for interfacing the Step outputs and Encoder signals was the ability to create test LabVIEW FPGA bitfiles for only the motion code. This allowed rapid testing with FPGA Front Panel Interactive Communication mode and greatly shortened the time needed to hone the motion code.

The LabVIEW environment also makes reuse of this code on other projects in the future very simple. As long as the FPGA IO Nodes are updated the code can be ported to multiple sbRIO and cRIO targets.

DATA SCIENCE AUTOMATION

Conclusion

LabVIEW FPGA, LabVIEW Real-Time, and LabVIEW for Windows allowed DSA to easily meet the customer requirements for providing hardware level control and speed monitoring of three separate stepper motors in a biomedical device. Using LabVIEW also allows for easy transfer of this code to other projects utilizing stepper motors with or without speed feedback verification and SPI communication. The ease of writing modular code for the FPGA carrying over modular coding techniques allows for this ease of code transfers between projects.

Contact Information

Ryan W. Vallieu, info@DSAutomation.com
Data Science Automation, 375 Valley Brook Road, McMurray, PA 15317-3370
www.DSAutomation.com