5-2010

Software Algorithms to Coordinate and Improve Voltage Sag Ridethrough Capabilities of Networked Industrial Processes

Owen Parks
Clemson University, oparks@clemson.edu

SOFTWARE ALGORITHMS TO COORDINATE AND IMPROVE VOLTAGE SAG RIDETHROUGH CAPABILITIES OF NETWORKED INDUSTRIAL PROCESSES

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Owen C. Parks
May 2010

ABSTRACT

For those who design, operate, and troubleshoot industrial processes, electric power quality is a subject that requires much consideration. Processes that use electronic sensors, actuators, and computation devices are heavily reliant on a stable, consistent input power source. When a power quality event such as a voltage fluctuation occurs, automation equipment often behaves unpredictably and causes process malfunction or failure.

Because industrial power consumers often blame their electric utility for these events, some utilities offer process susceptibility studies as a service for their customers. During a typical study, utility technicians and engineers perform in-house tests on suspect components or systems using voltage sag generating equipment. These tests determine device malfunction thresholds and establish an event failure timeline. Test results provide data for applying mitigation solutions, where the most critical or susceptible loads receive a higher priority for improvement. While effective, this approach often requires the addition of costly hardware.

This study presents novel software algorithms that coordinate and improve process ridethrough capabilities of network connected industrial processes. An add-on PC interfacing with an automation network executes a routine that detects voltage sags, performs a fast measurement of sag parameters, and determines an expected process response. Rather than implement a ‘cure all’ reaction for every disturbance scenario, mitigation routines are executed based upon the expected response. Underlying design constraints of this study are to minimize or avoid the installation of conventional ridethrough hardware and adhere to a software architecture that is unintrusive to existing controllers.

Voltage sag detection is performed with a real-time analysis of incoming voltages and is triggered from RMS voltage derivative threshold crossings. Having recognized the presence of a voltage sag, the algorithm determines the sag magnitude with a peak detection method, and can associate the measured magnitude/phase combination with previously recorded process data. Either the sag characteristics or historical process response data is then analyzed to determine the expected process response. Sags that can potentially force motor drives to trip offline cause the process to respond to an expected shutdown. Voltage sag magnitude/phasing combinations that have been shown to cause no process disruption are ignored. Combinations which have caused only instrument signal corruption and significant process variable deviations trigger the mitigation routine to address faulted control signals only. Drive fault mitigation responses consist of a software-only drive coast routine and an improved drive coast routine requiring the addition of basic switching hardware. Out of tolerance process errors are mitigated with output control command substitution or input signal substitution routines.

Verification of software functionality is achieved with an experimental automated process -- a textile unwind/rewind system that operates at a controlled linespeed and tension. Detailed analysis and simulation is performed on both component and system-wide levels. Unmitigated and mitigated process voltage sag responses are recorded and matched with the theoretical process model. Although customization is required to apply the algorithms to the specific design of the textile tension control process, experimentation with this test bed system serves as a satisfactory proof of concept for the software routines. As a result, the methods developed in this study can improve the task of process power quality mitigation by customizing solutions for individual processes, avoiding the application of power quality mitigation solutions where they are not required, coordinating corrective actions by utilizing existing automation network functionality, and ultimately reducing the need for costly hardware installation and maintenance.

DEDICATION

Dedicated to the memory of Katherine ‘Kay’ Parks, 1916-2006.

ACKNOWLEDGEMENTS

Although doctoral research requires intense individual study and self-motivated efforts, this work would not be a reality without the assistance of others. I am deeply indebted to:

my parents, Robert and Virginia Parks, for their steadfast love and nurturing throughout my entire life;

Dr. Michael A. Bridgwood, my research advisor, for his guidance and wisdom during all of my academic pursuits at Clemson;

Dr. E. Randolph Collins, Jr., Dr. Adly A. Girgis, and Dr. James K. Peterson for their advice and assistance while serving as research committee members;

my colleagues, both past and present, in the Clemson University Power Quality Laboratory for their aid and companionship during the countless hours spent on coursework and research;

the people of Schneider Electric/Square D Company of Seneca, SC for their unwavering encouragement during the final stages of my graduate work; and

Robert L. Morgan and Duke Power Company for their generous support, and for providing essential insight into the practical applications of this research.

Owen C. Parks

May 2010

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF TABLES

List of Tables (Continued)

Table Page

LIST OF FIGURES

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

Figure Page

List of Figures (Continued)

Figure | Page

List of Figures (Continued)

List of Figures (Continued)

Figure Page

CHAPTER 1
INTRODUCTION AND INVESTIGATION SUMMARY

Power Quality Background

As electronics become increasingly prevalent in many applications, effective device or process operation becomes more dependent on the quality of incoming electric power. Test studies have shown that even commonplace devices such as digital clocks [1,2] and personal computers [3,4] are susceptible to malfunctions caused by power source magnitude and waveform disturbances. For some consumers of electricity, power quality issues may not be more than an annoyance. However, because of their scale and dependence on electronics, commercial and industrial consumers face potentially large losses of time and money caused by power quality events.

Events and Effects

The effects of power quality events are widely dependent on their characteristics. IEEE Standard 1159-1995 defines and categorizes power quality events into three major categories of transients, waveform distortions, and voltage variations [5]. Transients are brief distortions in a waveform that do not occur periodically. A common example of a power system transient is a brief supply overvoltage caused by utility capacitor switching, that in turn can cause equipment to fail or malfunction [6]. Waveform distortions occur on a periodic basis, and are identifiable as a consistent deviation from a true sine wave in a supply voltage. Waveform distortions include DC offsets, waveform notching, harmonics, interharmonics, and electrical noise [5]. Their effects, such as clock drift [1] and premature equipment failure due to overheating [7] are more evident in the long term. Voltage variations occur in the steady state as sustained undervoltages and overvoltages, and on a transient basis as voltage sags and swells [5].

Consumers are susceptible to power quality events in numerous ways. For facilities such as hospitals, uninterrupted power is of critical importance. This mandates backup generation and smooth source transfer capabilities as part of the site power distribution design [8]. Customers who are heavily reliant on information technology equipment generally find that wiring and grounding errors are the principal causes of their power quality problems, yet experience additional losses due to interruptions and deep voltage sags [9]. Computers and process control equipment often utilize internal protection against low operating voltages, which increases their sensitivity to brief disturbances [10], and in turn leaves their users vulnerable to incur the costs associated with their malfunctions. These costs can include factors such as lost raw materials, lost production time, and even damaged equipment [11].

Voltage sags

Voltage sags receive close attention as troublesome power quality events, and are considered among the most important power quality problems facing industrial and commercial customers today [12]. Voltage sags are defined as a drop in voltage magnitude from between 10% to 90% of the nominal value, for a duration of one-half cycle to one minute [5]. Figure 1.1 shows an example of a 20%, $300ms$ voltage sag waveform. Most frequently, voltage sags are caused by lightning induced single line to ground faults [13]. They may also be caused by other factors such as high current motor starting [14] and miscellaneous power system faults due to traffic/construction accidents, animal contact, or tree contact [15]. Voltage sags have many quantifiable characteristics, which can be influenced by factors such as loading conditions, power system network topologies, and the presence of embedded generation [16]. In addition to magnitude and duration, voltage sags possess characteristics such as phase angle shift, point of wave of inception, and point of wave of recovery [17]. Furthermore, dynamic loads such as motor operation during and after a voltage sag can influence a sag's transient magnitude [18]. Even though many factors contribute to voltage sag properties, they are sometimes predictable when caused by certain events such as remote distribution fault clearing [19].

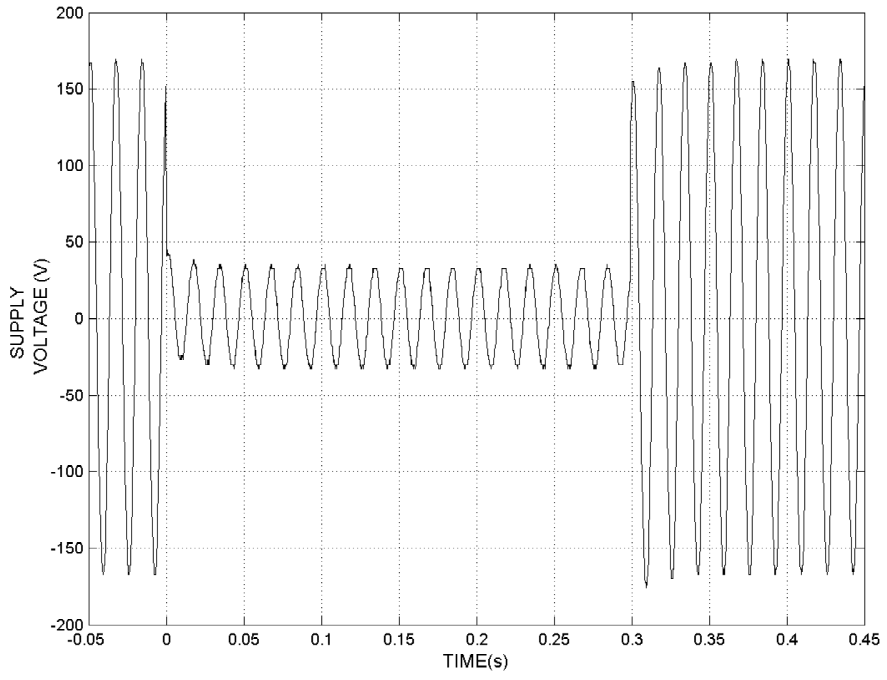


Figure 1.1 Example voltage sag waveform (20%, 300ms).

Voltage Sags and Process Manufacturing

Because of their complexity, industrial processes are particularly vulnerable to voltage sags. When a process has no response to a voltage sag, it is said to 'ride through' the event. Otherwise, one of three scenarios will generally occur: the process may shut down even though operating parameters remain within specification, the process may shut down by exceeding process parameter failsafe levels, or the process does not shut down but instead produces an out of specification product [20].

Many types of processes behave in this manner. Textile handling systems, for example, are highly susceptible to voltage sag disturbances. A review of over 60 textile facilities performed by the Electric Power Research Institute (EPRI), Duke Power, Carolina Power and Light, Georgia Power, and Northeast Utilities showed that every segment of the textile industry is susceptible to power quality problems [21]. Furthermore, an increased need to compete in global markets has fueled the need to increase textile automation, which in turn increases vulnerability throughout the industry [21]. Other industries that rely heavily on processes, such as electronics manufacturing and paper production, are also highly vulnerable to voltage sags [22, 23]. In industries that rely on extrusion operations, where cleanup and restarting are complicated procedures, costs per event can be on the order of \$10,000, with 20-25 events occurring per year [24]. Even non-continuous processes such as computer numerical control (CNC) operations can fault or produce out of tolerance products due to voltage sags [20].

Some commonly used devices are often blamed for process malfunctions. Adjustable speed drives can decrease their output speed during a voltage sag, and reacceleration times may be lengthened by current limitations intrinsic to the drive [25]. AC contactor motor starters can cause complete process shutdowns because of their susceptibility to voltage sags, and are sometimes the weakest link in an entire process [26]. Their ridethrough behavior is also sensitive to the point on wave of voltage sag inception [27], which makes identifying their dropout threshold a complicated task. Programmable logic controllers (PLC) have several aspects of sag vulnerability. A PEAC study of PLC susceptibility showed that a PLC power supply can cause PLC dropout (and subsequent process shutdown), and standard 120VAC discrete inputs can be misinterpreted as false control signals [28]. False output signals can also be issued by a PLC that is stricken by a voltage sag [29].

Evaluating a process's susceptibility to voltage sags requires knowledge of both the process mechanics and the equipment in use. Understanding process variables and how they are affected by voltage disturbances is a critical part of an overall study of system susceptibility [20]. For process equipment, it is recommended that susceptibility studies establish a device dropout hierarchy, and compare that hierarchy to the number of times per year sags of a particular magnitude and duration will occur [30, 31]. This approach encourages troubleshooters to consider the likelihood that a device will fail along with the economic factors that are involved in improving that device's ridethrough capabilities.

Studies of common information technology equipment have been performed by the Information Technology Industry Council (ITI, formerly Computer & Business Equipment Manufacturers Association), and have yielded information regarding typical responses to voltage disturbances of varying magnitudes and durations. These are graphically represented in the 'ITI Curve', which is shown in figure 1.2. This curve defines different regions of operation for disturbances of varying magnitude and duration. For voltage sags, where the nominal voltage decreases below 90%, tolerance of at least one cycle for even the deepest sags is common. At higher magnitudes, equipment can often ride through sustained sags. When a device does not ride through, it is said to operate in the 'no damage region', which indicates dropout or malfunction, but leads to no lasting damage [32].

Using a graphical representation similar to that seen in the ITI Curve, test results on process equipment may be plotted to indicate their voltage sag susceptibility in relation to other process devices. The availability of high power output, on-demand voltage sag generators makes process-wide sag testing now possible. This test equipment employs specialized field excitation of a synchronous generator, or connects an intermediate tap switching transformer between source and load under test [33]. Using sag generators to test equipment is an effective way to determine device sensitivity, and repeated testing over a range of magnitudes and durations can define a device's dropout thresholds as they relate to other devices or standards [34].

For complex processes such as those found in the textile industry, determining the sensitivity of component parts is considered an effective method of establishing overall process susceptibility. Solutions first aim at improving the voltage sag tolerance of the weakest and most critical process components, with each solution implemented following an analysis of its economic viability [35]. Device interaction can also yield important information about process susceptibility. For example, sag response case studies have been successful in predicting a process sag response by breaking down a process into interconnected subsystems, then tracking the disturbance as it propagates through them [36,37]. This helps determine each subsystem's contribution to the response without treating components as isolated entities.

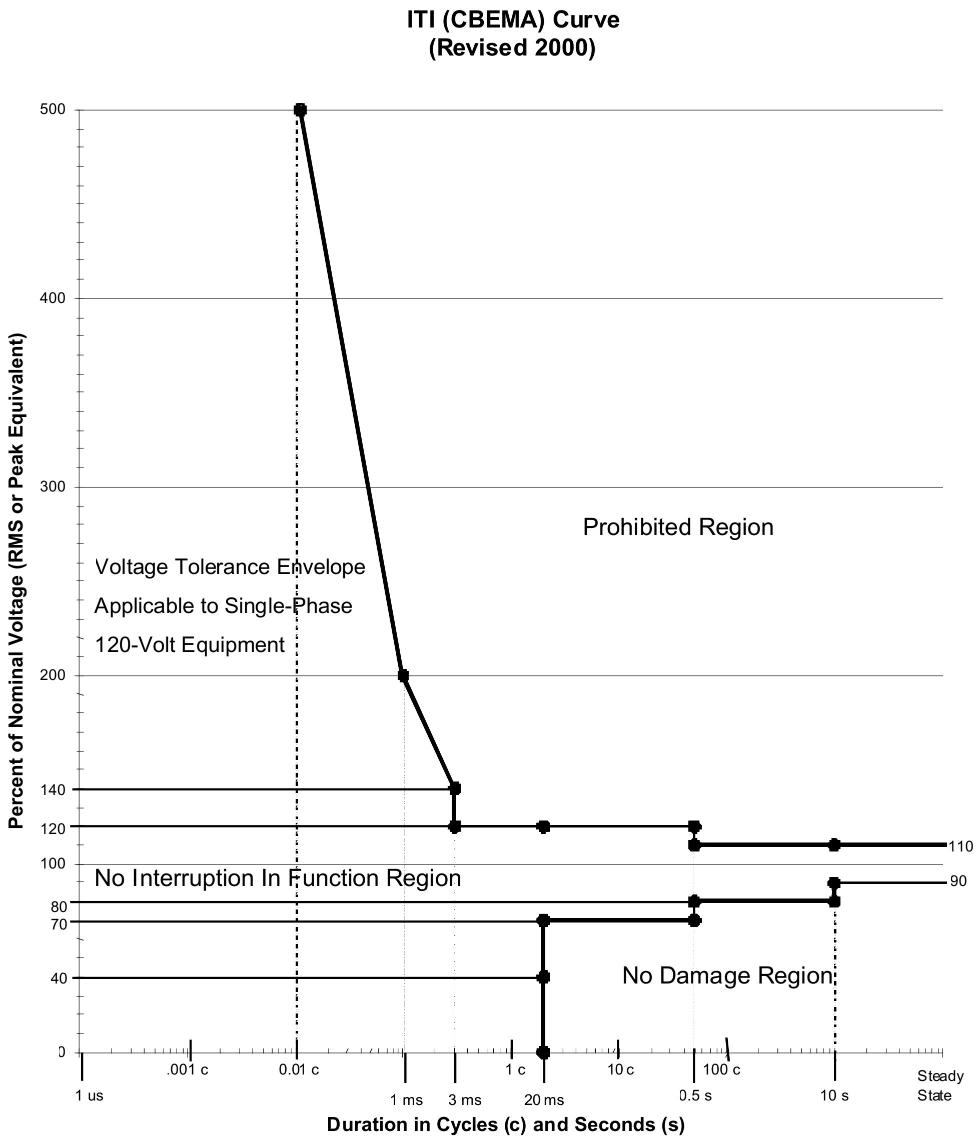


Figure 1.2 ITI (CBEMA) curve of typical information technology equipment tolerances to voltage disturbances [32].

Service, Education, and Research

Electric utilities face both challenges and opportunities when addressing power quality issues. In a deregulated electric power industry, customer service is increasingly important for utility business. This provides motivation for utilities to actively address customer power quality issues [38]. Customer complaints may be handled by power quality service programs that offer diagnostic and repair services that operate on both the utility and customer sides of the power system. Additionally, promotion of research and development is considered a valuable aspect of a utility power quality program because laboratory results may be regularly transferred to utility technicians to provide them with a stronger knowledge base to work from [39].

Even in a service environment hospitable to addressing customer complaints, there exists a growing understanding that solving power quality problems is the shared responsibility of utilities, electrical systems designers, installers, equipment manufacturers, and end users [40]. Equipment manufacturers must understand the power quality environment so they may effectively comply with specified ridethrough requests, or simply improve product ridethrough in general [41]. When constructing and upgrading facilities, systems designers and process integrators who are aware of power quality issues may alter purchasing decisions by weighing the added cost of built-in equipment ridethrough against projected downtime costs [42]. When implementing a mitigation solution, end users must take economics into consideration by accounting for factors such as purchase costs, installation costs, maintenance costs, and remaining unsolved downtime costs [43].

Conventional Mitigation Solutions

Power quality mitigation solutions may be implemented in many ways. From the utility side, the number of customer voltage sags can be reduced by improving fault prevention practices and modifying fault clearing methods [24]. Use of equipment such as electronic tap changers can compensate for loss of voltage, but these devices often have an operational delay which exceeds the ridethrough times of sensitive loads [10]. A utility end approach to power quality mitigation does have limitations. For example, mitigation of sags caused by faults at transmission voltages is considered impractical on the utility side because it requires protecting transmission lines that may be extremely long or outside a utility's property. For these types of sags, mitigation solutions are best implemented on a customer or device specific level [44].

Customers may install specialized equipment to improve the quality of their power. For very critical loads, devices such as static transfer switches (STS) or uninterruptible power supplies (UPS) may be used. Static transfer switches can compensate for a complete loss of power by switching to a secondary source using electronic switching [45]. STS use requires a reliable backup power source that is capable of meeting its downstream load demands. This source may be a standby power generator or battery bank. Uninterruptible power supplies use battery banks for their backup energy source, and may be switched online using an STS scheme, or remain online at all times [46]. UPS protection works well for low power requirements, but becomes economically unfeasible as the power demands and subsequent battery maintenance costs increase [10].

Devices designed for voltage sag mitigation are not required to supply power during a sustained interruption, and therefore only require stored energy to supply loads briefly. A simple ridethrough technology is the motor-generator (MG) set, which has relatively high efficiency and low initial cost [24]. The energy stored in a large rotating mass keeps momentary voltage disturbances from significantly affecting the mass's rotational speed, and hence the output power produced by the generator. An additional advantage of MG set usage is that clean power is produced on-site, which blocks waveform distortions from being passed through from the utility. Even though MG sets can provide significant ridethrough, their use is primarily seen in industrial environments because of their size, maintenance requirements, and noisy operation [10].

Series voltage controllers (SVC), also called dynamic voltage restorers (DVR), use capacitive energy storage rather than mechanical. These devices are inserted upstream of a sensitive load, and inject variable additive voltage to compensate for drops from nominal voltage [10]. An alternative DVR design allows for series compensation at existing distribution transformers, which eliminates the need for purchasing and installing insertion transformers [47]. SVC/DVR is an attractive means of mitigation to large customers with many sensitive loads, but are costly and cannot protect against interruptions or against sags generated within a plant [10].

Other mitigation techniques use magnetic characteristics to mitigate power quality disturbances. Superconducting magnetic energy storage devices (SCMES) use cryogenically cooled superconducting magnets to store energy, and deliver it in a manner similar to a UPS. They take less space than their UPS equivalents, but are expensive because of their cooling requirements [12]. Magnetic synthesizers, which are typically used for large loads, convert electric energy into magnetic energy through nonlinear chokes, then synthesize an output waveform using stored energy in saturation transformers and capacitors [46]. Advanced static var compensators, which are used for reactive power management, may also be used for voltage sag mitigation. They employ existing equipment that is used for other purposes, but require complicated controls and are adversely affected by phase angle jumps in a voltage sag [48]. Constant voltage transformers (CVT), also called ferroresonant transformers, use magnetic saturation properties to lessen the effects of voltage disturbances. They are tuned to specific load requirements, and are best suited for steady loads [10]. They can also help mitigate waveform disturbances such as harmonics and notching [49]. In practice, CVTs have been shown to be effective in protecting control equipment and PLCs in a coordinated process [50].

On a device level, solutions may be implemented to reduce the occurrence of voltage sags or improve device ridethrough. High motor starting currents can be limited by ‘soft start’ devices, or by programming a motor drive to slowly accelerate a load. Ridethrough of induction motor drives may be improved with advanced pulse width modulation techniques in the presence of a voltage sag [51]. Addition of extra capacitance in power supplies can improve ridethrough performance at low costs, but requires internal modifications to existing equipment, and may cause premature rectifier diode failure [52]. Process contactors in a may have their ridethrough improved by making modifications to their magnetic circuits [26].

Software Mitigation Solutions

Software based solutions for avoiding a detrimental voltage sag process response are also possible, and form the central issue of this dissertation. Upon detection of a voltage sag, a process may switch to an alternate control algorithm, and then return to normal operating conditions afterward [36]. This solution requires detailed knowledge of process mechanics and controls. Additionally, a fast acting voltage sag detector must be integrated into the process controls. Recent developments in sag detection technology have shown that detectors may be economically manufactured and implemented in a process to serve as a trigger for alternate control [53]. Newer sag detector designs even utilize microprocessor controls for increased reaction times and tolerance to steady state waveform disturbances [54]. Voltage sag detection using *dq* input voltage analysis has also been shown to have fast response times, but has a poor tolerance to waveform disturbances such as harmonics [55].

Control in the presence of sensor failure is addressed in [56-58], but requires redundant sensors and a control algorithm designed to accommodate the redundancy. Here methods such as weighting functions or majority decision are used to determine an adopted value to feed back into a control algorithm. These involve identifying and responding to faulted sensor outputs, yet do not account for input parameters such as supply voltage characteristics. Examples of observer control for complex processes are shown in [59] and [60]. This diminishes the possibility of signal corruption by voltage sags by reducing an overall sensor count, but requires computationally expensive calculations and reliable input signals for implementation.

Research Summary

Motivation and Purpose

In recent years, manufacturers have increased their use of sensor/actuator bus systems in factory processes. Before the advent of this technology, automated systems typically consisted of a PLC or computer controller connected to multiple input and output signal lines, where each line is dedicated to carrying status or control information between the controller and a single device. As process complexity grows, the number of required input and output signal lines increases as well. System installation and maintenance costs are consequently increased due to the large number of connections and lines that exist. These characteristics gave rise to the introduction and widespread use of sensor/actuator communications bus systems, which use networking technology to connect multiple factory devices to a single communications cable.

Access to a process communications bus creates an opportunity for developing improved software-based voltage sag mitigation methods. The conventional approach to improving process ridethrough focuses on adding voltage sag compensation equipment to critical hardware, or issuing 'cure-all' commands to a process when a voltage sag or its effects are detected. The presence of a data bus allows for the creation of monitoring and control software that may easily interface with the communications bus and issue alternate control and override commands in the presence of a voltage sag. This software may also interface with the incoming power supply for inclusion of voltage sag detection and measurement routines. In this environment, a customized approach to voltage sag mitigation may be used, where measured voltage sag characteristics influence the method of mitigation response. Connections with the data bus also help facilitate recording a voltage sag response history for use in determining an expected process response.

Voltage Sag Ridethrough Software

In this investigation, coordinated voltage sag response software is designed and implemented in a test bed process. The software design calls for avoiding the addition of conventional ridethrough hardware and minimizing modifications to existing control algorithms. It resides on a standalone PC that interfaces with both the process communications bus and the incoming power supply. The software performs the following functions:

1. Continuously monitors critical process signals and three phase line voltages.
2. Detects voltage sag inception and extinction using a fast detection algorithm.
3. Performs sag measurements (magnitude, duration) while the sag is in progress.
4. Records unmitigated process responses for future reference.
5. Uses voltage sag measurements to determine an expected process response by either accessing recorded process data or calculating an expected response using mathematical models of process behavior.
6. Determines an appropriate mitigation response based on both the expected response and preselected user input.
7. Issues commands to the process controller (PLC) that override the existing control algorithm and mitigate the process voltage sag response.
8. Returns the process to normal operating conditions after the voltage sag has ended.

For the purposes of experimental verification, the software is applied to a dedicated test bed process. This process is an integrated textile tension control system. Implementation of the software in an experimental environment requires a high level of customization, and therefore calls for analysis and simulation of the unmitigated and mitigated process responses. Understanding individual device behavior is also critical when implementing a process-wide mitigation strategy. This requires the development of subsystem device behavior models for use in a simulation environment and in the software mitigation routines themselves.

The details of this investigation are divided into three main sections. First, the experimental textile tension controller design and modeling is presented. In this forum, mathematical models and simulation are discussed. The unmitigated voltage sag process response is matched with the theoretical model obtained from simulation. This is followed by an explanation of the design and theoretical operation of the software routines used to mitigate the tension controller's voltage sag responses. Lastly, an experimental evaluation of the effectiveness of the software algorithms is presented as they are applied to the process.

CHAPTER 2
OPERATION AND RESPONSE OF EXPERIMENTAL PROCESS

This chapter describes the design, construction, and response of a model textile tension control process that is used for applications of software based ridethrough algorithms. A thorough mathematical analysis is included, followed by a discussion of process simulation. An investigation into the unmitigated voltage sag response of both individual components and the entire process ensues.

Topology and General Overview

A textile handling workstation was constructed for the purposes of determining process responses to power quality events and experimentating with mitigation strategies. The system is a multiple input-multiple output (MIMO) textile winding and tension controller, with a design philosophy of being self-contained and dedicated to research experiments. Textile was chosen as the process medium because of the prevalence of textile manufacturing plants in North and South Carolina and the associated possibility of nearby industry implementation of methods developed in this research. Because it does not function as a critical part of an existing manufacturing facility, the textile tension controller may be altered and experimented with in greater detail than a production-level process.

The mechanical layout of the textile tension controller is shown in Figure 2.1. Two AC motors actuate the payoff and takeup reels where the textile is accumulated. A stationary load cell measures web tension, tachometers measure reel speeds, and discrete proximity sensors determine end of cycle package accumulation on the takeup and payoff reels. Secondary analog instruments are included to measure process values when a primary analog instrument is affected by a voltage sag. The secondary load cell tension sensor and spool tachometers are not connected to the control system and serve solely as a source of backup measurement.

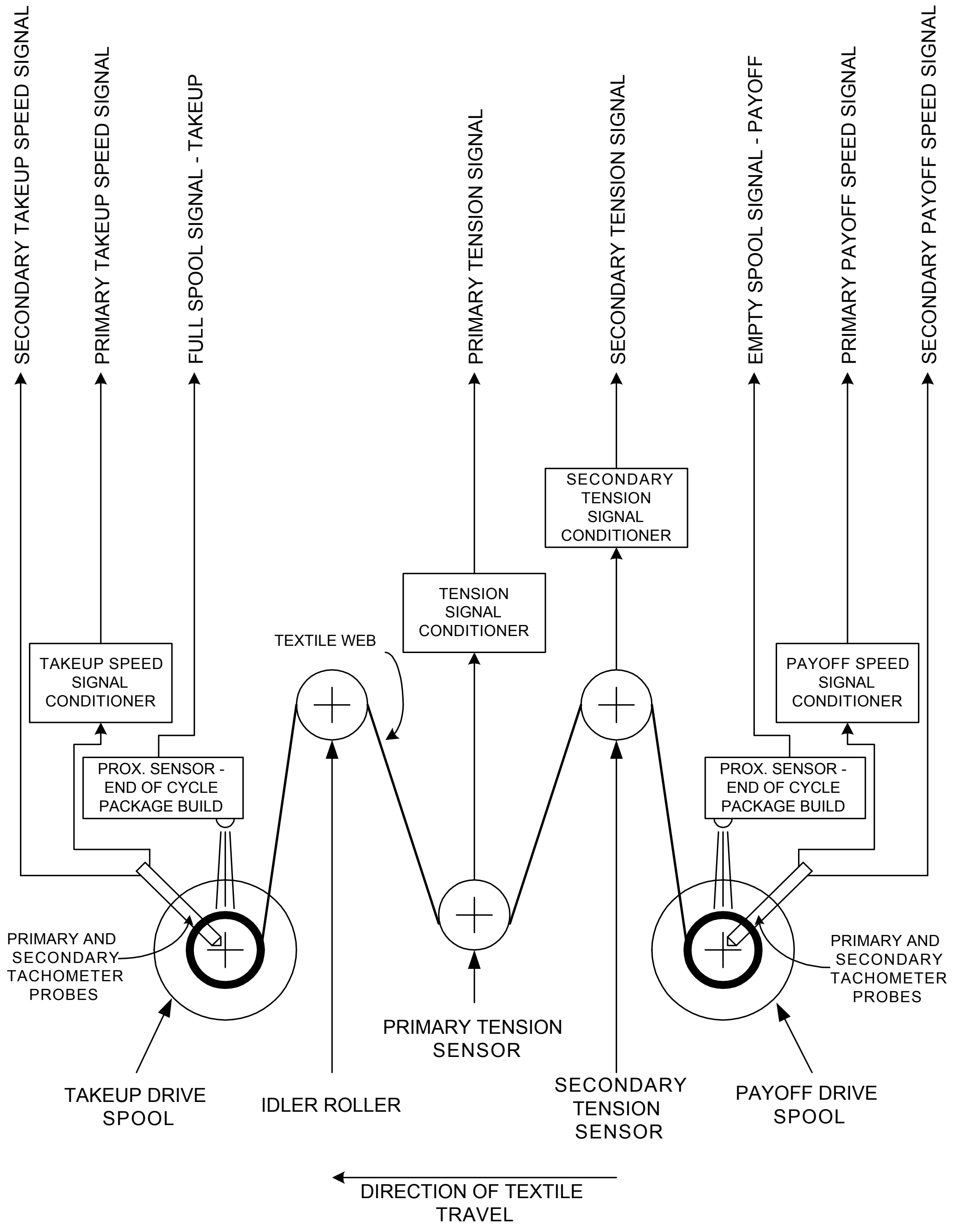

SECONDARY TAKEUP SPEED SIGNAL
PRIMARY TAKEUP SPEED SIGNAL
FULL SPOOL SIGNAL - TAKEUP
PRIMARY TENSION SIGNAL
SECONDARY TENSION SIGNAL
EMPTY SPOOL SIGNAL - PAYOFF
PRIMARY PAYOFF SPEED SIGNAL
SECONDARY PAYOFF SPEED SIGNAL
SECONDARY TENSION SIGNAL CONDITIONER
TENSION SIGNAL CONDITIONER
TEXTILE WEB
TAKEUP SPEED SIGNAL CONDITIONER
PAYOFF SPEED SIGNAL CONDITIONER
PROX. SENSOR - END OF CYCLE PACKAGE BUILD
PROX. SENSOR - END OF CYCLE PACKAGE BUILD
PRIMARY AND SECONDARY TACHOMETER PROBES
PRIMARY AND SECONDARY TACHOMETER PROBES
PRIMARY TENSION SENSOR
TAKEUP DRIVE SPOOL
IDLER ROLLER
SECONDARY TENSION SENSOR
PAYOFF DRIVE SPOOL
DIRECTION OF TEXTILE TRAVEL

Figure 2.1 Experimental textile tension control system mechanical layout and instrumentation scheme.

Figure 2.2 details the electrical interconnections of the textile tension controller. The 0-10VDC primary instrument feedback signals are connected to an I/O block which samples the analog signals and converts them to signed word data values. The data is transferred along a Siemens Profibus DP network into a Siemens PLC. The PLC receives user defined start/stop and setpoint commands from a standalone PC running a LabView human-machine interface (HMI) program. The PLC executes the main control algorithm and feeds the output commands back into the Profibus network, where they are received by two Siemens AC motor drives. The drives each control an AC induction motor, which is directly connected to the drive reels. A Siemens AS-I bus transfers end of cycle signals generated by the spool accumulation sensors and operates independent of the Profibus network. Programming and data bus monitoring interfaces are also connected to the system. For a complete system equipment list the reader is referred to Appendix A.

Textile tension control stand design is based on duplicating elements commonly found in industrial processes. The use of load cell tension measurement was chosen over a dancer arm system because of its mechanical simplicity. A comparison of dancer arm and load cell control strategies is presented in [61], and states that although differences exist in modeling and control of each system, there are no significant advantages of employing either method. A Profibus DP data bus network was chosen for device interconnection because of its prevalence in industry. This is supported by results of a 1998 study performed by Venture Development Corporation which indicated that Profibus DP was among the top three industrial communications buses in popularity and usage [62].

Control strategies and mechanical modeling methods for web processes vary based on the web characteristics. Strategies for controlling web tension using observers are presented in [59] and [60], but are often unique to a process and involve tension state calculation beyond the abilities of a typical PLC. Ultimately, spring-damper dynamic models similar to those described in [63] and [64] were chosen as the modeling method for the textile mechanics because they require fewer constants and calculation to determine web tension in a PLC environment. Control of the textile tension control

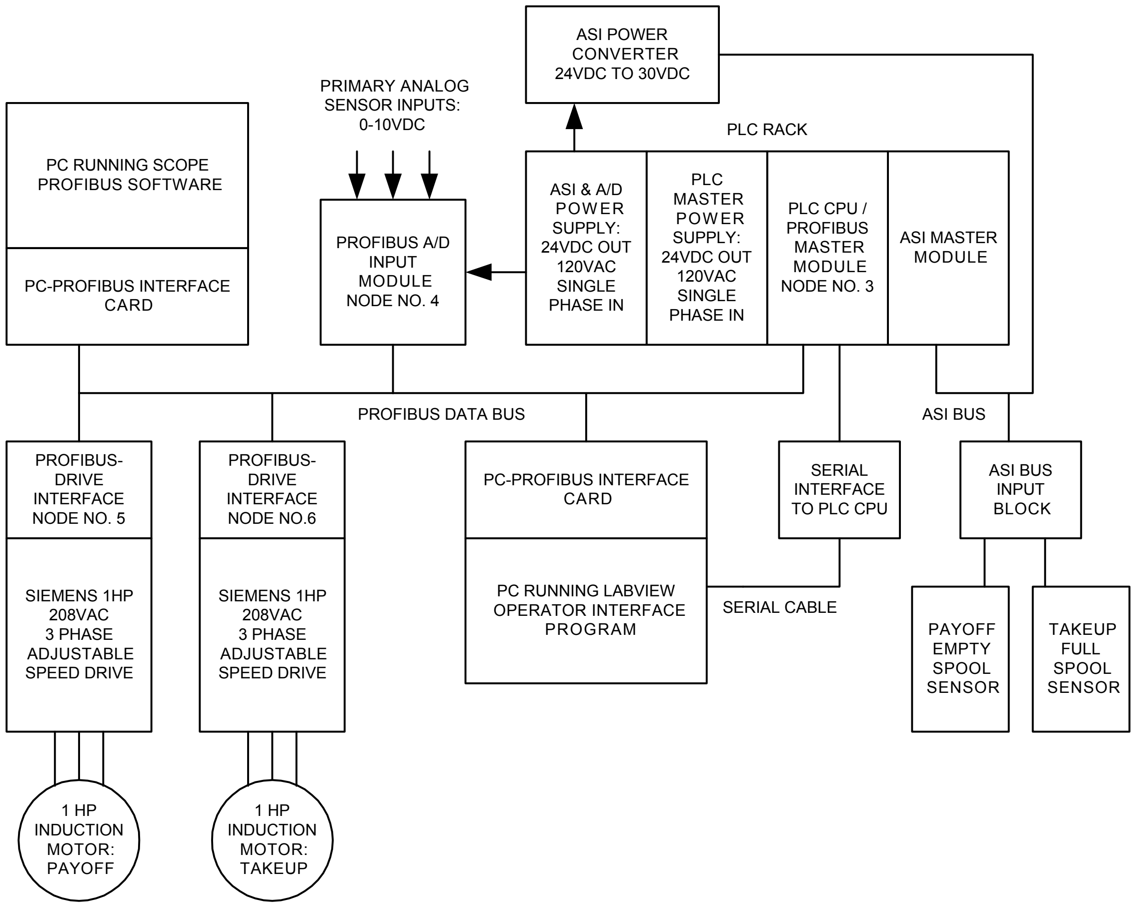

ASI POWER CONVERTER 24VDC TO 30VDC
PRIMARY ANALOG SENSOR INPUTS: 0-10VDC
PLC RACK
PC RUNNING SCOPE PROFIBUS SOFTWARE
PC-PROFIBUS INTERFACE CARD
PROFIBUS A/D INPUT MODULE NODE NO. 4
ASI & A/D POWER SUPPLY: 24VDC OUT 120VAC SINGLE PHASE IN
PLC MASTER POWER SUPPLY: 24VDC OUT 120VAC SINGLE PHASE IN
PLC CPU / PROFIBUS MASTER MODULE NODE NO. 3
ASI MASTER MODULE
PROFIBUS DATA BUS
ASI BUS
PROFIBUS-DRIVE INTERFACE NODE NO. 5
PROFIBUS-DRIVE INTERFACE NODE NO.6
PC-PROFIBUS INTERFACE CARD
SERIAL INTERFACE TO PLC CPU
ASI BUS INPUT BLOCK
SIEMENS 1HP 208VAC 3 PHASE ADJUSTABLE SPEED DRIVE
SIEMENS 1HP 208VAC 3 PHASE ADJUSTABLE SPEED DRIVE
PC RUNNING LABVIEW OPERATOR INTERFACE PROGRAM
SERIAL CABLE
PAYOFF EMPTY SPOOL SENSOR
TAKEUP FULL SPOOL SENSOR
1 HP INDUCTION MOTOR: PAYOFF
1 HP INDUCTION MOTOR: TAKEUP

Figure 2.2 Equipment configuration of experimental textile tension control system.

system is achieved by using two proportional-integral (PI) controllers with additive state gain terms to compensate for variations in linespeed and tension.

Mathematical Subsystem Models

The tension controller signal flow diagram is shown in Figure 2.3. The mechanical system is represented by a linear time-invariant state space system, with inputs of motor torque, and outputs of linespeed and web tension. The state space plant model derivation is provided in Appendix B.

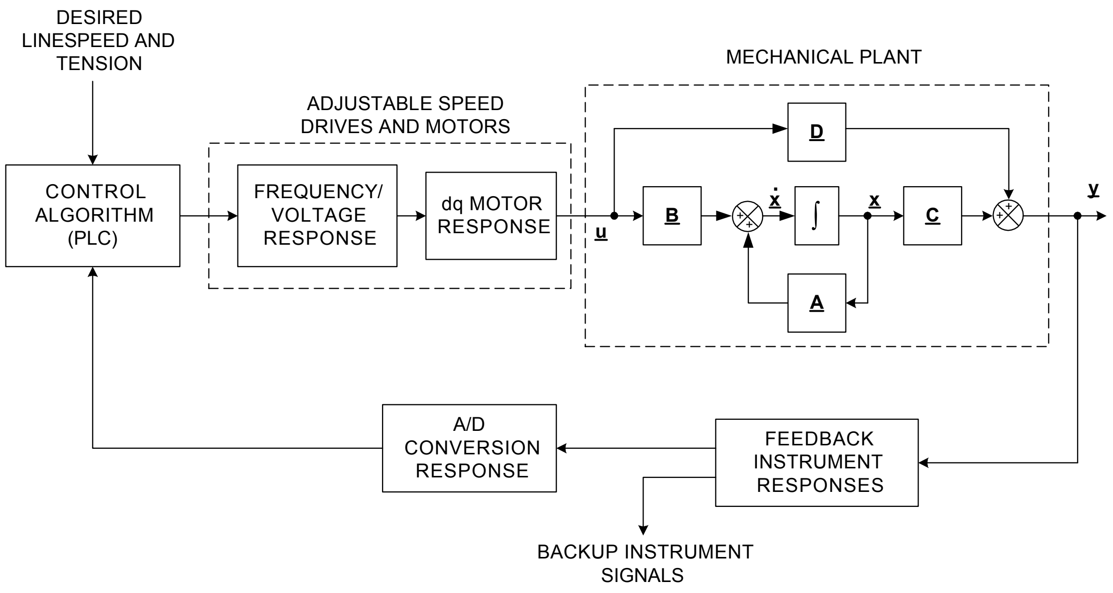

DESIRED LINESPEED AND TENSION
CONTROL ALGORITHM (PLC)
ADJUSTABLE SPEED DRIVES AND MOTORS
FREQUENCY/ VOLTAGE RESPONSE
dq MOTOR RESPONSE
MECHANICAL PLANT
u
D
B
ẋ
∫
x
C
A
y
A/D CONVERSION RESPONSE
FEEDBACK INSTRUMENT RESPONSES
BACKUP INSTRUMENT SIGNALS

Figure 2.3 Tension control system block diagram.

Feedback instruments under normal operating conditions are modeled as linear first order systems, where the measured mechanical value serves as an input, and an analog 0-10VDC signal the output. The A/D converter block is also modeled in this way, except the output is a signed 16-bit word, and the input an instrument signal voltage. In its generic form, the first order relationship is described by

$$\tau \frac{d}{dt} y(t) + y(t) = Kx(t) + B, \tag{2.1}$$

where $x(t)$ is the system input, $y(t)$ is the system output, K is a gain constant, B a constant output offset, and τ a time constant. To determine a first order subsystem output, we solve for the system output $y(t)$ in terms of a sum of integrals. Rearranging Equation 2.1 gives us

$$\frac{d}{dt} y(t) = \frac{K}{\tau} x(t) - \frac{1}{\tau}(y(t) - B), \tag{2.2}$$

and integrating both sides yields

$$y(t) = \frac{K}{\tau} \int_0^t x(t)\ dt - \frac{1}{\tau} \int_0^t (y(t) - B)\ dt. \tag{2.3}$$

Equation 2.3 may then be used to determine $y(t)$, provided that the initial conditions of the integrals are known.

The variable speed AC drives are modeled as systems that deliver a three phase sinusoidal voltage to the induction motors. The issued frequency, f_e, varies as a first order system with a signed word reference input and an electrical frequency output. The AC drives operate using a constant V/Hz control scheme, which decreases the output voltage magnitude with decreases in output frequency. At very low frequencies, the voltage is increased slightly from the constant V/Hz line, and at zero Hertz output frequency, the voltage is held above zero Volts at a constant level. This entire scheme creates a frequency-RMS voltage relationship which may be described by

$$V_{out} = V_{ebase} e^{m_e f_e}, \ (f_e < 6\ Hz) \tag{2.4}$$

$$V_{out} = m_l f_e + V_{offset}, \ (f_e \geq 6\ Hz) \tag{2.5}$$

where f_e is the motor output frequency, m_e is the exponential multiplier for low frequency operation, V_{ebase} is the zero Hertz base RMS voltage, m_l is the linear segment V/Hz slope, V_{offset} is the linear segment V/Hz offset, and V_{out} the RMS line-line motor output voltage.

The motors are modeled using direct-quadrature (dq) motor theory [65], where the motor inputs are sinusoidal three phase voltages, and the output is the developed motor torque. To determine the developed motor torque, we calculate the dq voltages from the drive output frequency and RMS line-line voltage magnitudes. The first step is to determine the instantaneous three phase stator voltages using

$$v_{as} = \sqrt{2} V_{out} \cos(\theta), \tag{2.6}$$

$$v_{bs} = \sqrt{2} V_{out} \cos(\theta - \frac{2\pi}{3}), \tag{2.7}$$

and

$$v_{cs} = \sqrt{2} V_{out} \cos(\theta + \frac{2\pi}{3}), \tag{2.8}$$

where $\theta = \omega_e t$. These voltages are then used to calculate the stationary dq voltages with

$$v_{qs}^s = \frac{2}{3} v_{as} - \frac{1}{3} v_{bs} - \frac{1}{3} v_{cs}, \tag{2.9}$$

and

$$v_{ds}^s = -\frac{1}{\sqrt{3}} v_{bs} + \frac{1}{\sqrt{3}} v_{cs}. \tag{2.10}$$

The stationary voltages are then transformed into the synchronously rotating frame voltages by

$$v_{qs} = v_{qs}^s \cos(\theta) - v_{ds}^s \sin(\theta), \tag{2.11}$$

and

$$v_{ds} = v_{qs}^s \sin(\theta) + v_{ds}^s \cos(\theta). \tag{2.12}$$

Additionally, for a shorted squirrel-cage induction motor the *dq* rotor voltages are

$$v_{qr} = v_{dr} = 0. \tag{2.13}$$

The motor flux relationships in the *dq* model are

$$v_{qs} = R_s i_{qs} + \frac{d}{dt}\Psi_{qs} + \omega_e \Psi_{ds}, \tag{2.14}$$

$$v_{ds} = R_s i_{ds} + \frac{d}{dt}\Psi_{ds} - \omega_e \Psi_{qs}, \tag{2.15}$$

$$v_{qr} = R_r i_{qr} + \frac{d}{dt}\Psi_{qr} + (\omega_e - \omega_r)\Psi_{dr}, \tag{2.16}$$

and

$$v_{dr} = R_r i_{dr} + \frac{d}{dt}\Psi_{dr} - (\omega_e - \omega_r)\Psi_{qr}. \tag{2.17}$$

Equations 2.14-2.17 may be rearranged and integrated to solve for the *dq* fluxes, which gives

$$\Psi_{qs} = \int_0^t v_{qs}\ dt - \int_0^t R_s i_{qs}\ dt - \int_0^t \omega_e \Psi_{ds}\ dt, \tag{2.18}$$

$$\Psi_{ds} = \int_0^t v_{ds}\ dt - \int_0^t R_s i_{ds}\ dt - \int_0^t \omega_e \Psi_{qs}\ dt, \tag{2.19}$$

$$\Psi_{qr} = \int_0^t v_{qr}\ dt - \int_0^t R_r i_{qr}\ dt - \int_0^t (\omega_e - \omega_r)\Psi_{dr}\ dt, \tag{2.20}$$

and

$$\Psi_{dr} = \int_0^t v_{dr}\ dt - \int_0^t R_r i_{dr}\ dt - \int_0^t (\omega_e - \omega_r)\Psi_{qr}\ dt, \tag{2.21}$$

which again may be easily calculated, provided that the initial conditions are known. Having determined the flux components, the flux-current transform

$$\begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{qr} \\ i_{dr} \end{bmatrix} = \begin{bmatrix} (L_{ls}+L_m) & 0 & L_m & 0 \\ 0 & (L_{ls}+L_m) & 0 & L_m \\ L_m & 0 & (L_{ls}+L_m) & 0 \\ 0 & L_m & 0 & (L_{ls}+L_m) \end{bmatrix}^{-1} \begin{bmatrix} \Psi_{qs} \\ \Psi_{ds} \\ \Psi_{qr} \\ \Psi_{dr} \end{bmatrix} \tag{2.22}$$

is used to calculate the *dq* currents in the machine. From these currents, the developed torque is calculated by

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right) L_m(i_{qs}i_{dr} - i_{ds}i_{qr}), \tag{2.23}$$

where P is the number of poles in the machine. For a detailed derivation of *dq* motor theory, the reader is referred to [65].

Figure 2.4 illustrates the control algorithm that the PLC executes. The linespeed controller is essentially a takeup reel speed controller. The desired linespeed is divided by the takeup reel radius to obtain the desired takeup reel speed. The actual takeup reel speed is then subtracted from the desired value to create an error signal (SP-PV operation). The error signal is then fed into a proportional-integral (PI) controller. The load torque caused by tension is calculated using the takeup reel radius and the tension feedback signal. This is multiplied by a gain constant and added to the PI controller output as a tension compensation term. Finally, a gain intrinsic to PI control in the Siemens PLC is applied, and the reference motor frequency signal is sent to the takeup motor drive.

The tension control loop behaves similarly. The desired tension is multiplied by the payoff reel radius to calculate a desired payoff tension torque. This is subtracted from the actual payoff tension torque to create an error signal (PV-SP operation), which is delivered to another PI controller. A state gain term to compensate for linespeed variations is added to the output of the PI controller, which is then multiplied by the intrinsic PI control gain and sent to the payoff drive.

Extensive testing of the textile tension controller was essential when determining the constants used in the plant and instrument mathematical models. Appendix C describes test procedures and provides tables of the determined values.

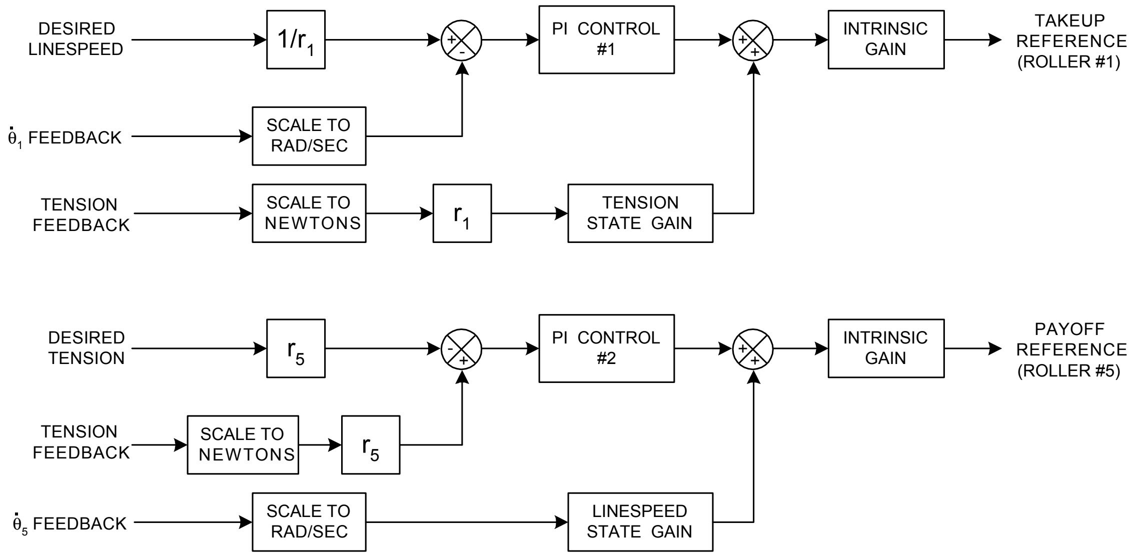


Figure 2.4 Controller for textile tension control system.

Simulation of Test Stand Behavior

The environment chosen for tension controller simulation was a line code implementation in Matlab. The use of line code was preferred over the graphical Matlab Simulink environment because the system's complexity makes a Simulink implementation unnecessarily complex and unwieldy. Furthermore, complete control over a model's behavior is more attainable in a line code environment. In a Simulink implementation, the ability to manipulate many built-in function blocks is limited.

General Overview

A flowchart for the code simulation is shown in Figure 2.5. A large main program with no use of subfunctions was favored over many small subfunctions under a main control program. This approach requires no variables to be passed between functions and maintains all variable assignments in memory during program execution. The duration of simulation was chosen to be a maximum of three seconds, which provides adequate time to analyze prefault, faulted, and postfault conditions. For limited ‘staircasing’ of system signals, a resolution of 10,000 points per second was chosen, or one evaluation point every $100\mu s$. Altogether, the simulation may calculate 30,000 data points for each signal. Arrays of every intermediate (non-output) variable are stored during simulation to help facilitate signal analysis throughout the system.

Variable Initialization

Setting the initial conditions of all variables is essential for establishing the starting points of system signals. This requires both forward and backward calculation of variables based upon the initial conditions of linespeed and tension.

The mechanical system initially requires all of the states in $\mathbf{x}$ and $\dot{\mathbf{x}}$ to be defined. These consist of values for all θ_i, $\dot{\theta}_i$, and $\ddot{\theta}_i$. Given that the linespeed is constant at $t = 0$, we calculate the values of $\dot{\theta}_i$ and $\ddot{\theta}_i$ as

$$\dot{\theta}_i = \frac{v_{t=0}}{r_i}, \tag{2.24}$$

and

$$\ddot{\theta}_i = 0. \tag{2.25}$$

Calculations for the initial values of θ_i requires arrangement of Equations B.25, B.8, B.9, and B.10 in matrix form, where the values of θ_i are unknown quantities. To square the constants matrix, the value of θ_3 is set to 10π (a value that would not lead to a negative initial value for θ_1). This arrangement creates the matrix equation

$$\begin{bmatrix} 2f_{p,t=0} \\ B_2\dot{\theta}_2 \\ B_3\dot{\theta}_3 \\ B_4\dot{\theta}_4 \\ 10\pi \end{bmatrix} = \begin{bmatrix} 0 & r_2K_{23} & -r_3K_{23} + r_3K_{34} \\ r_1r_2K_{12} & -r_2^2K_{12} - r_2^2K_{23} & r_2r_3K_{23} \\ 0 & r_2r_3K_{23} & -r_3^2K_{23} - r_3^2K_{34} \\ 0 & 0 & r_3r_4K_{34} \\ 0 & 0 & 1 \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} -r_4K_{34} & 0 \\ 0 & 0 \\ r_3r_4K_{34} & 0 \\ -r_4^2K_{34} - r_4^2K_{45} & r_4r_5K_{45} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix}, \tag{2.26}$$

where the solution for matrix $\boldsymbol{\theta}$ is found by multiplying the inverse of the constants matrix by the left hand side of Equation 2.26. Once values for $\mathbf{x}$ and $\dot{\mathbf{x}}$ were determined, the input torques, $\mathbf{u}$ are determined using Equation B.1.

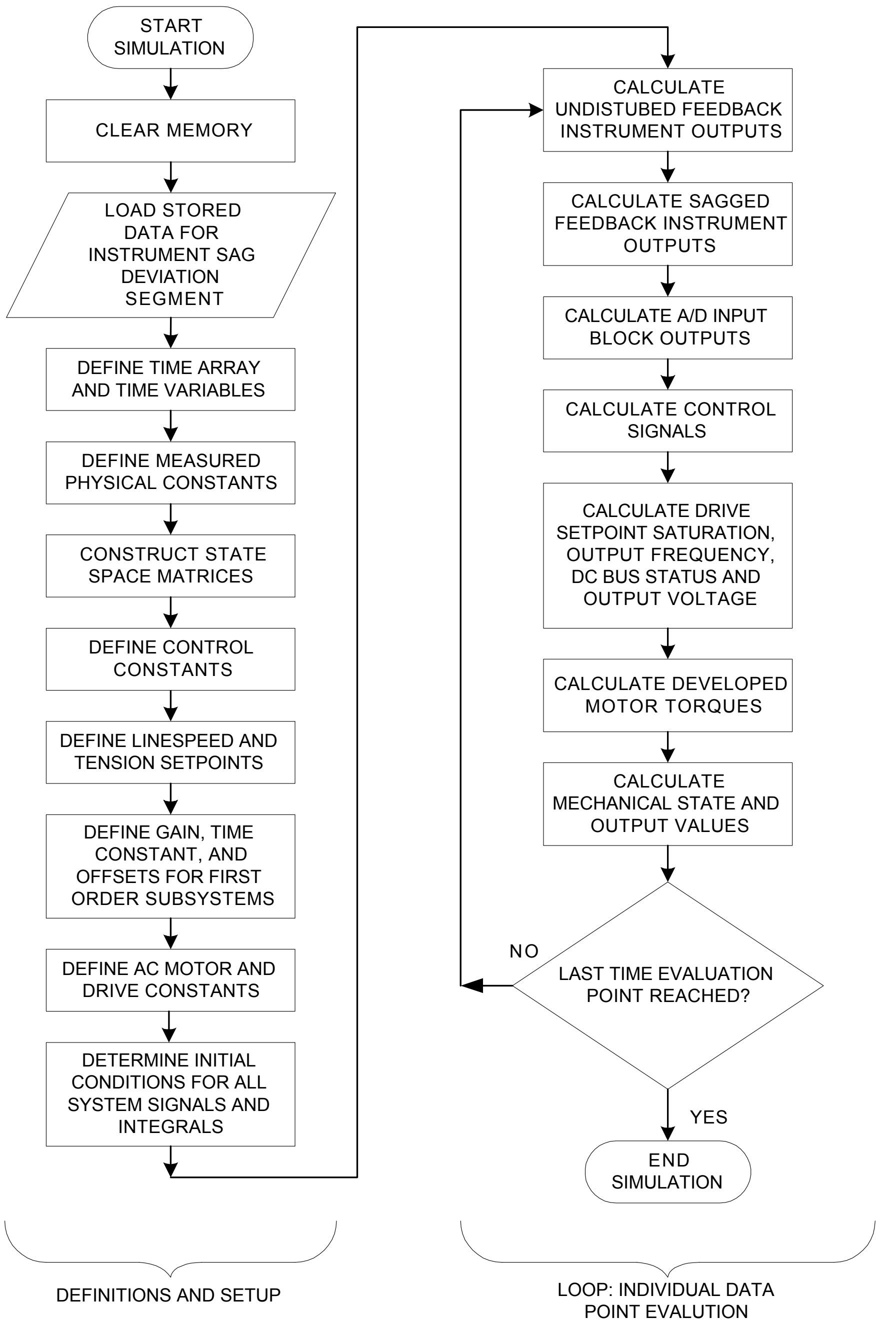

START SIMULATION
CLEAR MEMORY
LOAD STORED DATA FOR INSTRUMENT SAG DEVIATION SEGMENT
DEFINE TIME ARRAY AND TIME VARIABLES
DEFINE MEASURED PHYSICAL CONSTANTS
CONSTRUCT STATE SPACE MATRICES
DEFINE CONTROL CONSTANTS
DEFINE LINESPEED AND TENSION SETPOINTS
DEFINE GAIN, TIME CONSTANT, AND OFFSETS FOR FIRST ORDER SUBSYSTEMS
DEFINE AC MOTOR AND DRIVE CONSTANTS
DETERMINE INITIAL CONDITIONS FOR ALL SYSTEM SIGNALS AND INTEGRALS
CALCULATE UNDISTUBED FEEDBACK INSTRUMENT OUTPUTS
CALCULATE SAGGED FEEDBACK INSTRUMENT OUTPUTS
CALCULATE A/D INPUT BLOCK OUTPUTS
CALCULATE CONTROL SIGNALS
CALCULATE DRIVE SETPOINT SATURATION, OUTPUT FREQUENCY, DC BUS STATUS AND OUTPUT VOLTAGE
CALCULATE DEVELOPED MOTOR TORQUES
CALCULATE MECHANICAL STATE AND OUTPUT VALUES
LAST TIME EVALUATION POINT REACHED?
NO
YES
END SIMULATION
DEFINITIONS AND SETUP
LOOP: INDIVIDUAL DATA POINT EVALUTION

Figure 2.5 Flowchart of tension controller voltage sag simulation program.

The instrument output initial conditions were determined by multiplying the physical input by the instrument gain, then adding the instrument offset. This operation was also performed for the A/D inputs. Note that this is a steady state interpretation of Equation 2.1. The initial conditions of the integral sums used to calculate the outputs of these devices were then assigned a value of the output at $t = 0$.

Calculation of the initial *dq* voltage, flux, and current conditions for the motors could not be done in reverse using the output torques because several unique combinations of *dq* currents could lead to the same developed torque. Therefore, for each motor an iterative scan and forward calculation through multiple electrical frequency setpoints was performed, and the resulting output torque compared to the actual initial torque. The input electrical frequency that yielded an output torque equal to the actual initial output torque was then stored as the initial condition for the motor's desired electrical frequency. Calculation of initial line voltages were determined using the V/Hz drive scheme described by Equations 2.4 and 2.5. The initial *dq* voltages were calculated using Equations 2.6 through 2.13, and *dq* currents were calculated assuming steady state operation by

$$\begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{qr} \\ i_{dr} \end{bmatrix} = \begin{bmatrix} R_s & \omega_e L_s & 0 & \omega_e L_m \\ -\omega_e L_s & R_s & -\omega_e L_m & 0 \\ 0 & (\omega_e - \omega_r) L_m & R_r & (\omega_e - \omega_r) L_r \\ -(\omega_e - \omega_r) L_m & 0 & -(\omega_e - \omega_r) L_r & R_r \end{bmatrix}^{-1} \begin{bmatrix} v_{qs} \\ v_{ds} \\ v_{qr} \\ v_{dr} \end{bmatrix}. \tag{2.27}$$

Initial motor fluxes were then calculated using the flux-current transform in Equation 2.22, and the initial values of the corresponding integral sums were assigned the values of the fluxes at $t = 0$.

Some control signals also required calculation of their initial conditions. The outputs of the integral controllers were calculated assuming that the controller errors at $t = 0$ are zero, and the proportional contributions to controller output are zero as well. With these values set, the integral sums were back calculated using the initial electrical frequency outputs and the initial feedback signal values.

Numeric Evaluation of Integrals

The first order solution in Equation 2.3 is obtained numerically, where the value of the integrals at $t = t - dt$ is calculated and multiplied by dt, then added to the stored integral sum to yield the output value. Since the integrals are calculated using values from the previous step in time, any error in this calculation can be decreased by using smaller dt time step. This evaluation method was used for the feedback instruments, A/D converter, first order drive setpoint transitions, and the dq flux calculations.

Data Sampling

The data sampling that occurs in the PLC, data bus, and I/O modules is modeled by the inclusion of zero order holds in the simulation. This is achieved by referencing signals at previous times which are determined by

$$t_{ref} = T_s \left\lfloor \frac{t}{T_s} \right\rfloor \tag{2.28}$$

where T_s is the zero order hold sampling period, and t_{ref} is the previous time reference.

Process Response to Voltage Sags

The textile tension control stand's overall response to voltage sags is dependent on the individual hardware sag responses. Because of this, the analysis of process behavior must initially concentrate on the component devices.

Analog Instrumentation

Assessment of instrument voltage sag responses may take a pass/fail approach, where device responses are described with terms such as 'immune' or 'tolerant' to voltage sags [66]. More clearly specified are voltage sag tolerances described by CBEMA or ITI curves where magnitude and duration thresholds are defined in order to determine device compliance with a particular specification [46]. Meeting these standards is open to interpretation for analog process sensors because during some sags the device will continue to operate, but may not be considered as surviving the event due to an erroneous output signal.

In 2000, a study performed by EPRI Solutions (formerly EPRI PEAC) examined the voltage sag response characteristics of a wide range of process sensors. The findings from this study offer some insight into general sensor behavior which we aim to analyze. It was shown that sensor responses were not sensitive to the point-on-wave characteristic of voltage sags, which suggests that point-on-wave variations would be an unnecessary addition to an instrument sag response model. On the other hand, the sensor output level at the time a voltage sag is applied must be taken into consideration because the output level affects the device loading, which in turn affects the ridethrough time [67].

Figure 2.6 shows the response of the payoff roller tachometer, takeup roller tachometer, and tension sensor to a 0% sag (interruption) for a duration of 450ms. Our initial goal is to model the output response of each sensor so that they may be implemented in the Matlab process simulation routine. In the past, models of corrupted sensor outputs have focused on injecting a subtractive disturbance [68] or a variable gain [69] to describe the initial decays seen in Figure 2.6. The shortcomings of these models, however, are that erratic device discharge and gradual recovery are not accounted for. The improved instrument sag response model aims at including these factors, while viewing the instruments as 'black boxes', whose internal circuitry is of less importance than the overall input-output characteristics.

To develop a generic analog instrument sag response model, we begin by examining the interconnections that are most frequently used for these devices. Figure 2.7 shows common topologies for both AC and DC powered instruments. DC powered instruments are typically fed by a separate AC to DC converting power supply, which may also feed other DC loads. In this case, it is important to note that the presence of additional loads will affect the sag response of the instrument because of the energy demands these loads place on the supply. The physical input to the device also affects the response because it places varying demands on the instrument's stored energy during the sag. AC supplied instruments behave similarly, only without the presence of additional DC loads.

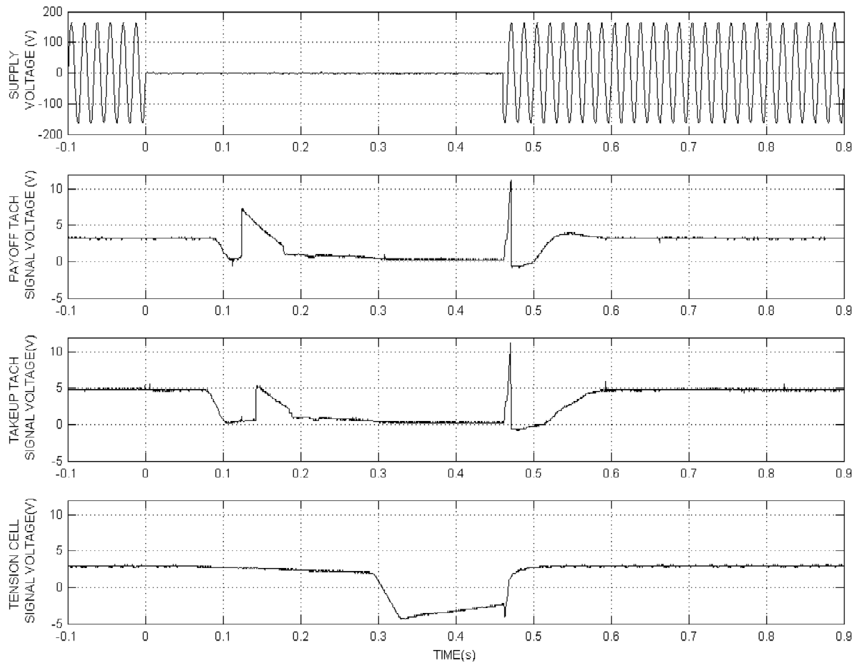


Figure 2.6 Textile tension control stand instrumentation voltage sag responses.

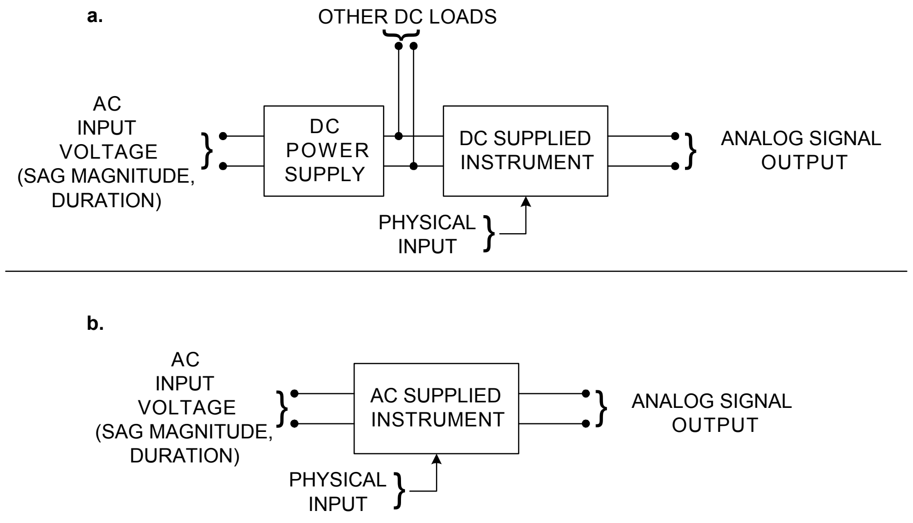


Figure 2.7 Instrument input-output topologies for a) DC sensor fed through separate power supply and b) AC supplied sensor.

Figure 2.8 illustrates the output response of several instruments when a 10% voltage sag is applied for a duration of $450ms$. Each response may be divided into a series of operating segments. The prefault segment occurs before any disturbances have been applied to the sensor. During this condition, the instrument behaves normally. Similarly, the postfault segment occurs after the supply disturbance and all of its effects have cleared, and is also treated as an undisturbed instrument response.

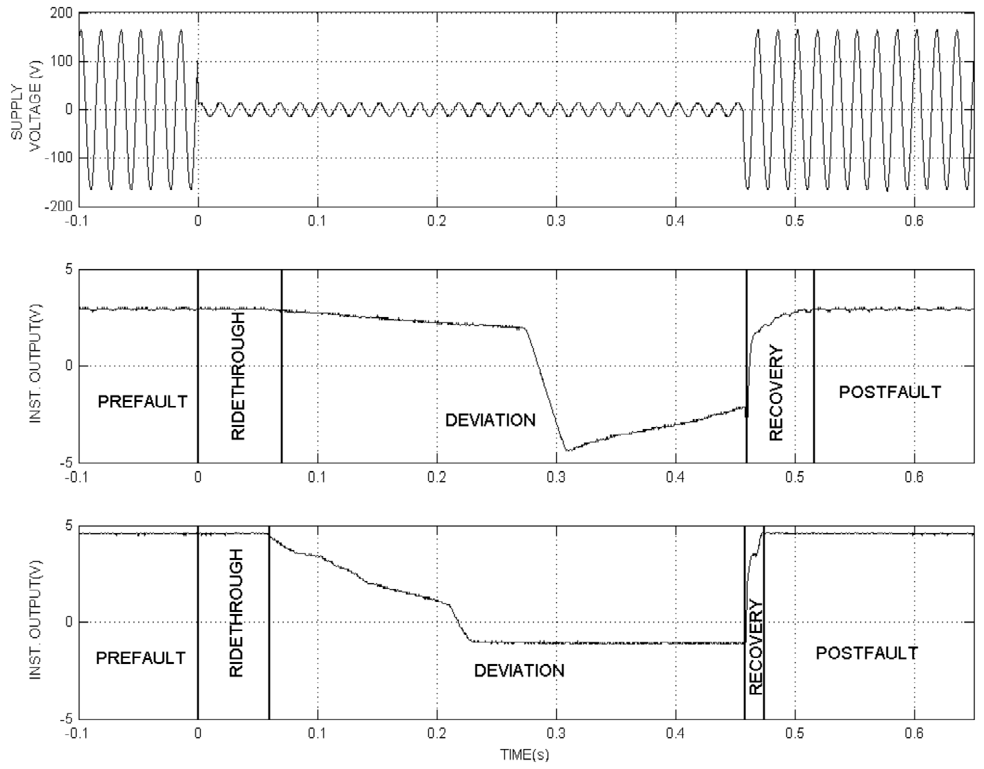


Figure 2.8 Instrument voltage sag responses divided into segments of operation.

Ridethrough Segment

The ridethrough segment of the voltage sag response occurs from the time of sag inception to the time at which the output signal begins to deteriorate, and is considered a general indication of the energy stored in a device's power supply. During this segment the instrument functions normally. The duration of the ridethrough segment is dependent on both sag magnitude and the physical input level, as illustrated in Figure 2.9. The magnitude of the voltage sag has an effect on the ridethrough times because different amounts of energy are drawn at various supply magnitudes, while the physical input level places varying demands on the instrument's stored energy.

When describing the ridethrough segment, sag duration is only a consideration if it is less than the ridethrough time given a certain magnitude and physical input level. If the sag duration is shorter, then the ridethrough time is equal to the sag duration. Under these circumstances, the instrument may be considered immune to a voltage sag, and the deviation and recovery segments are not present.

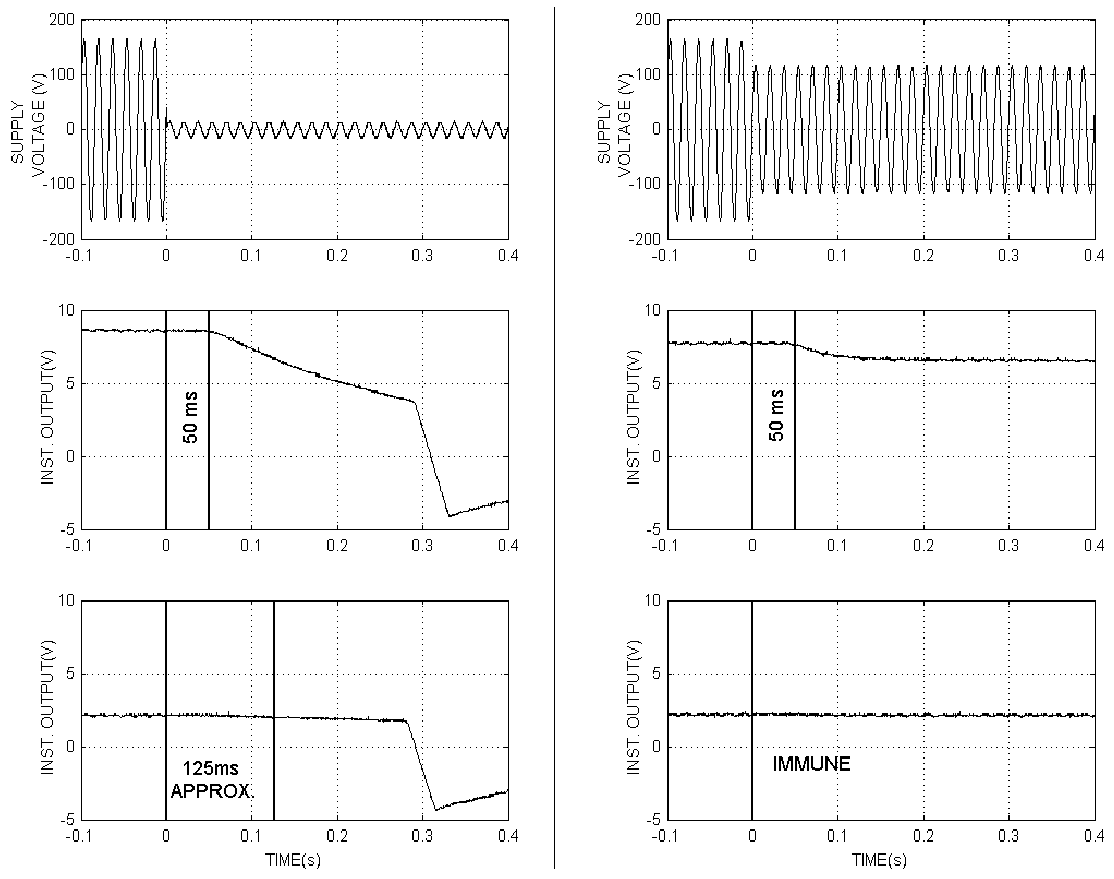


Figure 2.9 Effects of variations in sag magnitude and physical input level on ride-through segment of instrument voltage sag response.

Deviation Segment

The deviation segment of the instrument voltage sag response begins at the point where the output signal begins to deviate from its prefault and ridethrough values. It terminates at the time of sag extinction. Often this response takes on the form of a decaying exponential signal, but it can also contain spikes and transients when components in the instrument de-energize. Again, both the voltage sag magnitude and physical input level have an effect on the nature of this response. Figure 2.10 exemplifies this for several cases. The duration of the voltage sag only defines the time at which the segment ends, and does not affect the nature of the deviated waveform. Figure 2.11 demonstrates the effects of duration changes given constant magnitude and physical input conditions.

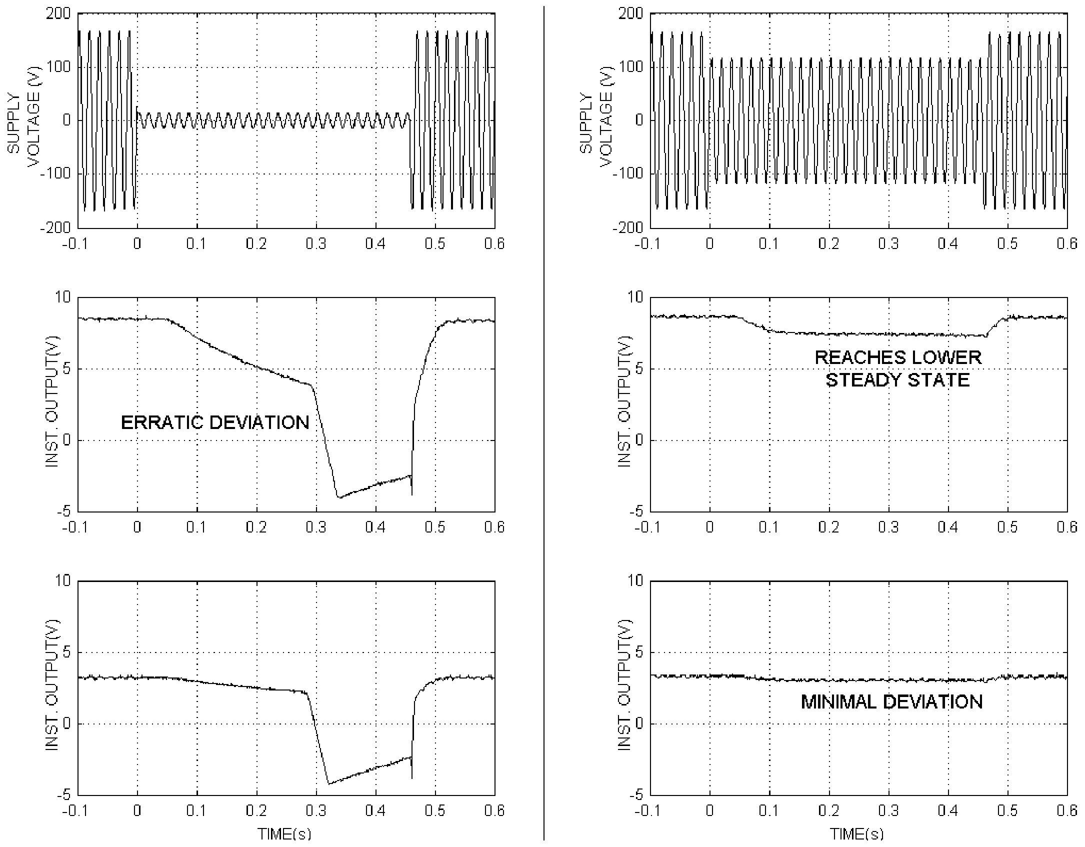


Figure 2.10 Effects of variations in sag magnitude and physical input level on deviation segment of instrument voltage sag response.

Recovery Segment

The recovery segment begins at the time of sag extinction. Typically this involves an exponential rise from the output at the time of sag extinction to the unfaulted instrument output level. Previous models have treated this as a simple step function, which implies that the output immediately returns to normal when the sag ends. Figure 2.12 illustrates the exponential nature of device recovery, and that factors of sag magnitude and physical input affect recovery times. Sag duration also has an effect on the recovery time, since the duration plays a part in determining initial conditions at the time of recovery.

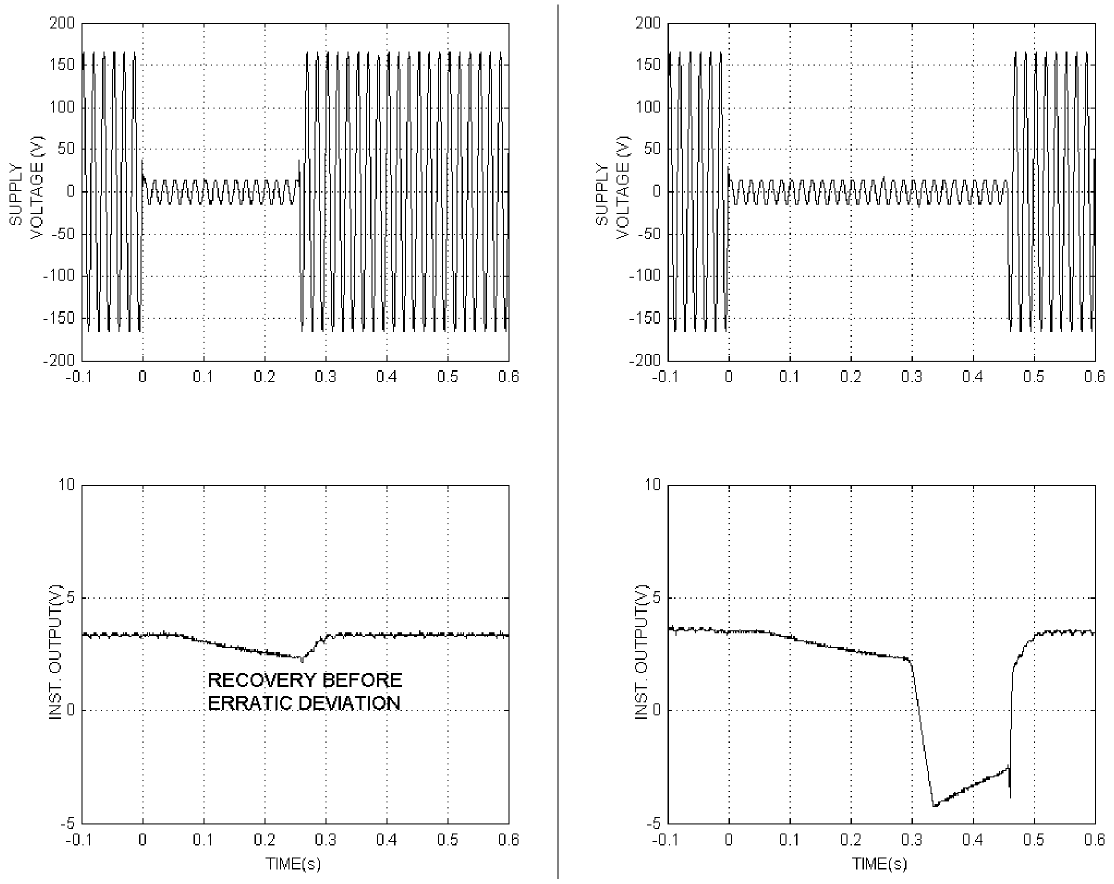


Figure 2.11 Effects of variations in sag duration on deviation segment of instrument voltage sag response.

An important property of device recovery is that the output signal gradually transitions from a corrupted signal to an uncorrupted one. When the device is fully recovered, the output is expected to be an accurate representation of the physical input. A mathematical description of this behavior must account for changing input levels during the recovery process, rather than use a constant to describe the final signal value. This may be modeled as

$$Output(t) = (Output_{t=t_{ext}})(e^{\frac{-(t-t_{rec})}{\tau_{rec}}}) + F(Input(t))(1 - e^{\frac{-(t-t_{rec})}{\tau_{rec}}}), \qquad (2.29)$$

where t_{rec} is the time of sag recovery, τ_{rec} the recovery rise time constant, F the transfer function for the undisturbed instrument, and $Input(t)$ the device physical

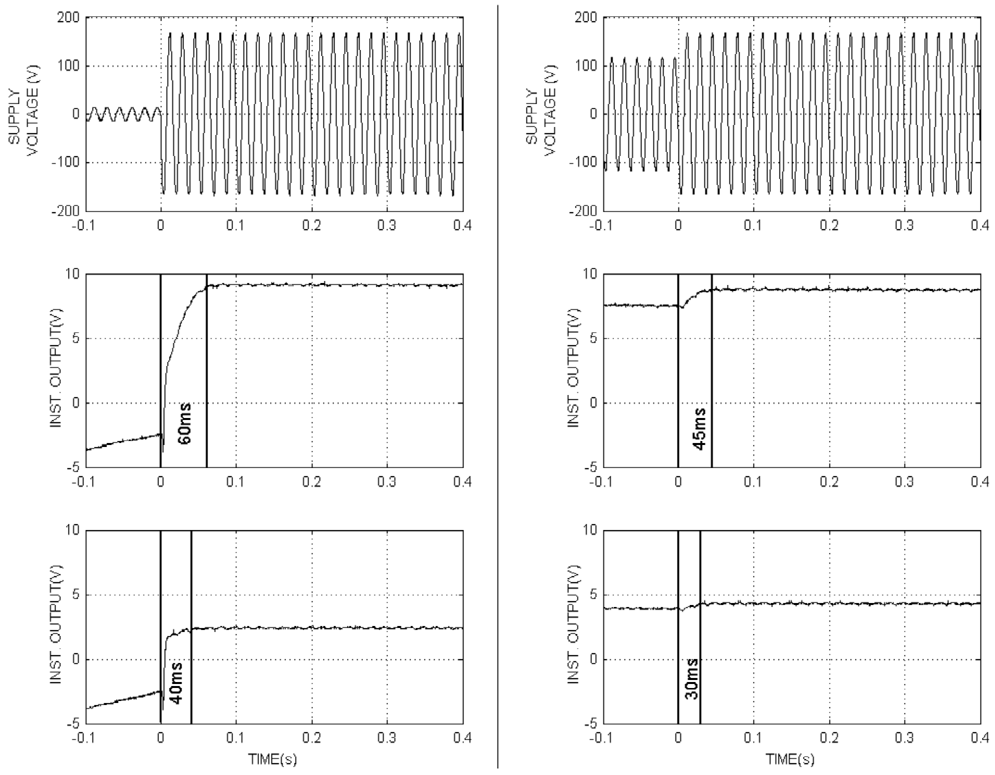


Figure 2.12 Effects of variations in sag magnitude and physical input level on recovery segment of instrument voltage sag response.

input as a function of time. This representation of the recovery response is a sum of the corrupted signal final value scaled to become less significant with time and the uncorrupted signal scaled to become more significant with time. Figure 2.13 shows a comparison of the theoretical output when applied to an actual recovery signal assuming a linear undisturbed instrument response and constant physical input.

Implementing Results in Simulation

The observed ridethrough times, maximum recovery time, and deviation segment trace data for a particular sag magnitude and device physical input combination may be applied in a simulation environment. To implement in the Matlab simulation of the textile tension control stand, an if-then structure is used to output each segment

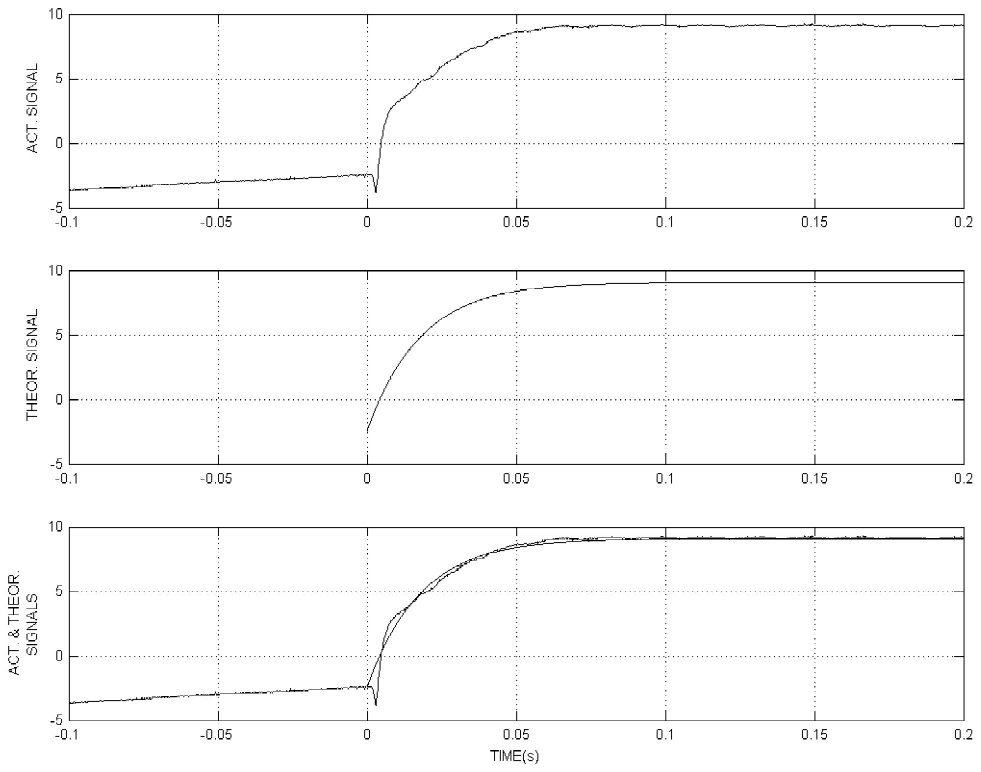


Figure 2.13 Application of mathematical recovery response model to instrument using constant physical input level.

response at a specific time. Figure 2.14 shows a flowchart for instrument sag response simulation and Table 2.1 describes the event schedule that controls the segment transitions. For prefault conditions, the normal instrument output is delivered to the remainder of the control system. The same is true for the ridethrough segment, up to the point where $t = t_{start} + t_{ridethru}$. This point marks the beginning of the deviation segment.

Because of the variety of deviated waveforms that may be produced by a particular instrument during the deviation segment, the program references the appropriate output to deliver from a sampled waveform obtained during sag testing. This avoids the impractical task of calculating output values based on a customized instrument response model, and meets the requirement of being able to simulate a value that cor-

Table 2.1 Event schedule for implementation of improved instrument voltage sag response model in simulation.

SEGMENT	TIME FRAME
Prefault	$t_{start} < t < t_{incep}$
Ridethrough	$t_{incep} < t < (t_{incep} + t_{ridethru})$
Deviation	$(t_{incep} + t_{ridethru}) < t < t_{rec}$
Recovery	$t_{rec} < t < (t_{ext} + 10\tau_{ridethru})$
Postfault	$(t_{rec} + 10\tau_{ridethru}) < t < t_{end}$

rectly represents an erratic output waveform. In this step, it is important to maintain consistency between sample times in simulation and in sag testing. Furthermore, it is preferable to use a high sampling frequency to avoid waveform 'staircasing' during the deviation segment simulation. The duration of the theoretical sag may be varied by switching from the sampled waveform in the deviation segment to the recovery segment before the entire sampled waveform is sent as an output.

When the sag has ended at t_{rec}, the program outputs an implementation of Equation 2.29. Here the value of τ_{rec}, the recovery time constant, is equal to one fifth of the maximum recovery time measured for the device. The equation is calculated and delivered as a device output until ten recovery time constants have elapsed following t_{ext}, at which point the normal operating conditions are again restored as the device transfer function. The process simulation Matlab code, including the instrument sag response models, are included in Appendix D.

Results for instrumentation voltage sag tests are shown in Appendix E. Each sensor was tested to determine ridethrough times at varying physical input levels. The physical levels used were taken over a range from the lowest allowable process physical input level to the highest, and did not use the output limits of 0-10VDC as the test limits. Sag magnitudes ranged from 0% to 80% in 20% increments and durations were adjusted to determine the maximum recovery time.

AC Motor Drives

Both of the AC motor drives operate using rectifier/inverter topology. The main effect that voltage sags have on the AC motor drives is to lower the DC bus magnitude

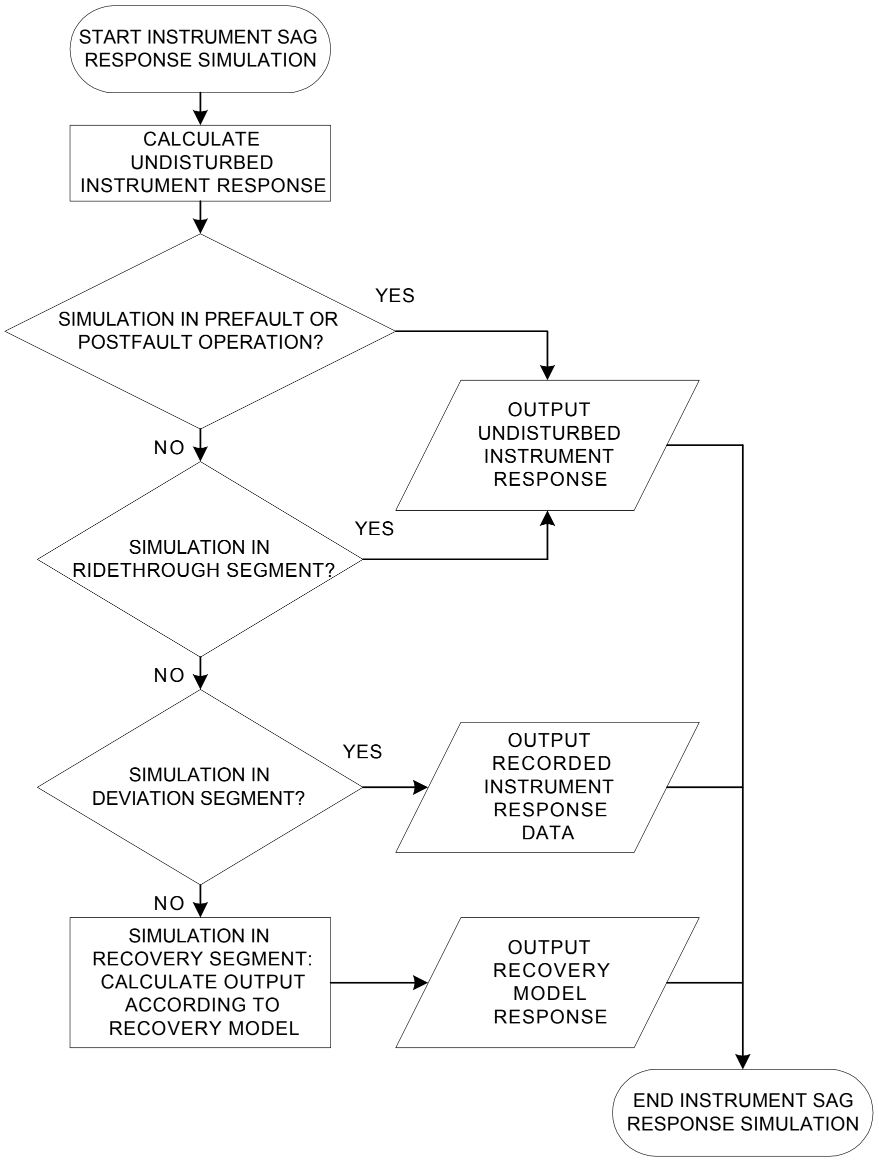

START INSTRUMENT SAG RESPONSE SIMULATION
CALCULATE UNDISTURBED INSTRUMENT RESPONSE
SIMULATION IN PREFAULT OR POSTFAULT OPERATION?
YES
NO
OUTPUT UNDISTURBED INSTRUMENT RESPONSE
SIMULATION IN RIDETHROUGH SEGMENT?
YES
NO
SIMULATION IN DEVIATION SEGMENT?
YES
NO
OUTPUT RECORDED INSTRUMENT RESPONSE DATA
SIMULATION IN RECOVERY SEGMENT: CALCULATE OUTPUT ACCORDING TO RECOVERY MODEL
OUTPUT RECOVERY MODEL RESPONSE
END INSTRUMENT SAG RESPONSE SIMULATION

Figure 2.14 Flowchart for simulation of instrument voltage sag response.

between the rectifier and controlled inverter below their nominal prefault levels. Both the magnitude and phasing combination of the incoming voltage sags have an effect on the DC bus level. Figure 2.15 shows the DC bus response for two different events. The left column shows a 60%, three phase voltage sag of 450ms duration and the corresponding DC bus response. For this sag, the DC bus level drops below the trip level of approximately 200VDC, where the drive halts its output to the motor. Afterward, the DC bus magnitude continues to decline, but at a lower rate because energy is no longer consumed by the running motor. After an output trip and sag recovery, the DC bus returns to its nominal magnitude, but experiences a brief period of ripple that matches the peaks of the incoming sinusoidal voltages.

The right column of Figure 2.15 shows the DC bus magnitude of the same drive when it is stricken by an 80%, three phase voltage sag of 450ms duration. In this case, the DC bus magnitude drops sharply after the sag inception, but remains above the 200VDC trip level. The motor does not trip offline, and the DC bus does not ripple after sag recovery.

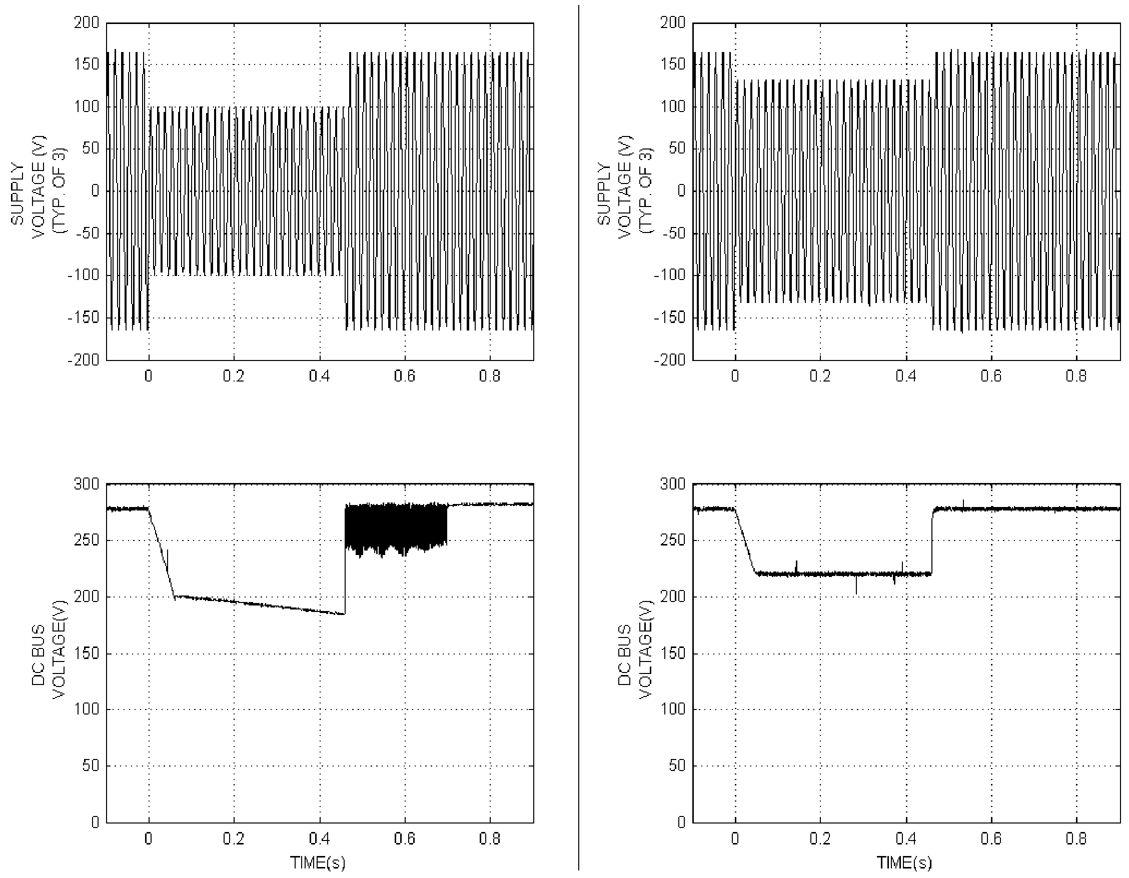


Figure 2.15 DC bus levels during three phase sags of magnitude 60% (left) and 80% (right).

For unbalanced sags, the response is similar. Maintenance of the DC bus above the trip level defines its ability to ride through the event, and bus decay occurs in the same fashion as in the balanced case. For sag magnitude and phasing combinations that cause an output trip, significant DC bus ripple can occur after the dropout. For combinations that do not cause a dropout, increased ripple can occur during the sag when the DC bus has reached its steady state depressed level.

Figure 2.15 illustrated the difference in the DC bus magnitude for both trip and ridethrough conditions, but assumed a sag duration long enough for the magnitude and phasing combination to be the most significant factors in determining the drive's ability to withstand the sag. Figure 2.16 shows the effects that shortened durations can have on the drive DC bus magnitude. The left column shows the response to varying durations for a sag magnitude/phase combination that causes an output trip, and shows the rapid decay of the DC bus under these conditions. During the sag, the drive briefly continues to run the motor, but quickly reaches the dropout voltage and halts the output. This suggests that while the duration of the incoming sag is a factor in determining drive tolerance, it is only critical during the first few cycles of the sag. The right column of Figure 2.16 shows the effects of varying sag duration for a magnitude/phasing combination that does not cause a drive output trip. Here these effects are even less significant, as sag duration only determines the length of time that the DC bus stays at a depressed level.

The rate of decent of the DC bus is also a variable characteristic, and is dependent on motor loading. Figure 2.17 demonstrates this effect. For the sag shown in the top waveform, the DC bus is shown for various loading conditions. The second waveform shows the steep decent in the DC bus during heavily loaded conditions, and the third waveform shows a more gradual decay when the motor is lightly loaded. The last waveform shows the no load rate of DC bus decay, which was measured when the motor output was de-energized. During a sag magnitude and phasing combination that will ultimately cause a drive trip, the DC bus magnitude is clearly dependent on the motor load, and decays at a far slower rate when the motor output is off altogether.

Several variables have been shown to affect the ridethrough characteristics of the textile tension control stand's AC motor drives. While motor loading and sag duration are factors, they are overshadowed by the influence of sag magnitude and phasing combinations. In the case of the experimental process, dropout characteristics were tested for a range of magnitude and phasing combinations. These results are listed in Appendix F. Each test was conducted by delivering a 450ms sag of a certain magnitude and phasing combination. During the tests, the motors were energized and running in the textile process. The recorded response is the drive's ability to ride through the event without dropping out and requiring a user restart.

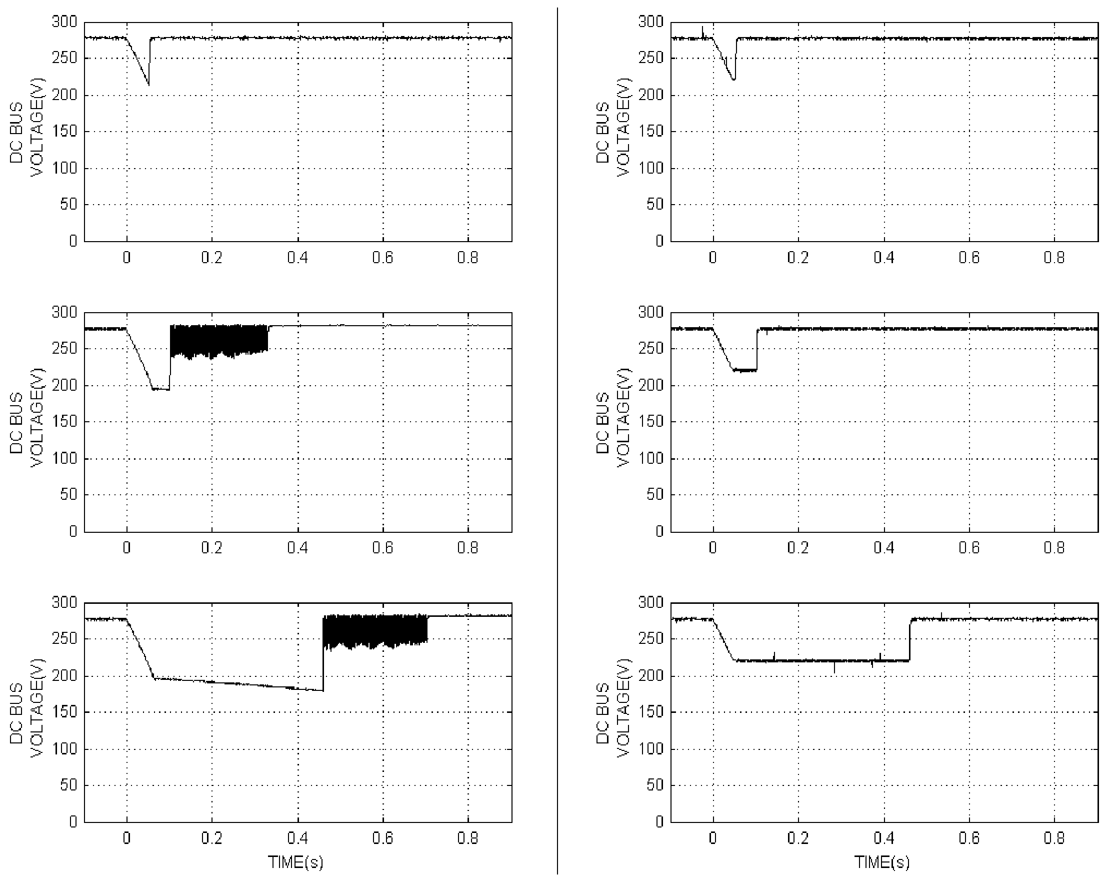


Figure 2.16 Sag duration effects on the motor drive DC bus levels.

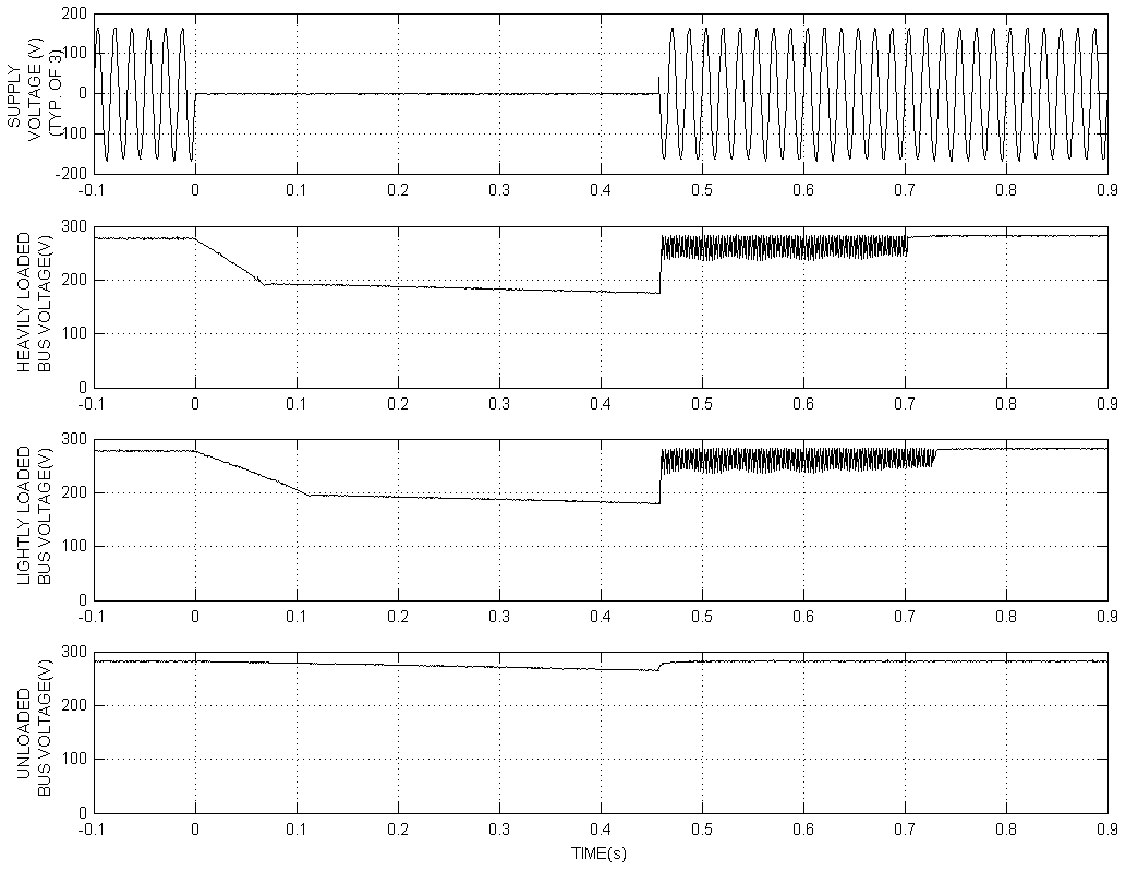


Figure 2.17 Loading effects on the motor drive DC bus decay rates.

Simulation of these effects are implemented in the main Matlab simulation program by forcing drive output voltages to zero after calculations for the DC bus levels indicate that the dropout threshold has been crossed. In the simulation, the measurable sag characteristics are not considered. Instead, the observed DC bus decay rate is inserted directly into the routine to simulate a sag that causes drive dropout.

Additional Immune Hardware

Additional hardware present in the textile tension control stand (see Figure 2.2) was tested with a sag generator for a $450ms$, 0% sag (interruption) and rode through the test events without disturbance. These devices are listed in Table 2.2. Additionally, all AC contactors present in the system were bypassed for system-wide voltage sag tests, since their ridethrough characteristics can be mitigated with inexpensive hold-in devices.

Table 2.2 Tension stand hardware that showed no dropout response.

DEVICE	**NOTES**
Siemens PLC power supply	24VDC, feeds PLC only
Siemens PLC	fed by dedicated supply
5A, 24VDC supply	supplies I/O block, AS-I converter, limit sensors
AS-I network power converter	24VDC to 30VDC converter
AS-I network master	fed by 5A, 24VDC supply through converter
Profibus I/O block	fed by 5A, 24VDC supply

Combined Process

Once individual device voltage sag responses are understood, the integrated process is examined as a whole. If we initially consider the process as one entity, then the voltage sag tolerance may be visualized as shown in Figure 2.18. Each axis of this graph represents the line-neutral voltage magnitude of an incoming power supply phase. Voltage sags that fall within the shaded region cause the process to malfunction, while sags that fall outside the region have no effect.

All sags that were delivered to the textile tension controller to generate Figure 2.18 were 450ms in duration, which allowed ample time for the AC motor drives to trip offline, and for the instrumentation to deviate from their prefault values significantly. The boundary between regions falls at the midpoint between a sag test magnitude point which yielded a disruption and one that did not. Figure 2.18 therefore represents the 450ms process disturbance/no disturbance test results with 20% sag magnitude increments, for a total of 216 tests.

The response may be further divided into regions of specific malfunction. The most catastrophic of these regions is shown in Figure 2.19. This region indicates the sag magnitude and phasing combinations where the AC drives drop offline, which constitutes a complete process shutdown. This type of failure requires user input to restart the process and manually rethread the machine.

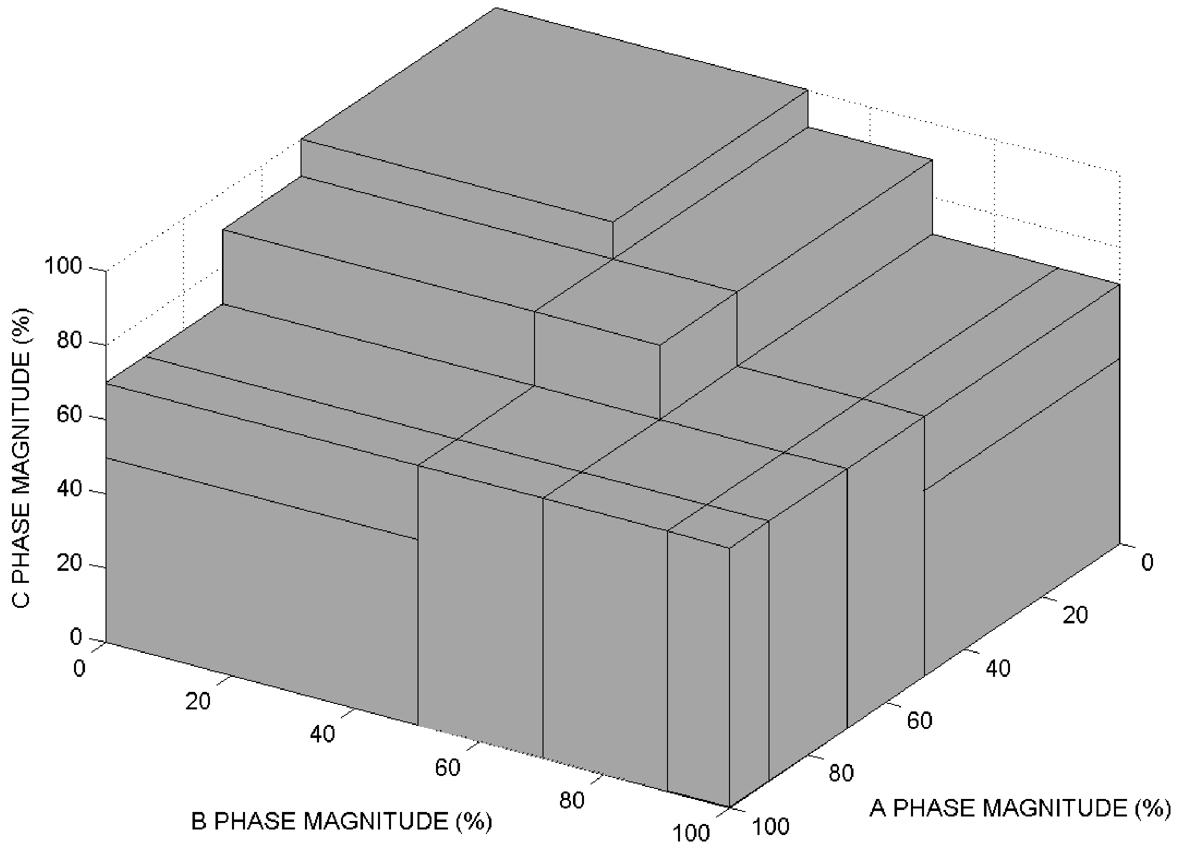


Figure 2.18 Region of voltage sags for any process disturbance in textile tension control system.

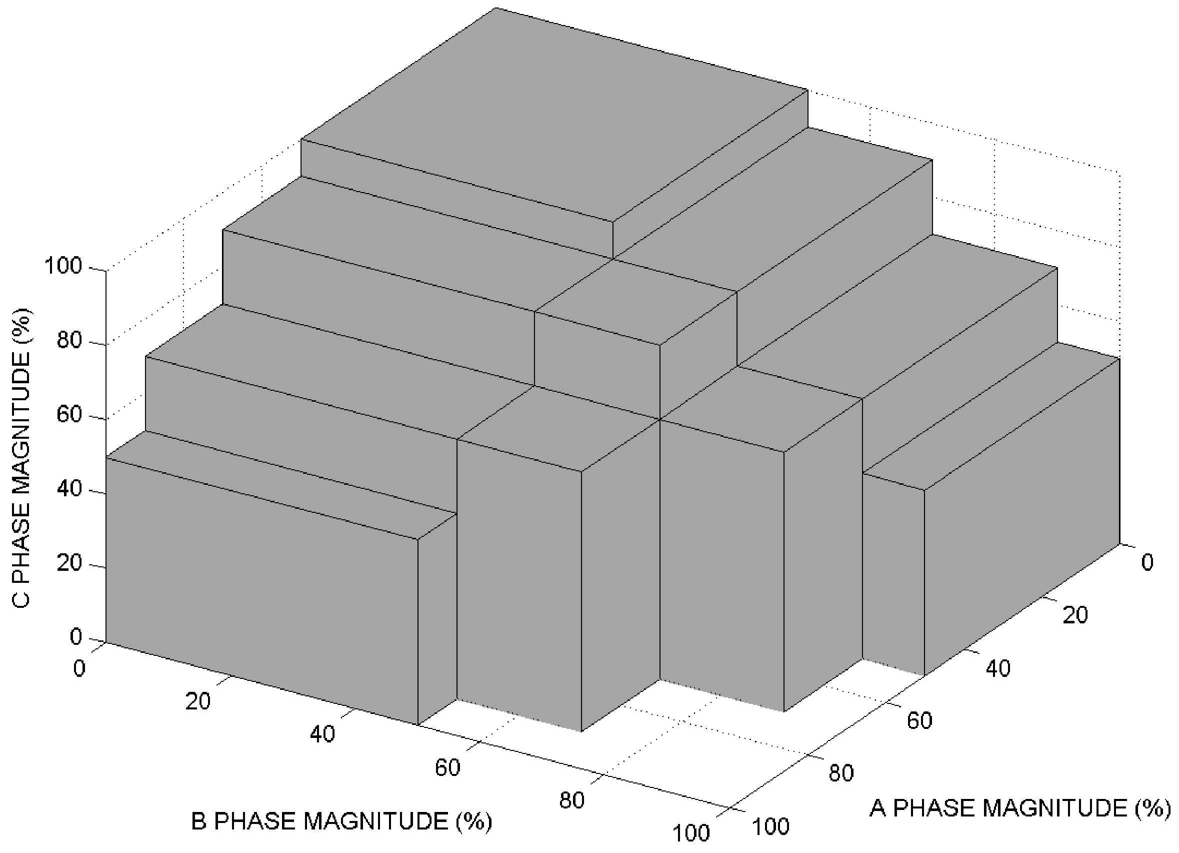


Figure 2.19 Region of voltage sags for AC motor drive dropout in textile tension control system.

The AC drive dropout region boundaries may be described mathematically. Figure 2.20 shows a scatter plot of AC drive voltage sag dropout points in two phases. Here phase C is held at a constant 0% magnitude, while phases A and B are variable. Using the magnitude midpoints between tests that caused a dropout and those that rode through, a threshold is established with the line

$$A + B = 1.5, \tag{2.30}$$

where A and B are the per unit phase voltage magnitudes. This threshold is similar for plots of phases B versus C and C versus A with the missing phase voltage held at zero. Three inequality statements result from these boundary lines. The logical AND of the inequalities form an expression to indicate a sag's presence in the AC drive dropout region. This may be expressed by

$$Q_1 = (A + B < 1.5) \quad AND \quad (B + C < 1.5) \quad AND \quad (C + A < 1.5), \tag{2.31}$$

where Q_1 is a Boolean term indicating sag presence in the AC drive dropout region.

A comparison of actual and simulated behavior for this case is shown in Figure 2.21. An immediate loss of tension occurs when the drives drop offline, and the process linespeed slowly decelerates. Only slight differences exist between the experimental and simulated waveforms. The experimentally recorded tension waveform shows a slightly slower rate of decent to the zero tension state than its simulated counterpart, which may be attributed to the first order delay of the tension cell. In the simulated linespeed waveform, the decay experiences a momentary reversal that is not seen in the measured linespeed. This is caused by the mechanical modeling of the rollers remaining coupled during the zero tension state, while the actual system experiences decoupled rollers with web slack between them.

Although the AC drive dropout region accounts for the majority of possible process malfunctions, another region exists that requires attention. This malfunction region is shown in Figure 2.22, and represents the voltage sags that cause process variables to go out of tolerance. In this region, the AC drives do not trip offline due to a DC bus undervoltage. Instead, the instrumentation delivers erroneous process feedback

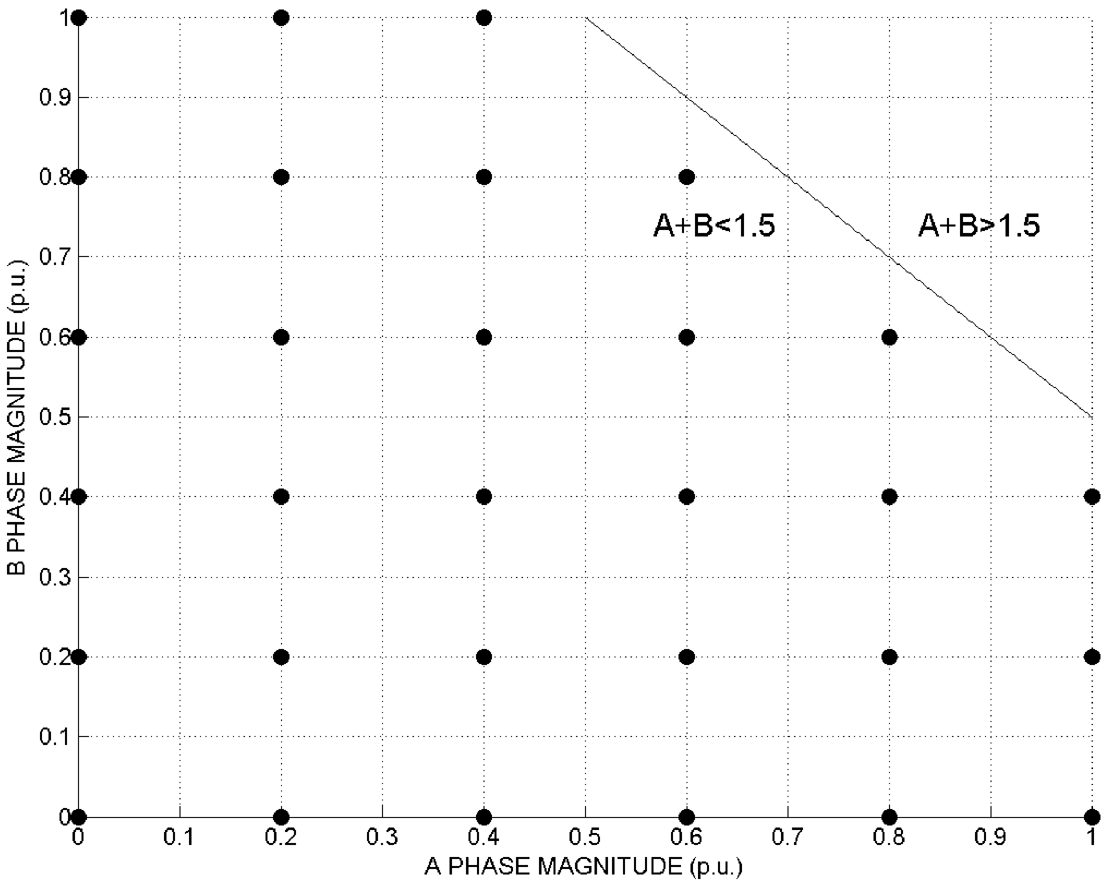


Figure 2.20 Two phase graph of AC drive dropout points and dropout boundary estimation line.

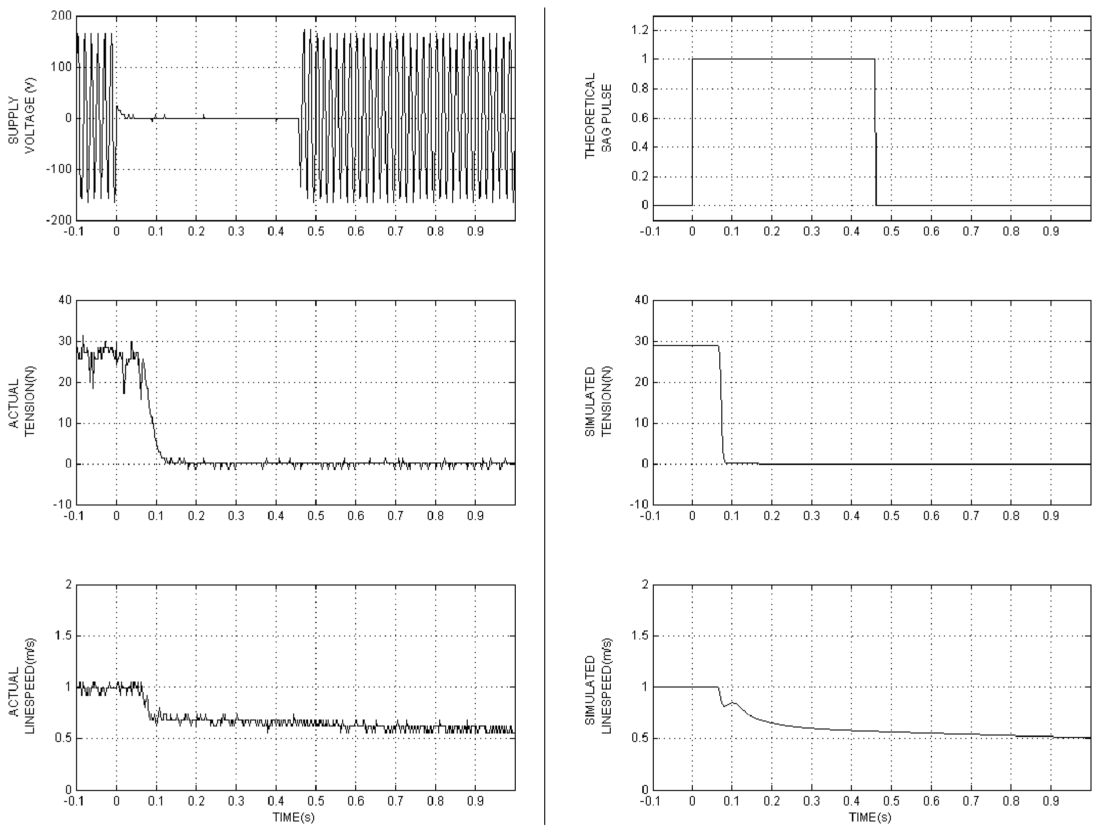


Figure 2.21 Actual and simulated textile tension controller responses during motor drive dropout.

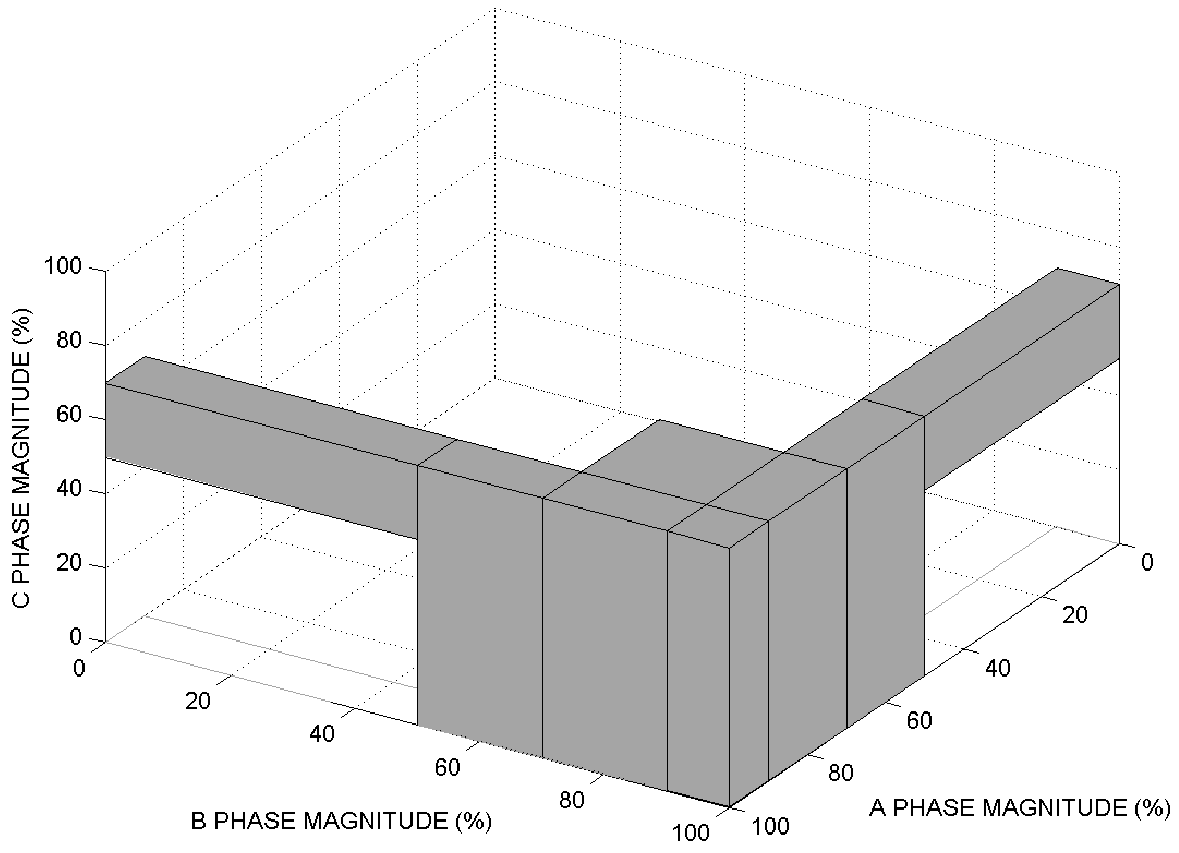


Figure 2.22 Region of voltage sags for instrumentation malfunction in textile tension control system.

information to the controller. This in turn causes the controller to call for a change in the process that is not required, which results in a deviation from desired process values. Some sags that affect the instrumentation are outside the indicated region, but still exist within the AC drive dropout region. They are therefore neglected because AC drive dropout is the more dominant, catastrophic variety of process malfunction.

Sag presence in the faulted instrumentation region may also be described in mathematical terms. By inspection of Figure 2.22, we observe that the faulted instrumentation region is present where the Phase C magnitude is below 70%, and the sag magnitude and phasing combination falls outside the AC drive dropout region. This is expressed as

$$Q_2 = (Q_1 = FALSE) \quad AND \quad (C < 0.7), \tag{2.32}$$

where Q_2 is a Boolean value indicating a sag's presence in the faulted instrumentation region.

Simulation of an instrumentation level malfunction is compared with measured values in Figure 2.23 using the Matlab simulation. A single phase, 0% voltage sag (interruption) is delivered to Phase C, and both the tension cell and tachometers are driven into a faulted state. Their feedback values are misinterpreted as physical disturbances, and compensated for by the PLC control algorithm. The actual web tension is driven to over double its prefault value, and the linespeed sees a significant disturbance as well. The ridethrough, deviation, and recovery segments in the instrument responses are all clearly visible, and their theoretical models follow closely. Discrepancies between the measured and simulated responses can be attributed to slight differences in measured component gains, unmodeled roller imbalance effects, and noise.

Summary

A network connected textile tension control system was constructed with the intended purpose of serving as a test bed for developing software based voltage sag mitigation algorithms. After mathematical analysis and detailed measurement of

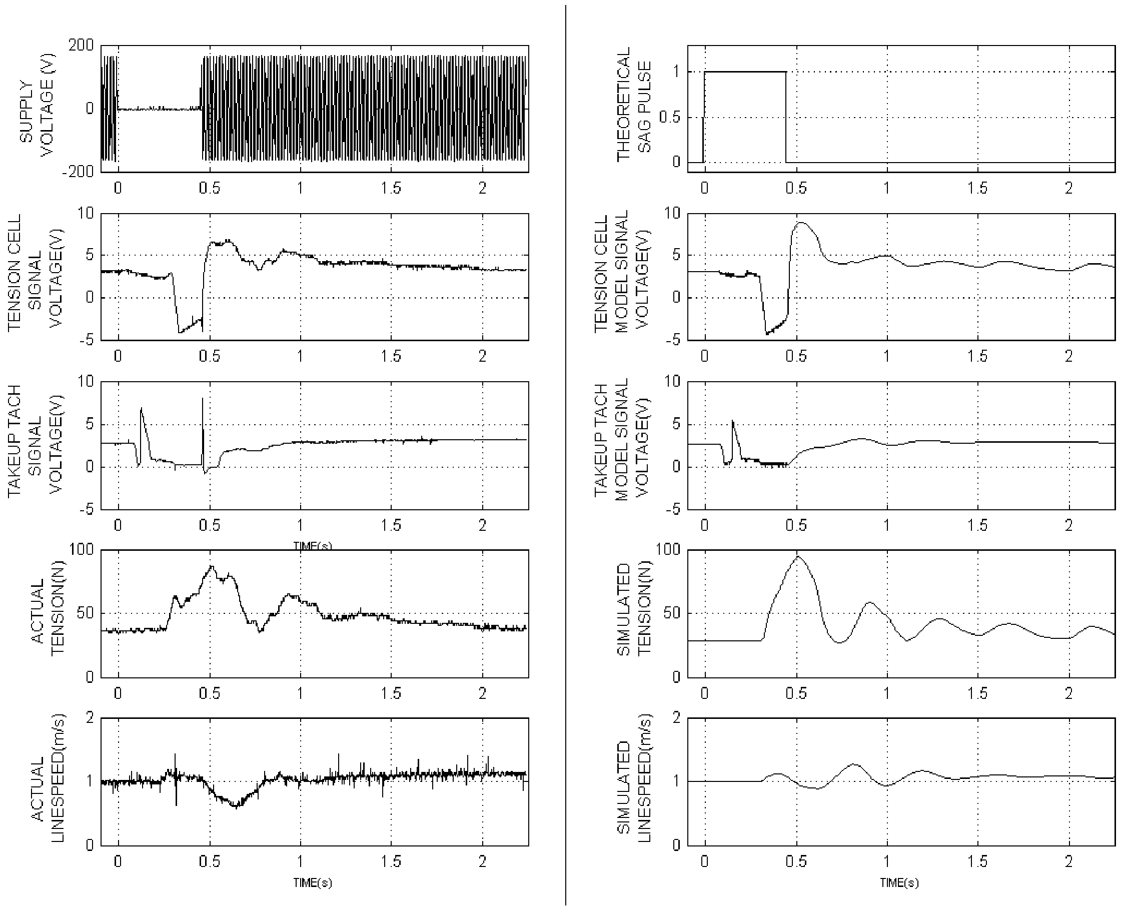


Figure 2.23 Comparison of actual and simulated textile tension controller response to 450ms, 0% sag (interruption) affecting the instrumentation only.

constants, a Matlab based program was written to simulate the tension controller's dynamic behavior.

During analysis of individual device behavior, it was found that an improved instrumentation voltage sag response model was required to effectively predict the process voltage sag output response. This model improves upon previous mathematical instrument response models by accounting for erratic device output and modeling gradual instrument recovery characteristics. The developed model was implemented in the Matlab simulation. AC motor drive voltage sag responses were found to follow a ride through or trip dichotomy, the result of which is highly dependent on sag magnitude and phase combinations. Other devices such as DC power supplies were found to have no effect on the process voltage sag response due to their preexisting ridethrough and light loading.

Using a series of voltage sag tests at $450ms$ durations, varying forms of the combined process response were observed. These consist of three distinct regions:

1. an AC drive dropout region, which was found to be the most catastrophic because of the required user input to remedy;
2. a faulted instrumentation region, which causes corrupted feedback signals to force the existing control algorithm to call for unnecessary corrective action;
3. a no disruption region, where the process behaves normally and rides through the voltage sag.

Simulations of the AC drive dropout region and the faulted instrumentation region were performed and matched with experimental results.

Using the textile tension control stand as a research test bed, our focus now turns to the software architecture and algorithm development used to mitigate the voltage sag responses observed in this system.

CHAPTER 3
SOFTWARE DESIGN

Introduction

A coordinated voltage sag mitigation software suite was designed and implemented for use with the textile tension control stand. This program aims at minimizing the addition of hardware and being unintrusive to existing process controllers.

The software resides on a standalone PC, termed 'Ridethrough PC', and interfaces with both the existing Profibus automation network and the process main three-phase power supply. The Ridethrough PC executes a single program which continually monitors the incoming power supply, detects voltage sags, performs fast sag parameter measurements, determines an expected process response, and executes a mitigation routine based on user input and the expected process voltage sag response. Historical process data can also be stored in the Ridethrough PC. Process control changes are executed in the existing PLC, but are called upon by override commands originating from the Ridethrough PC.

Communications and Interfacing

Figure 3.1 shows the general interfacing topology of the Ridethrough PC. Interfacing details are described in terms of its component subsystems.

PC Specifications

The Ridethrough PC contains a 2.66 GHz Pentium 4 processor, 512 MB of RAM, and runs the Windows XP Professional operating system. LabView version 6.1 is the main software environment for programming and execution of the detection and mitigation routines. LabView was chosen because of its data acquisition interface and analysis capabilities. During the software development process, LabView's graphical programming environment allowed for extensive variable monitoring and analysis, thereby decreasing troubleshooting time.

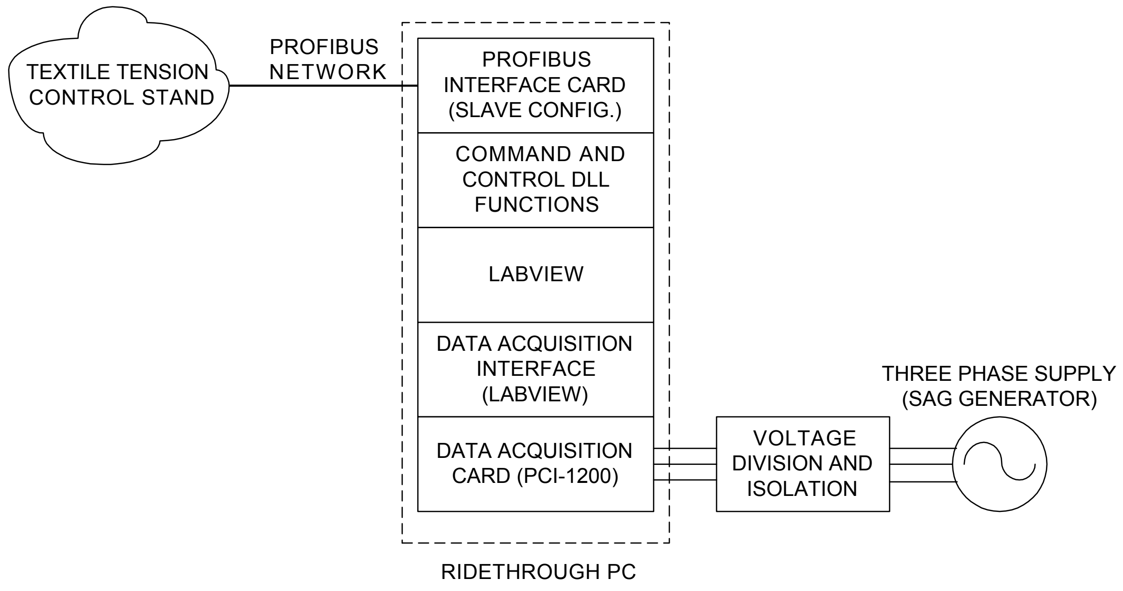

TEXTILE TENSION CONTROL STAND
PROFIBUS NETWORK
PROFIBUS INTERFACE CARD (SLAVE CONFIG.)
COMMAND AND CONTROL DLL FUNCTIONS
LABVIEW
DATA ACQUISITION INTERFACE (LABVIEW)
DATA ACQUISITION CARD (PCI-1200)
VOLTAGE DIVISION AND ISOLATION
THREE PHASE SUPPLY (SAG GENERATOR)
RIDETHROUGH PC

Figure 3.1 Ridethrough PC interface structure.

Profibus Interface

The Profibus interface card used in the Ridethrough PC is a Woodhead/SST automation network interface card, model SST-PFB3-PCI. It is capable of performing as both a Profibus network master or a network slave device. In the case of the Ridethrough PC, it is programmed as a slave device with 16 bytes of transferred input data and 16 bytes of output data. It can function at Profibus baud rates of up to 12Mbps, but is set to operate at the process network's baud rate of 1.5Mbps.

Function calls to the Profibus interface card were performed in LabView by calling individual functions in a dynamic link library (.dll) file. Upon startup, the LabView program calls a subroutine that opens the card for communications, defines network and node parameters, and switches to an online state. During the mitigation program, read and write functions are performed during every scan cycle of the monitor program. The speed at which data is passed to the PLC from the LabView program is therefore defined by the scan cycle time of the ridethrough program and the Profibus network scan cycle time. When execution of the ridethrough program is terminated, a sequence to switch the card offline from the Profibus network is executed. LabView code for Profibus card initialization, I/O, and shutdown is provided in Appendix G.

PLC Communications

The PLC received basic programming modifications to accommodate an additional node on the Profibus network. The additional Ridethrough PC slave node was added to the network using the .gsd messaging configuration file provided by the Profibus card manufacturer. A user-defined setting of 16 input bytes and 16 output bytes was defined in the network configuration environment.

Addition of another node in the Profibus slave scan list added approximately 650μs to the network scan time, and averages $5.9ms$ per scan with the additional node. Memory requirements of the Ridethrough PC did not constitute an appreciable change in PLC memory usage, with the existing controller utilizing only 10% of the available 98,304 bytes of memory space.

Data Acquisition

A National Instruments PCI-1200 data acquisition card was used for power supply signal monitoring. This card allows up to eight analog inputs (0-10V or $\pm$5V), and samples inputs with 12 bit resolution at a maximum rate of 100kS/s. The Ridethrough PC uses four of these analog input channels. Three are for each supply voltage, and the fourth is used for a trigger signal during testing. Sampling rates are set at 6kHz for each channel, which yields 100 data points per cycle of 60Hz voltage. Analog and digital outputs are also available with the PCI-1200.

Initialization and access to the data acquisition card is accomplished through functions provided with LabView. Parameters of device identification, desired input channels, and sampling rates are user defined inputs to these functions. Outputs of these functions are raw data, actual sampling times, and scan buffer backlog levels.

Power Supply Interface

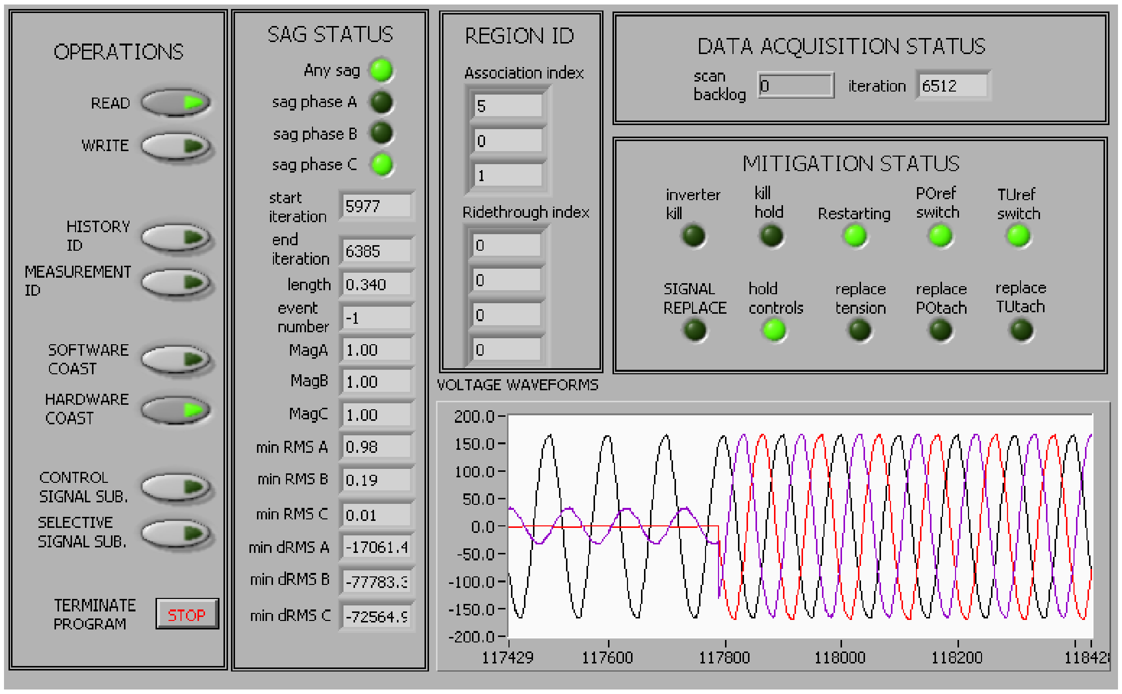
The data acquisition card is connected to the main three phase power supply through separate voltage dividers that output 9.24V peak-to-peak for a 120VAC RMS input. The signals are then isolated with a Gould voltage isolation amplifier set to a 1:1 attenuation. These signals are then fed directly into the data acquisition card.

User Interface

After calling initialization functions for the Profibus and data acquisition cards, the ridethrough program executes a user-terminated while loop in which user commands may dictate the following:

1. Read historical process response information into memory.
2. Write recorded response information in a data file after program termination (for use in subsequent program executions).
3. Associate incoming sags with those recorded during process data accumulation.
4. Calculate the process expected voltage sag response.
5. Execute ridethrough algorithms based on user algorithm selection and the expected process response.

Indicators of sag magnitude and phasing are available through the user interface. The graphical environment also allows for a 'virtual oscilloscope' display of incoming voltage waveforms. The user interface is shown in Figure 3.2.


OPERATIONS
READ
WRITE
HISTORY ID
MEASUREMENT ID
SOFTWARE COAST
HARDWARE COAST
CONTROL SIGNAL SUB.
SELECTIVE SIGNAL SUB.
TERMINATE PROGRAM STOP

SAG STATUS
Any sag
sag phase A
sag phase B
sag phase C
start iteration 5977
end iteration 6385
length 0.340
event number -1
MagA 1.00
MagB 1.00
MagC 1.00
min RMS A 0.98
min RMS B 0.19
min RMS C 0.01
min dRMS A -17061.4
min dRMS B -77783.3
min dRMS C -72564.9

REGION ID
Association index
5
0
1
Ridethrough index
0
0
0
0

DATA ACQUISITION STATUS
scan backlog 0
iteration 6512

MITIGATION STATUS
inverter kill
kill hold
Restarting
POref switch
TUref switch
SIGNAL REPLACE
hold controls
replace tension
replace POtach
replace TUtach

VOLTAGE WAVEFORMS

Figure 3.2 Screenshot of Ridethrough PC user interface.

Existing System Modifications

Changes to the existing system consist of an additional subroutine inserted at the end of the main PLC ladder program, and a series of switching commands included in the main program. The existing PLC program is modified to continuously output critical process data and node status information to the Ridethrough PC, and also accept alternate control commands dictated by the Ridethrough PC during and after a voltage sag.

Signal Switching

Under normal operation, the main PLC program transfers process data from the Profibus input buffer to the PLC memory area. Interrupt switches were inserted in all of the main signal transfer commands, which is achieved with a 'normally closed contact' operation. This enables the move command to execute if the transfer bit is low. Transfer bits are defined in the Ridethrough PC control word, and their corresponding signals are listed in Table 3.1. The main PLC ladder logic code is listed in Appendix H.

Also included in the control word is a drive coast command, which sends motor drive output disable commands on a rising edge and drive restart commands on a falling edge.

Table 3.1 Ridethrough PC control word structure.

BIT(S)	**COMMAND**
0	Drive coast command
1-7	unused
8	Replace bit - tension value
9	Replace bit - payoff tachometer value
10	Replace bit - takeup tachometer value
11	Replace bit - payoff drive output
12	Replace bit - takeup drive output
12	Replace bit - linespeed setpoint
14	Replace bit - tension setpoint
15	unused

PLC Subroutine

The added PLC subroutine transfers raw process data to the Ridethrough PC for analysis and storage along with a word for motor drive and I/O node status data (Table 3.2). Ladder code for the subroutine is listed in Appendix H, while functionality is described in the flowchart in Figure 3.3. The subroutine is called during every PLC scan cycle, and adds approximately $4ms$ to the total program execution time.

Table 3.2 Ridethrough PC status word structure.

BIT(S)	**STATUS**
0	Payoff drive active
1	Takeup drive active
2	I/O block active
3-15	unused

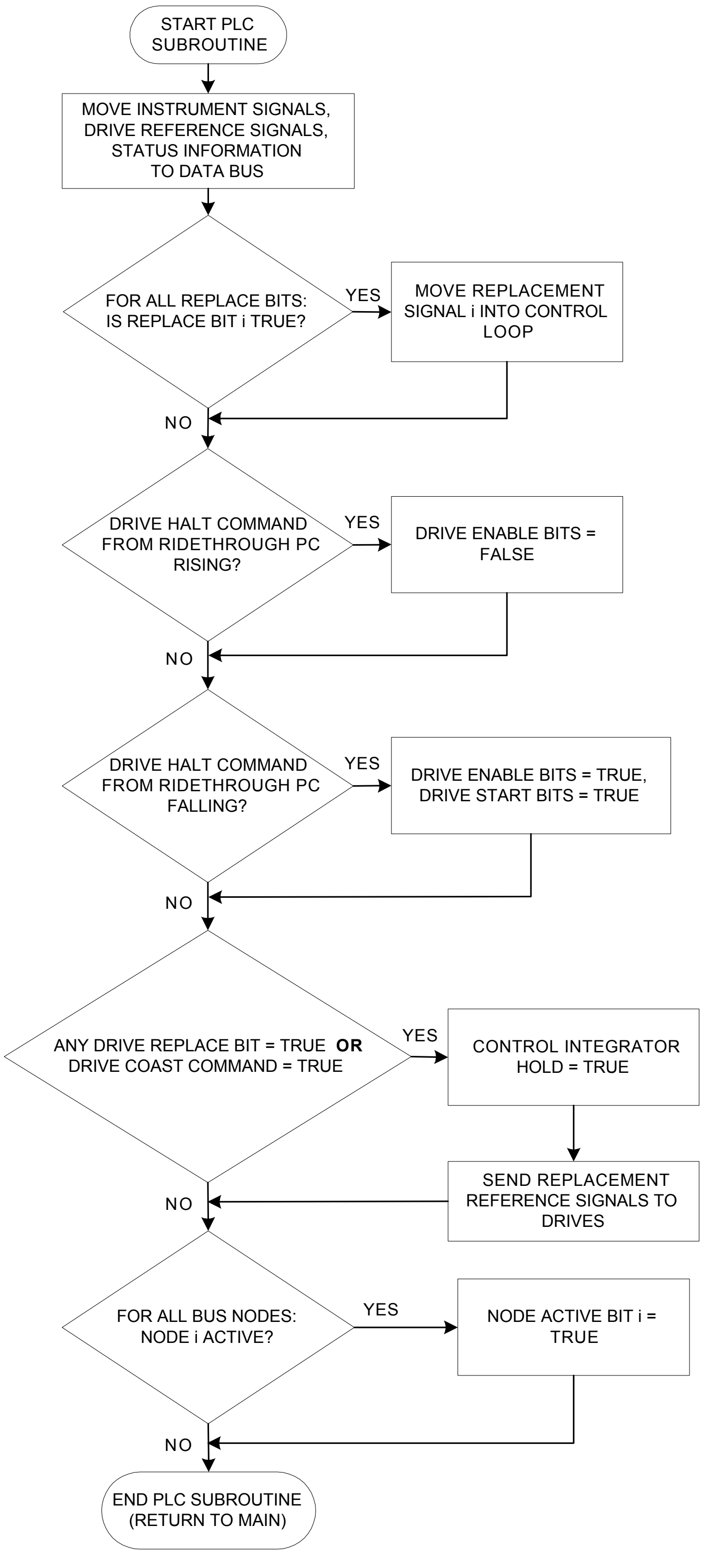

START PLC SUBROUTINE
MOVE INSTRUMENT SIGNALS, DRIVE REFERENCE SIGNALS, STATUS INFORMATION TO DATA BUS
FOR ALL REPLACE BITS: IS REPLACE BIT i TRUE?
YES
MOVE REPLACEMENT SIGNAL i INTO CONTROL LOOP
NO
DRIVE HALT COMMAND FROM RIDETHROUGH PC RISING?
YES
DRIVE ENABLE BITS = FALSE
NO
DRIVE HALT COMMAND FROM RIDETHROUGH PC FALLING?
YES
DRIVE ENABLE BITS = TRUE, DRIVE START BITS = TRUE
NO
ANY DRIVE REPLACE BIT = TRUE OR DRIVE COAST COMMAND = TRUE
YES
CONTROL INTEGRATOR HOLD = TRUE
SEND REPLACEMENT REFERENCE SIGNALS TO DRIVES
NO
FOR ALL BUS NODES: NODE i ACTIVE?
YES
NODE ACTIVE BIT i = TRUE
NO
END PLC SUBROUTINE (RETURN TO MAIN)

Figure 3.3 Flowchart of added PLC subroutine.

Additional Commands

Additional commands introduced into the PLC program were minimal. Integral control hold bits were added to the PI control blocks for the linespeed and tension controllers. Provisions for these commands already existed in the PI control blocks, but were unused before the addition of the Ridethrough PC. Lastly, the Ridethrough PC subroutine call command was added. This executes during every program cycle of the main PLC program

Voltage Sag Detection

Figure 3.4 shows a flowchart for the method of voltage sag detection and measurement. The Ridethrough PC accepts three phase voltage signals and samples them at a rate of 100 sample points per input voltage cycle (6kHz). The sampled data is delivered to the main program and subjected to a sag detection algorithm which generates a Boolean detect signal for each phase. Detection is performed by analysis of RMS voltage derivatives. Sag detect signals are then used by the main program to initiate data recording or process mitigation routines.

For each while loop cycle of the ridethrough program, five voltage data points are sampled from each phase. After scaling the input signals to compensate for the input voltage dividers, the data points are inserted into an existing voltage data array of 100 elements, where the oldest five samples are discarded and the newly sampled points are shift-inserted. After the discard and insert operation, the 100 element array is used to calculate the present RMS voltage with

$$V_{RMS_k} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} v_i^2}, \tag{3.1}$$

where the array size $n = 100$, k is the cycle index for the ridethrough program, and v_i the voltage sample data points. The RMS derivative is then calculated using

$$\frac{d}{dt}V_{RMS_k} = \frac{V_{RMS_k} - V_{RMS_{k-1}}}{dt} \tag{3.2}$$

where dt is the time differential between Ridethrough PC cycles, or

$$dt = \frac{5}{f_{sample}} = 0.833ms. \tag{3.3}$$

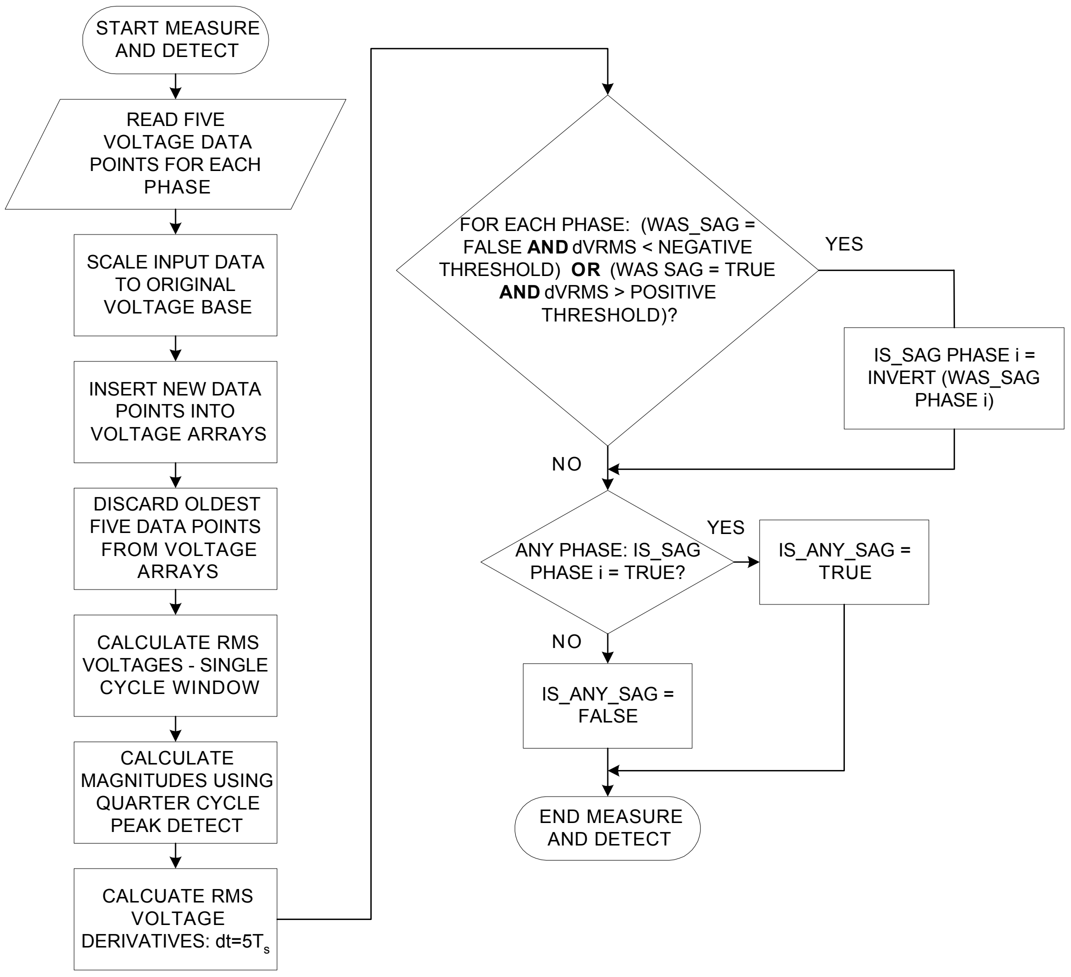

START MEASURE AND DETECT
READ FIVE VOLTAGE DATA POINTS FOR EACH PHASE
SCALE INPUT DATA TO ORIGINAL VOLTAGE BASE
INSERT NEW DATA POINTS INTO VOLTAGE ARRAYS
DISCARD OLDEST FIVE DATA POINTS FROM VOLTAGE ARRAYS
CALCULATE RMS VOLTAGES - SINGLE CYCLE WINDOW
CALCULATE MAGNITUDES USING QUARTER CYCLE PEAK DETECT
CALCUATE RMS VOLTAGE DERIVATIVES: dt=5T$_s$
FOR EACH PHASE: (WAS_SAG = FALSE AND dVRMS < NEGATIVE THRESHOLD) OR (WAS SAG = TRUE AND dVRMS > POSITIVE THRESHOLD)?
YES
IS_SAG PHASE i = INVERT (WAS_SAG PHASE i)
NO
ANY PHASE: IS_SAG PHASE i = TRUE?
YES
IS_ANY_SAG = TRUE
NO
IS_ANY_SAG = FALSE
END MEASURE AND DETECT

Figure 3.4 Flowchart of voltage sag measurement and detection routine.

In the prefault and postfault cases, the RMS derivative remains at zero. Sag inception is detected when the RMS derivative crosses a negative threshold, and recovery is detected when the RMS derivative crosses a positive threshold.

After initializing all detect signals to logic ‘false’, the detection algorithm performs an invert operation to the detect signal when the first below-threshold RMS derivative voltage occurs. Subsequent crossings of the negative threshold are ignored. The detect signal is inverted back to the logic ‘false’ state upon the first above-threshold RMS derivative voltage seen while in the ‘true’ state. Initial conditions are of critical importance, as unpaired inception or recovery triggers could potentially lead to a false detect signal.

Theoretical Response

The voltage sag algorithm was implemented in Matlab to simulate its performance. Figures 3.5-3.9 show several theoretical sag detection responses. Detection thresholds, set at $\pm 2kV/s$, are shown as bold lines that intersect with the RMS voltage derivative signal. The point on wave effect on response time is evident when comparing Figures 3.5 and 3.6. Here, the low magnitude sag of 20% shows a difference of $1.7ms$ in triggering times between $0°$ and $90°$. Nevertheless, the sag detection algorithm triggers in less than a quarter cycle - a significant improvement from using a basic RMS calculation. Figures 3.7 and 3.8 show the theoretical sag detection algorithm response when applied to higher magnitude sags of 80%. Detection times are still below one quarter cycle, but are slightly increased for the $0°$ point on wave case.

RMS derivative threshold levels are critical in adjusting the detection response. This is evident in Figure 3.9, where a sag of 89% does force the RMS voltage derivative signal below its detection threshold. This case of a missed detection for shallow sags may be alleviated by decreasing the detection thresholds, but at the cost of increased sensitivity to transients and noise.

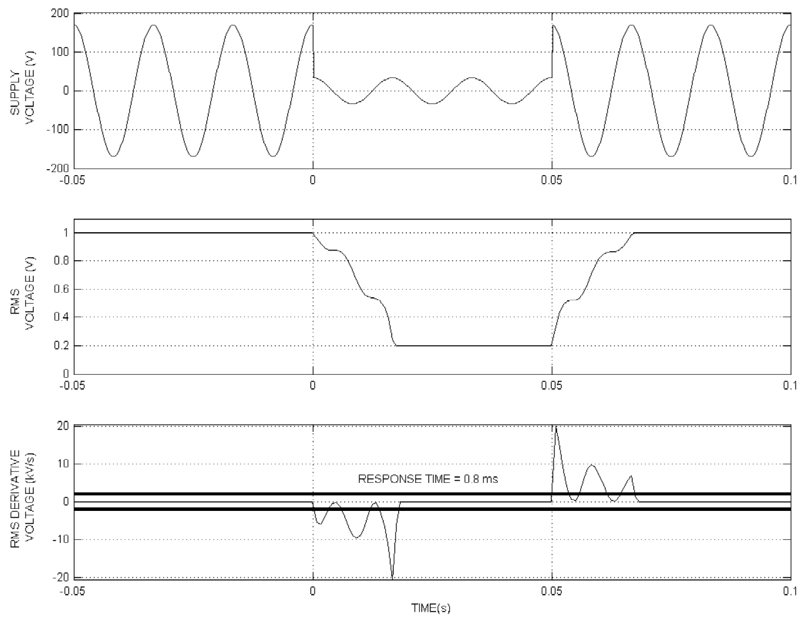


Figure 3.5 Simulated voltage sag (20%, 90°) with RMS voltage, RMS voltage derivative, and threshold signals.

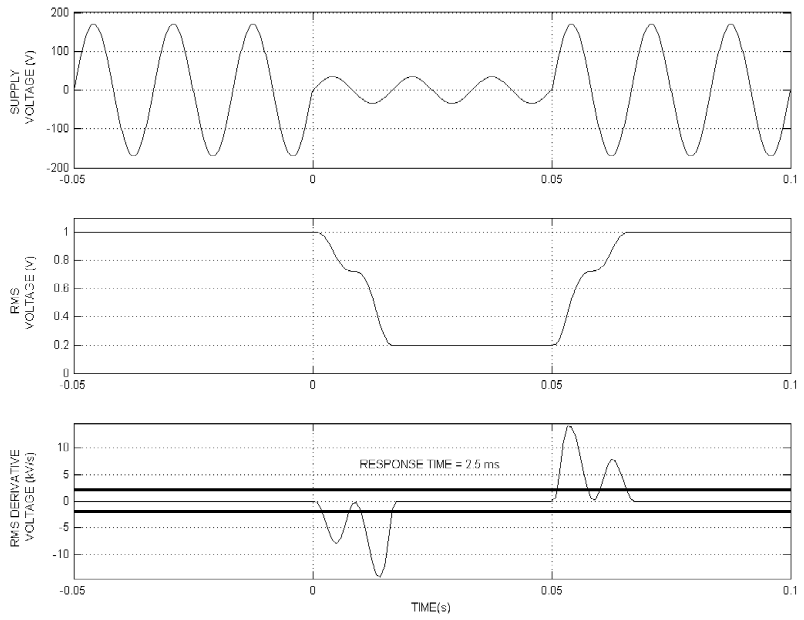


Figure 3.6 Simulated voltage sag (20%, 0°) with RMS voltage, RMS voltage derivative, and threshold signals.

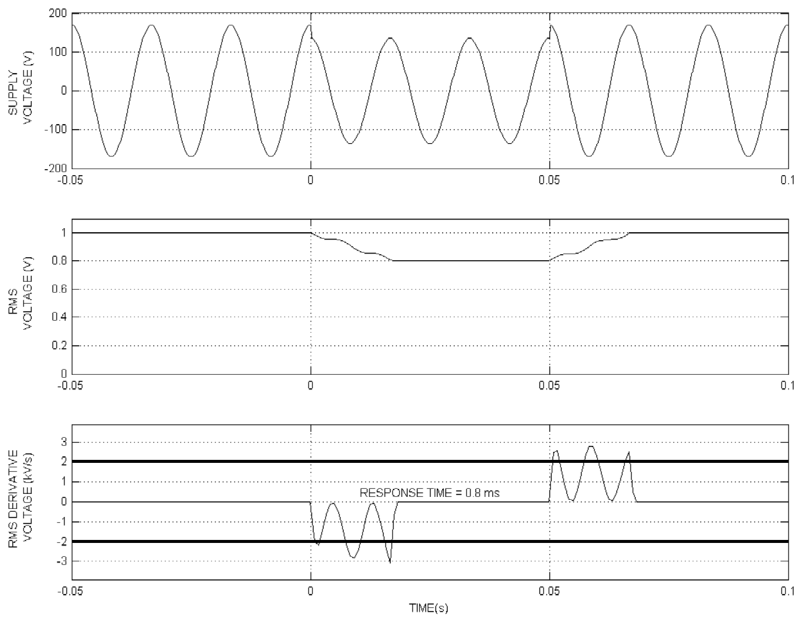


Figure 3.7 Simulated voltage sag (80%, 90°) with RMS voltage, RMS voltage derivative, and threshold signals.

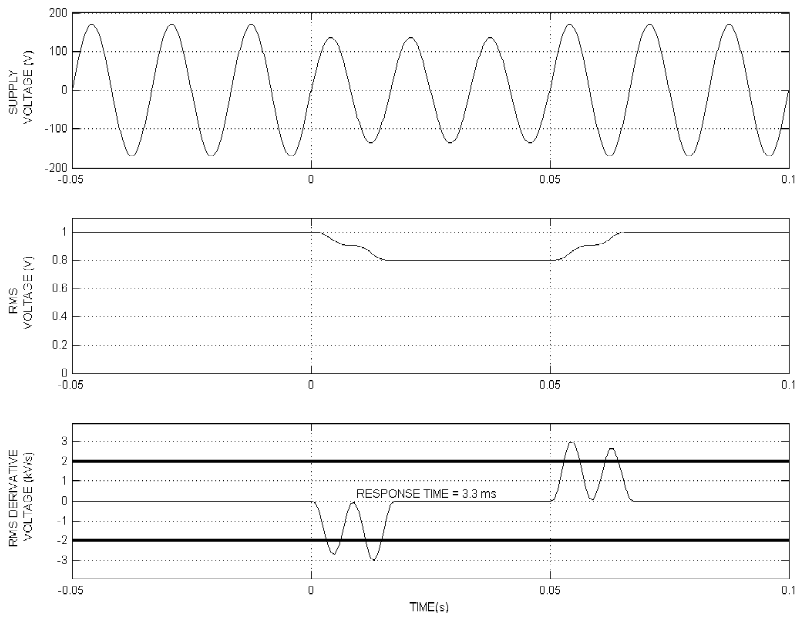


Figure 3.8 Simulated voltage sag (80%, 0°) with RMS voltage, RMS voltage derivative, and threshold signals.

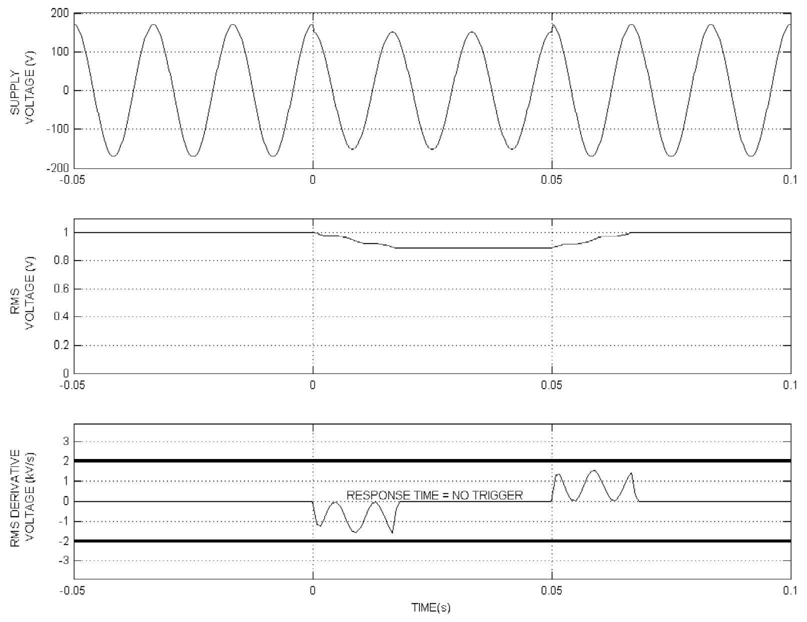


Figure 3.9 Simulated voltage sag (89%, 90°) with RMS voltage, RMS voltage derivative, and threshold signals.

Data Manipulation

Transferred Data

Data is transferred from the PLC to the Ridethrough PC in 16 byte (8 word) blocks. Tables 3.3 and 3.4 detail the structure of these data blocks. A transfer occurs during every Profibus network scan cycle, but the data updates to the PLC occur asynchronously. The PLC program and peripheral scan rates contribute to the actual update times for data sent to the Ridethrough PC, and the main Ridethrough PC loop execution time contributes to the update rate for the data sent to the PLC.

Event Data

Event data is stored in a multidimensional array that increases its main index once for each recorded event. Process signal data is stored in a series of subarrays that are updated in a shift/discard technique similar to the array update method used to monitor line voltages. In addition to these signal arrays, values of event length in

Table 3.3 Data transferred from PLC to ridethrough PC.

WORD	DESCRIPTION
0	Tension
1	Status word
2	Takeup tachometer
3	Payoff tachometer
4	Takeup output reference
5	Payoff output reference
6	Linespeed setpoint
7	Tension setpoint

Table 3.4 Data transferred from ridethrough PC to PLC.

WORD	DESCRIPTION
0	Tension replacement value
1	Control word
2	Takeup tachometer replacement value
3	Payoff tachometer replacement value
4	Takeup output reference replacement value
5	Payoff output reference replacement value
6	Linespeed setpoint replacement value
7	Tension setpoint replacement value

iterations and seconds is stored. Minimum RMS voltages and minimum RMS voltage derivatives for each phase are also measured for storage in the main array. Lastly, a Boolean array that indicates the phase combination of the sag is stored. Table 3.5 summarizes the values stored for each event. The storage trigger command occurs $1.110s$ after the beginning of a voltage sag, and captures $278ms$ of prefault data along with $1.110s$ of fault/postfault data. These lengths were determined in practice to be of sufficient length to capture all relevant process event data, while not affecting program execution by placing excessive demands on system resources. This operation only stores the data in memory; a file storage command is executed after the program has terminated.

The goal of creating a large process history is for future reference in the event of an incoming voltage sag. If measured reactions to sags are known, then the Ridethrough PC may access the data and determine the expected response to an incoming sag. A mitigation solution may then be applied to match the expected response. The final size of the event data array is 216 elements. With a magnitude step size of 20%, this stores an event for every allowable magnitude and phasing combination.

Table 3.5 Process data storage array elements.

ITEM	**DESCRIPTION**
0	Tension subarray (1667 elements)
1	Status subarray (1667 elements)
2	Takeup tachometer subarray (1667 elements)
3	Payoff tachometer subarray (1667 elements)
4	Takeup output subarray (1667 elements)
5	Payoff output subarray (1667 elements)
6	Linespeed setpoint subarray (1667 elements)
7	Tension setpoint subarray (1667 elements)
8	Length of event in iterations
9	Minimum phase A RMS voltage (%)
10	Minimum phase B RMS voltage (%)
11	Minimum phase C RMS voltage (%)
12	Minimum phase A RMS derivative voltage (V/s)
13	Minimum phase B RMS derivative voltage (V/s)
14	Minimum phase C RMS derivative voltage (V/s)
15	Length of event in seconds
16	Phasing combination Boolean array

Storage and Retrieval

Upon termination of the main program loop, event data may be recorded in a text file. The file grows to a size of approximately 23MB for the desired 216 stored events. Retrieval is performed at the beginning of program execution, and requires a user input to access the data file. The text file data is parsed into memory in single-event sized blocks. After completion of the read-in process, the original event data array is rearranged into a 6 x 6 x 6 array, where each dimension of the new data array represents an input voltage phase with varying magnitude (20% increments). Process historical data is later retrieved through this array, where the indexing terms correspond to the measured RMS phase voltage magnitudes.

Event Analysis and Association

Event characteristics are computed and stored using the procedure detailed in Figure 3.10. Among these calculations are minimum sag magnitude, minimum RMS voltage derivative values, sag length, and prefault signal averages. They are stored after the event has ended, but are updated every program cycle.

When a voltage sag is detected, a magnitude and phasing analysis is performed. This analysis takes one-quarter of a 60Hz cycle to complete. Once the sag magnitude and phasing characteristics are determined, association with a recorded event and response region identification may take place.

Sag Measurement

Sag magnitudes are measured with a peak detection routine. This is performed with an analysis of the input power data arrays. Given that

$$V_\Phi(t) = V_m \sin(\omega t), \tag{3.4}$$

and

$$\frac{d}{dt} V_\Phi(t) = V_m \omega \cos(\omega t), \tag{3.5}$$

we may be assured that the peak magnitude of the incoming voltage is present in either a one-quarter cycle sample set of the raw voltage data or the rescaled derivatives calculated with the same one-quarter cycle set. The aim is therefore to find the maximum magnitude amongst two sets of data. The first is a 25 sample set of the most recently acquired raw input voltage data,

$$V_{\Phi 1}(k) = V_\Phi(k), \quad k = 0...24, \tag{3.6}$$

where k is the array index. The second set contains 25 scaled derivatives determined by

$$V_{\Phi 2}(k) = \frac{V_\Phi(k) - V_\Phi(k-1)}{\omega T_s}, \quad k = 1...25. \tag{3.7}$$

where ω is the frequency in rad/s and T_s is the sampling period. The index k in Equation 3.7 exceeds that of Equation 3.6 by one, which makes the overall required

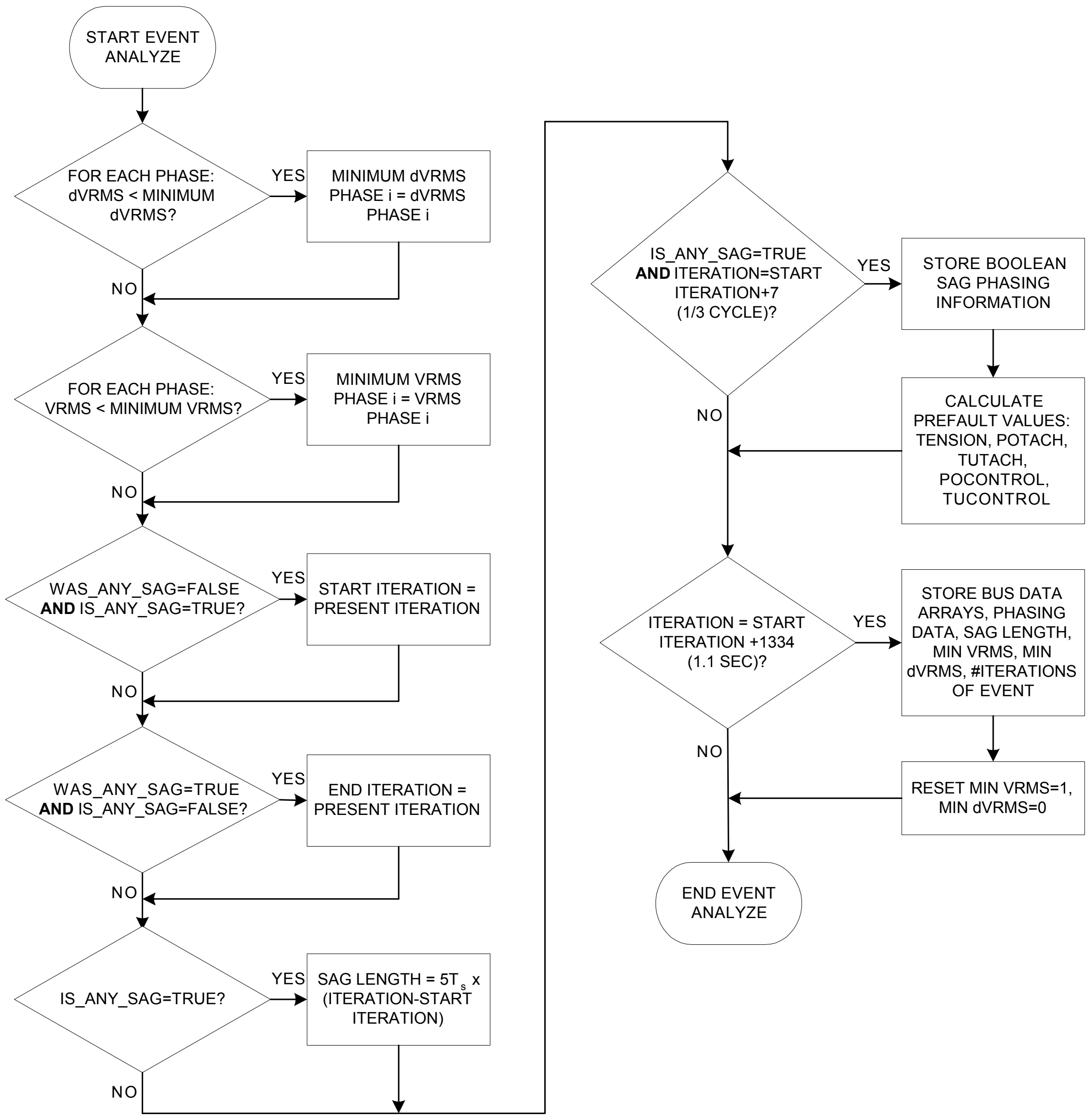

START EVENT ANALYZE
FOR EACH PHASE: dVRMS < MINIMUM dVRMS?
YES
MINIMUM dVRMS PHASE i = dVRMS PHASE i
NO
FOR EACH PHASE: VRMS < MINIMUM VRMS?
YES
MINIMUM VRMS PHASE i = VRMS PHASE i
NO
WAS_ANY_SAG=FALSE AND IS_ANY_SAG=TRUE?
YES
START ITERATION = PRESENT ITERATION
NO
WAS_ANY_SAG=TRUE AND IS_ANY_SAG=FALSE?
YES
END ITERATION = PRESENT ITERATION
NO
IS_ANY_SAG=TRUE?
YES
SAG LENGTH = 5T_s x (ITERATION-START ITERATION)
NO
IS_ANY_SAG=TRUE AND ITERATION=START ITERATION+7 (1/3 CYCLE)?
YES
STORE BOOLEAN SAG PHASING INFORMATION
CALCULATE PREFAULT VALUES: TENSION, POTACH, TUTACH, POCONTROL, TUCONTROL
NO
ITERATION = START ITERATION +1334 (1.1 SEC)?
YES
STORE BUS DATA ARRAYS, PHASING DATA, SAG LENGTH, MIN VRMS, MIN dVRMS, #ITERATIONS OF EVENT
RESET MIN VRMS=1, MIN dVRMS=0
NO
END EVENT ANALYZE

Figure 3.10 Flowchart of event analysis in ridethrough PC.

sample time one sample greater than a quarter cycle. The peak magnitude is then determined with

$$V_m = \max(|V_{\Phi 1}|, |V_{\Phi 2}|). \tag{3.8}$$

Figures 3.11 and 3.12 show the theoretical response to the peak detect routines given sags of 20% depth and 0° and 90° point on wave, respectively.

Event Association

After the phase voltage magnitudes have been determined, they are each divided by their nominal value to determine the per unit voltage sag magnitude. These are converted into integer array indices used to access the process history array. The array indices are generated using a nearest integer function of the form

$$w_\Phi = \left[\frac{V_{p.u.}}{\rho_v}\right], \tag{3.9}$$

where w_Φ is an array dimension index, $V_{p.u.}$ the per unit voltage magnitude, and ρ_v the per unit voltage step size of indexed sag magnitudes. With the Ridethrough PC using 20% increments, the output range of this function is $0 \leq w_\Phi \leq 6$.

After indices are generated for each phase, they are used to access the 6 x 6 x 6 data array to yield a process response recorded prior to the present voltage sag. Analysis of this data may then be used to determine the mitigation method.

Identification of Response Region

In the previous chapter, three different voltage sag response regions were identified for the textile tension control stand. They consist of an AC drive dropout region, a faulted instrumentation region, and a no disruption region. The measured magnitude and phasing combination is used to determine the response region of an incoming sag. The user selects one of two possible identification methods.

Identification Using Unmitigated Process History

After retrieving unmitigated process response data that corresponds to an in-progress voltage sag, the historical data is analyzed to determine the response region in which it lies. For determining membership in the AC drive dropout region, the

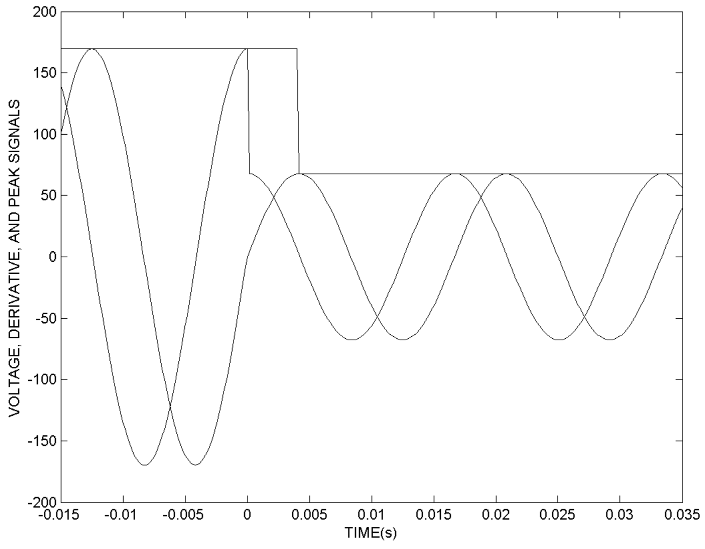


Figure 3.11 Theoretical response of peak detect routine to 40%, 0° point on wave voltage sag.

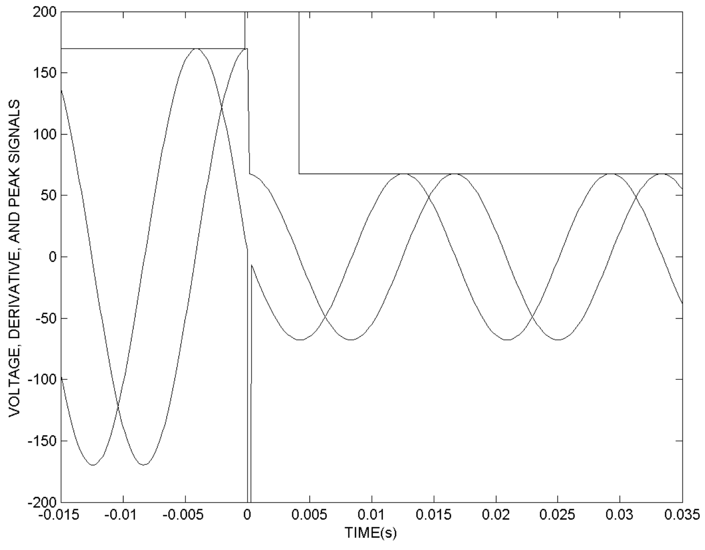


Figure 3.12 Theoretical response of peak detect routine to 40%, 90° point on wave voltage sag.

recorded status word from the PLC is scanned for fault bits indicating drive dropout. Membership in the faulted instrumentation region requires analysis of the recorded fault/postfault signal data and comparison to the average prefault value. Upper and lower tolerance thresholds are established, and the present sag is said to be in the faulted instrumentation region if any of the critical signals fall outside these thresholds. Upper and lower tolerance thresholds were set between $\pm5\%$-15% of their recorded prefault values. Steady state operating conditions are assumed, and the thresholds are left as user defined quantities. Identification of the no disruption region is unnecessary because it requires no mitigation response. This determination algorithm requires a fair amount of memory to store the history data block, but holds the distinct advantage of not requiring mathematical models to identify response regions. Figure 3.13 shows the flowchart for this method.

Identification Using Region Boundary Definitions

The region boundary definitions described in Equations 2.30, 2.31, and 2.32 may be implemented in code to perform a less computationally expensive expected response identification. This method does not require referencing historical process data, but instead requires an estimation of device thresholds and region boundaries. The flowchart for region boundary response identification is shown in Figure 3.14. Again, specific identification of the no disruption region is unnecessary, as it does not mandate a mitigation response.

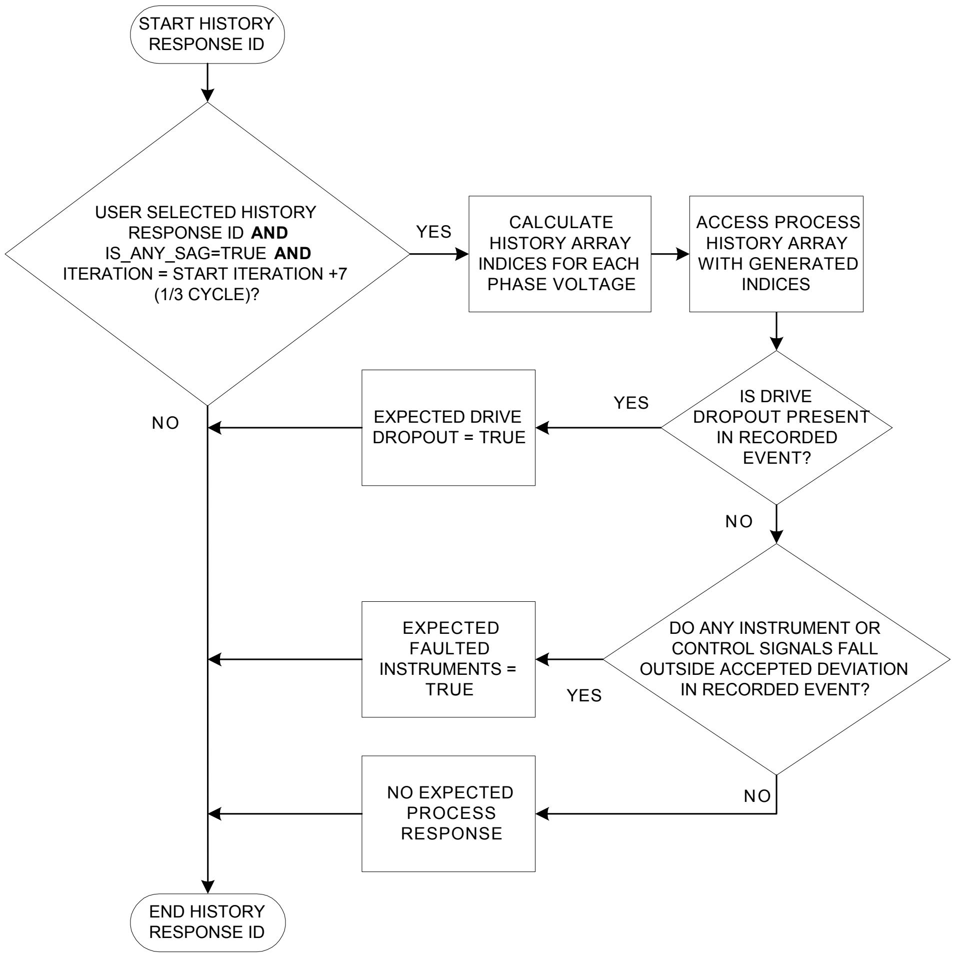

START HISTORY RESPONSE ID
USER SELECTED HISTORY RESPONSE ID AND IS_ANY_SAG=TRUE AND ITERATION = START ITERATION +7 (1/3 CYCLE)?
YES
CALCULATE HISTORY ARRAY INDICES FOR EACH PHASE VOLTAGE
ACCESS PROCESS HISTORY ARRAY WITH GENERATED INDICES
IS DRIVE DROPOUT PRESENT IN RECORDED EVENT?
YES
EXPECTED DRIVE DROPOUT = TRUE
NO
NO
DO ANY INSTRUMENT OR CONTROL SIGNALS FALL OUTSIDE ACCEPTED DEVIATION IN RECORDED EVENT?
YES
EXPECTED FAULTED INSTRUMENTS = TRUE
NO
NO EXPECTED PROCESS RESPONSE
END HISTORY RESPONSE ID

Figure 3.13 Flowchart for identification of expected process voltage sag response using process history.

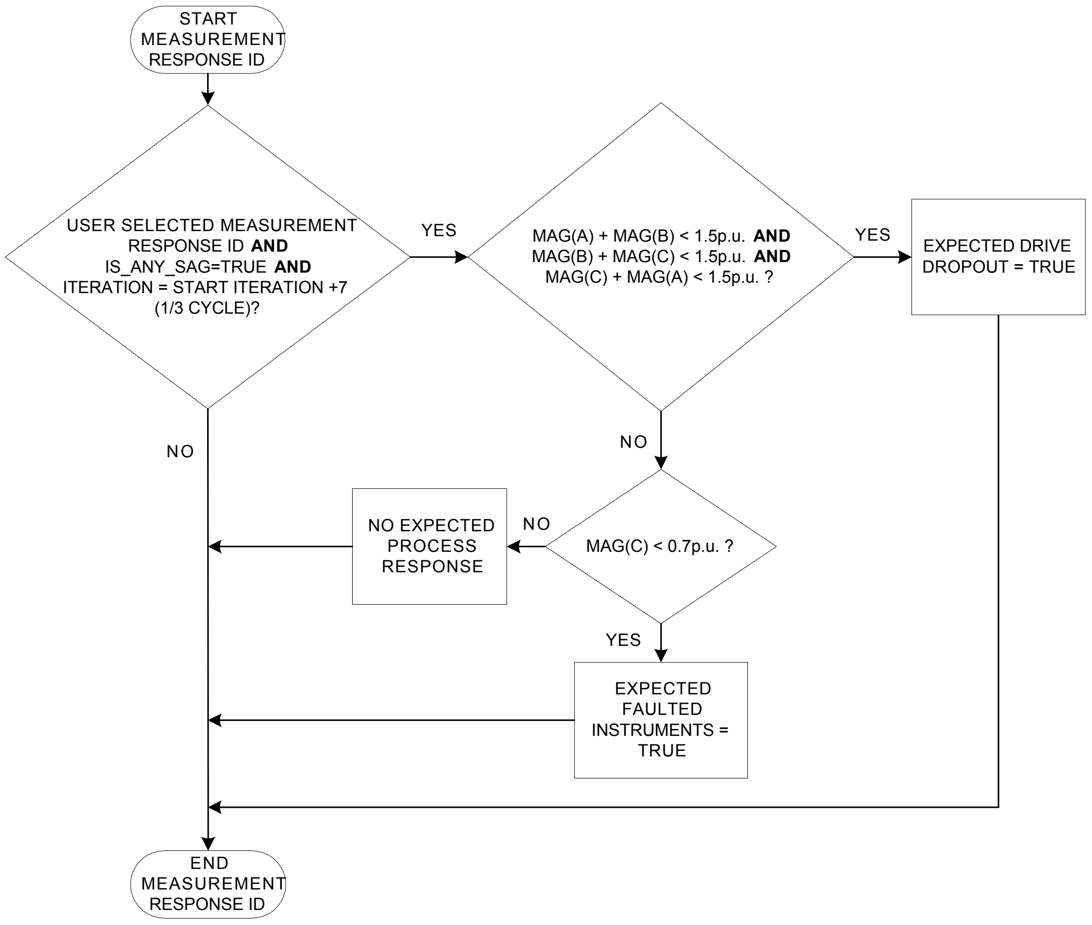

START MEASUREMENT RESPONSE ID
USER SELECTED MEASUREMENT RESPONSE ID AND IS_ANY_SAG=TRUE AND ITERATION = START ITERATION +7 (1/3 CYCLE)?
YES
MAG(A) + MAG(B) < 1.5p.u. AND MAG(B) + MAG(C) < 1.5p.u. AND MAG(C) + MAG(A) < 1.5p.u. ?
YES
EXPECTED DRIVE DROPOUT = TRUE
NO
NO
MAG(C) < 0.7p.u. ?
NO
NO EXPECTED PROCESS RESPONSE
YES
EXPECTED FAULTED INSTRUMENTS = TRUE
END MEASUREMENT RESPONSE ID

Figure 3.14 Flowchart for identification of expected process voltage sag response using threshold calculations.

Mitigation Routines

Once the expected response region of the incoming voltage sag is identified, then one of four ridethrough routines may be executed. Two routines aim at mitigating AC drive dropout faults and two address faulted instrumentation responses. When an incoming sag falls into the no disruption region, then the process is left undisturbed by the Ridethrough PC.

AC Drive Dropout Routines

Software Momentary Drive Coast

In the AC drive dropout region, the drives halt their output when their DC bus magnitude crosses its trip threshold. As shown in the unmitigated response study of the previous chapter, the rate of decent from the prefault magnitude to the trip level is significantly less when the motor is disconnected than when the motor is energized. The AC drive mitigation routines take advantage of this effect. Their aim is to switch the drives to a coast condition if a sag is expected to cause a DC bus trip, then re-energize them after the sag has completed. The flowchart of Figure 3.15 shows the procedure for the software momentary drive coast.

When the voltage sag has ended, the average prefault control values are substituted for the output control signals. Using the prefault values is a means of gradually returning the process to its normal state, instead of using feedback control which risks an extreme process variable overshoot upon recovery. After 1.5 seconds has elapsed on a postsag timer (a time sufficient enough to allow the process to return to normal operation), the standard feedback control conditions are restored. During the coast command, the PLC subroutine also calls for a PI control integrator hold. This allows feedback control to restore to prefault conditions without any signal drift occurring during the sag and process restart steps. The restarting algorithm flowchart is shown in Figure 3.16.

Naturally, this approach is expected to cause a process disturbance. In the case of the textile tension controller, the expected outcome is to experience a momentary

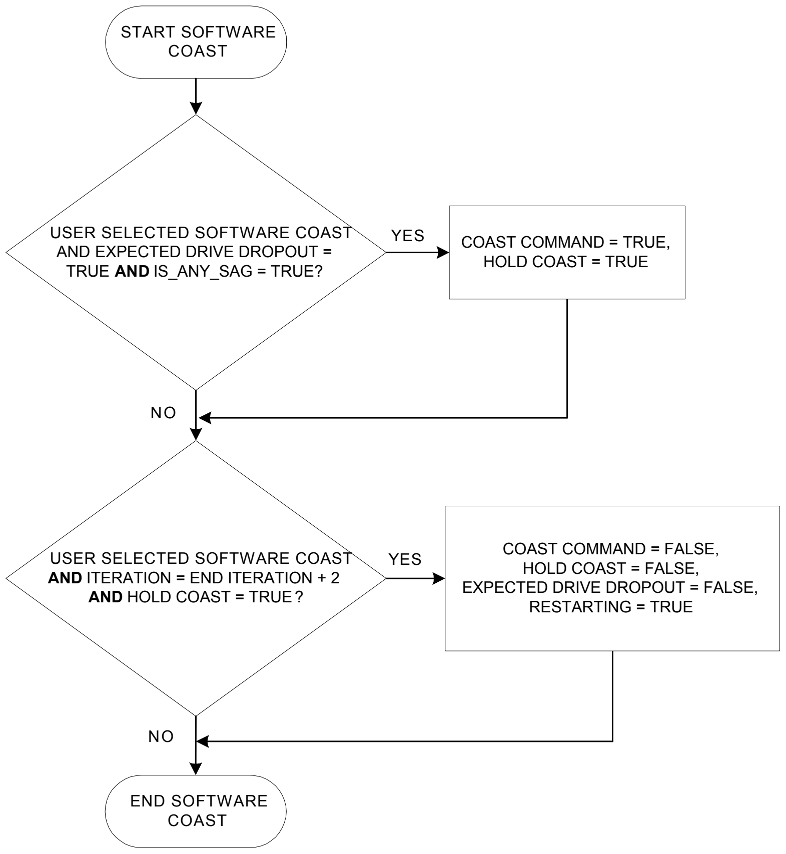

START SOFTWARE COAST
USER SELECTED SOFTWARE COAST AND EXPECTED DRIVE DROPOUT = TRUE **AND** IS_ANY_SAG = TRUE?
YES
COAST COMMAND = TRUE, HOLD COAST = TRUE
NO
USER SELECTED SOFTWARE COAST **AND** ITERATION = END ITERATION + 2 **AND** HOLD COAST = TRUE?
YES
COAST COMMAND = FALSE, HOLD COAST = FALSE, EXPECTED DRIVE DROPOUT = FALSE, RESTARTING = TRUE
NO
END SOFTWARE COAST

Figure 3.15 Flowchart of software momentary drive coast mitigation routine.

loss of tension along with changes in linespeed. Figure 3.17 illustrates the simulated process response to this mitigation method. The expected process disturbance, however, is a significant improvement over the unmitigated response. Without the ridethrough routine, the process shuts down, requires rethreading, and calls for a user issued restart.

The software momentary drive coast routine requires that the coast command reaches the drives before the DC bus crosses the trip threshold. Therefore, the effectiveness of the software coast is highly dependent on the system latency. Improvements in execution times will allow more time for the DC bus to decay at the de-energized rate before it crosses the trip threshold. The second AC drive dropout mitigation routine explores improving coast command performance in this manner.

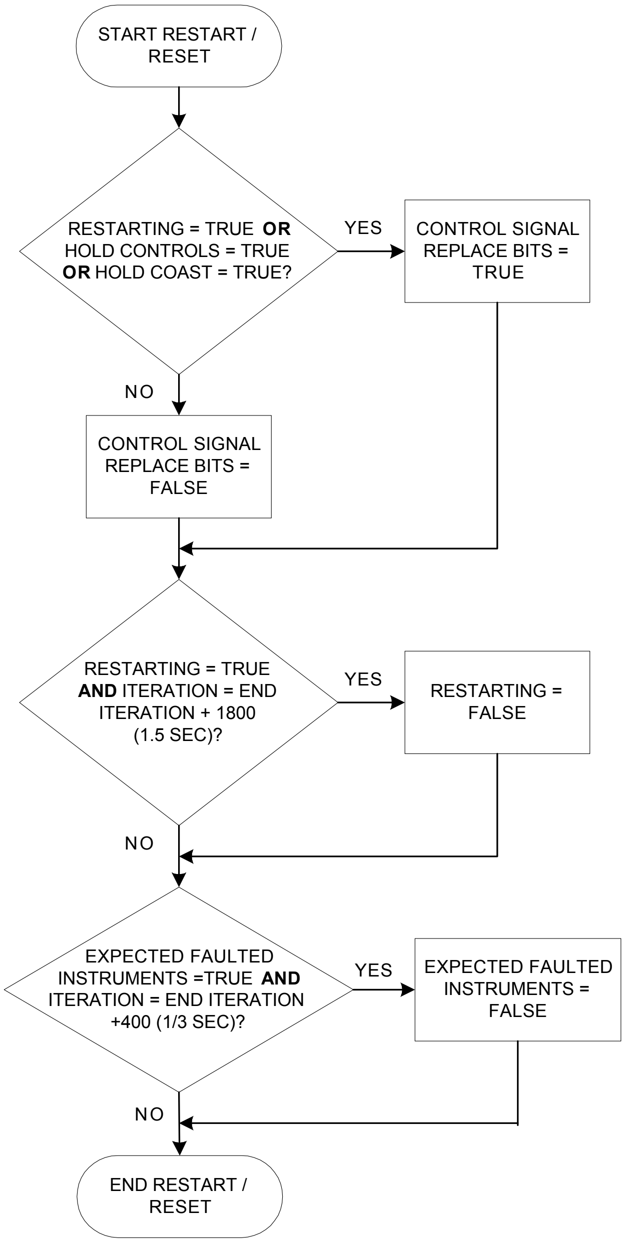

START RESTART / RESET
RESTARTING = TRUE OR HOLD CONTROLS = TRUE OR HOLD COAST = TRUE?
YES
CONTROL SIGNAL REPLACE BITS = TRUE
NO
CONTROL SIGNAL REPLACE BITS = FALSE
RESTARTING = TRUE AND ITERATION = END ITERATION + 1800 (1.5 SEC)?
YES
RESTARTING = FALSE
NO
EXPECTED FAULTED INSTRUMENTS =TRUE AND ITERATION = END ITERATION +400 (1/3 SEC)?
YES
EXPECTED FAULTED INSTRUMENTS = FALSE
NO
END RESTART / RESET

Figure 3.16 Flowchart of process restarting sequence following coast mitigation routine.

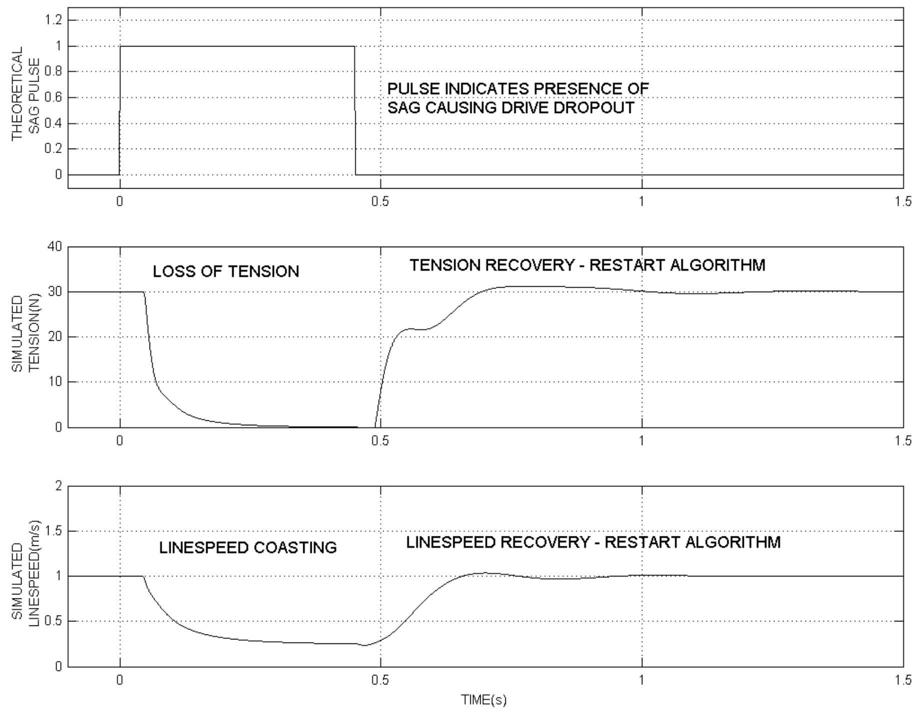


Figure 3.17 Theoretical process response to AC drive coast mitigation routines.

Hardware Assisted Momentary Drive Coast

The hardware momentary drive coast routine bypasses the Profibus network and PLC when performing an output halt to the payoff and takeup reel motors. This is accomplished by using an output channel available on the data acquisition card. When the halt command is given, the data acquisition card issues an output voltage which drives a bank of relays that physically disconnect the motors from the drives. In effect, this method performs the same operation as the software coast routine, but does not require a coast and restart command to the PLC. The output control signal replacement commands and the PI control integrators are controlled as they were in the software routine. The algorithm flowchart is shown in Figure 3.18 and the add-on circuit for this method is shown in Figure 3.19. This relay configuration was chosen for its ability to be driven entirely by the data acquisition card's analog output and available power supply. The data acquisition card sends a 0VDC or 5VDC signal from an analog output channel, which closes or opens an optical solid state relay. Operation of the solid state relay in turn operates a bank of 5VDC electromechanical relays that open and close the connection between the AC drives and the motors. Excitation voltage for the electromechanical relays is provided by a 5VDC, 1A supply from the data acquisition card. When the analog output is set to 0VDC, the drive-motor connection is closed. For the 5VDC analog output, the connections are opened, and the drive DC buses are prohibited from losing energy in the motors.

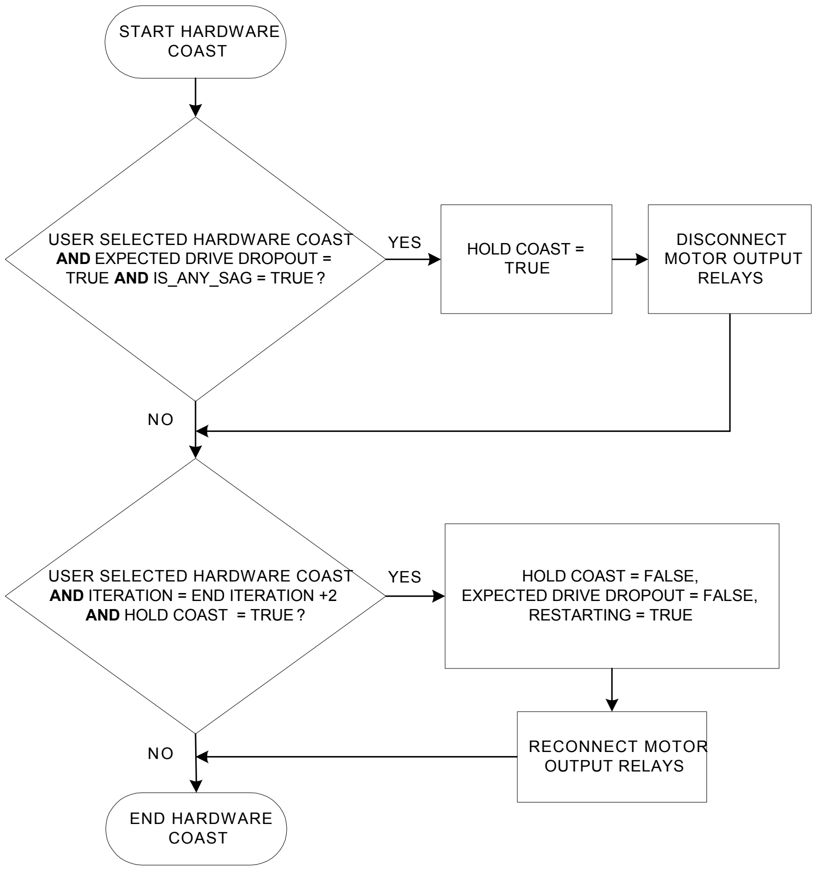

START HARDWARE COAST
USER SELECTED HARDWARE COAST AND EXPECTED DRIVE DROPOUT = TRUE AND IS_ANY_SAG = TRUE ?
YES
HOLD COAST = TRUE
DISCONNECT MOTOR OUTPUT RELAYS
NO
USER SELECTED HARDWARE COAST AND ITERATION = END ITERATION +2 AND HOLD COAST = TRUE ?
YES
HOLD COAST = FALSE, EXPECTED DRIVE DROPOUT = FALSE, RESTARTING = TRUE
RECONNECT MOTOR OUTPUT RELAYS
NO
END HARDWARE COAST

Figure 3.18 Flowchart of hardware assisted momentary drive coast mitigation routine.

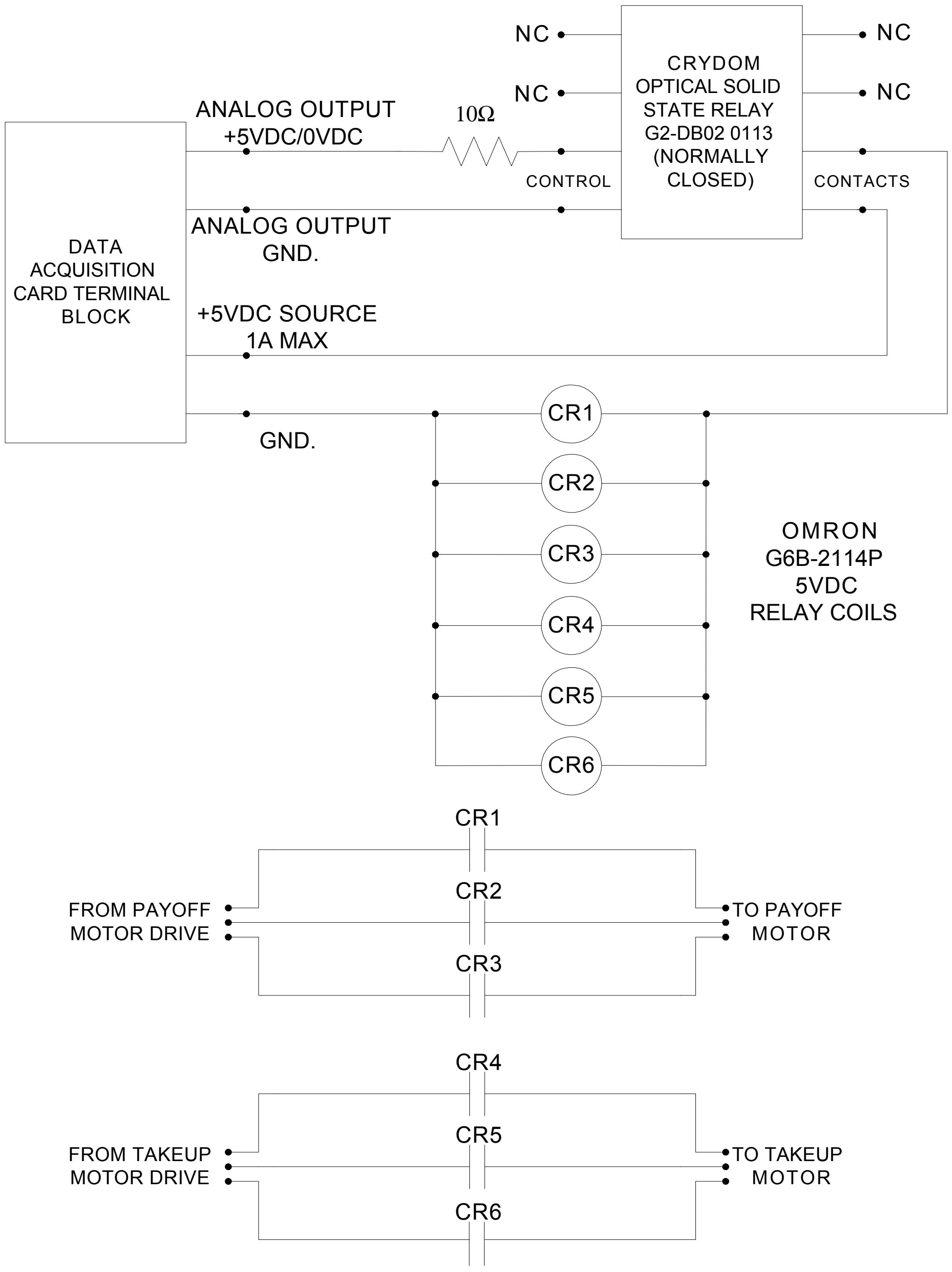


Figure 3.19 Add-on circuit for hardware assisted momentary drive coast mitigation routine.

The benefit of this method is that the coast command may reach the drives sooner than with the software routine, but at the cost of adding hardware. With a halt occurring sooner, the DC bus is given extra time to decay at the de-energized rate before the trip level is crossed. Improvements gained with this routine must therefore be weighed against the effort required to install the add-on circuit, along with the ability of the motor drives to permit an open-circuited motor output.

Faulted Instrument Mitigation Routines

Because responses in the faulted instrumentation region involve the propagation of erroneous system signals, the mitigation routines in this section concentrate on substitution of these signals with temporary values.

Control Signal Substitution

In this algorithm, if an incoming voltage sag is found to be in the faulted instrumentation region and the user selects the control signal substitution routine, then the Ridethrough PC selects the payoff and takeup control words to be replaced with their average prefault values. After a postsag delay of 1.5 seconds, the output signals are switched back to originate from the PLC controllers. During the execution of this routine, the PI integrators are held to allow for a smooth transition back to the original control conditions. This method is similar to the open loop method proposed in [68], but is only triggered for sags that fall into the faulted instrumentation region. The instruction flowchart for this mitigation method is shown in Figure 3.20.

As with the AC drive dropout routines, original conditions are restored a fixed amount of time after sag recovery. As shown in Figure 3.21, the expected response of this routine keeps the physical quantities of linespeed and tension constant, assuming steady state prefault operating conditions. This method does not take into account the variations in ridethrough and recovery times that are present with different sensors, for different sags, and for varying physical inputs. With increased specification of individual sensor responses, signal replacement may be a more customized affair, and allow the PI control loops to operate through a sag in this region.

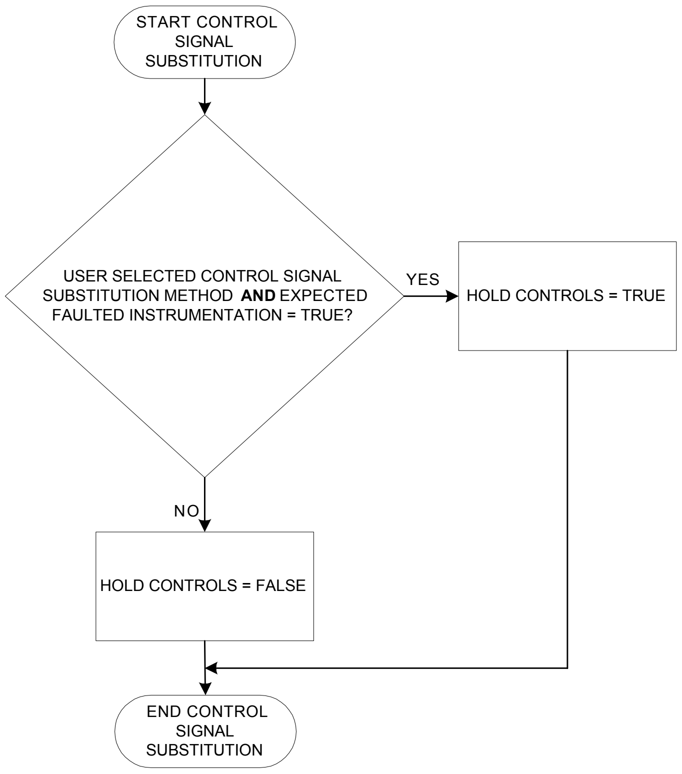

START CONTROL SIGNAL SUBSTITUTION
USER SELECTED CONTROL SIGNAL SUBSTITUTION METHOD **AND** EXPECTED FAULTED INSTRUMENTATION = TRUE?
YES
HOLD CONTROLS = TRUE
NO
HOLD CONTROLS = FALSE
END CONTROL SIGNAL SUBSTITUTION

Figure 3.20 Flowchart of control signal substitution mitigation routine.

Selective Instrument Signal Substitution

In the selective instrument signal substitution routine, the average prefault values of each sensor are determined and used along with the incoming sag magnitude to access an array of ridethrough times for each instrument. The values in these arrays are determined from previously stored sag testing results as recorded in Appendix E. Array indices are generated using Equation 3.9 and the equation

$$w_y = \left\lceil \frac{y_{avg} - y_{min}}{\rho_i} \right\rceil, \tag{3.10}$$

where w_y is the physical input level array index for accessing the ridethrough time, y_{avg} is the average prefault value of the physical input, y_{min} is the minimum physical value for the process, and ρ_i the step size in units of the physical input that instrument i measures (m/s, N). With a physical input and sag magnitude known, the accessed ridethrough time is used as a delay time for the instrument signal to be replaced with

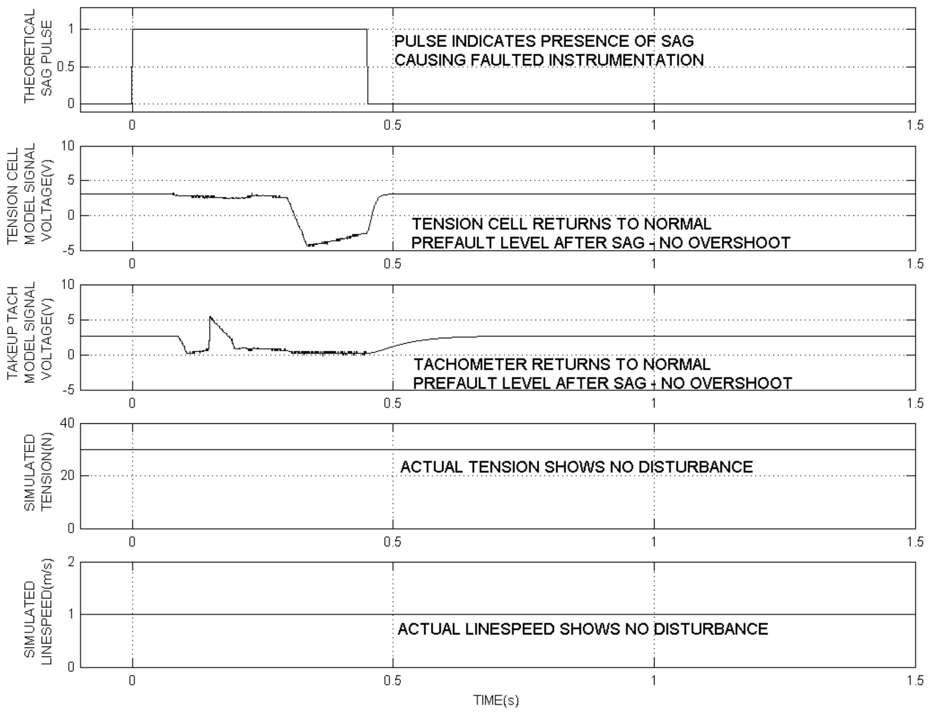


Figure 3.21 Theoretical process response to control signal substitution and selective signal substitution routines.

the average prefault value. In cases where the instrument rode through an event, an arbitrarily large delay time is stored in the array, which effectively prohibits the instrument signal from being replaced. Maximum recovery times are also accessed for each instrument, and after the sag is completed, the instrument signal is not routed back to the control algorithm until the postsag timer has elapsed the instrument maximum recovery time. A flowchart for this algorithm is shown in Figure 3.22.

The significant improvement this mitigation method makes over control signal substitution is that the instrument signals are allowed to pass into the control algorithm if their value is expected to be undisturbed. This allows for closed loop control while the instruments are in the ridethrough segment of their sag response. For sags of a shorter duration than the ridethrough times, the control algorithm is unaffected and the inherent instrument ridethrough is used to its full capacity. Also eliminated

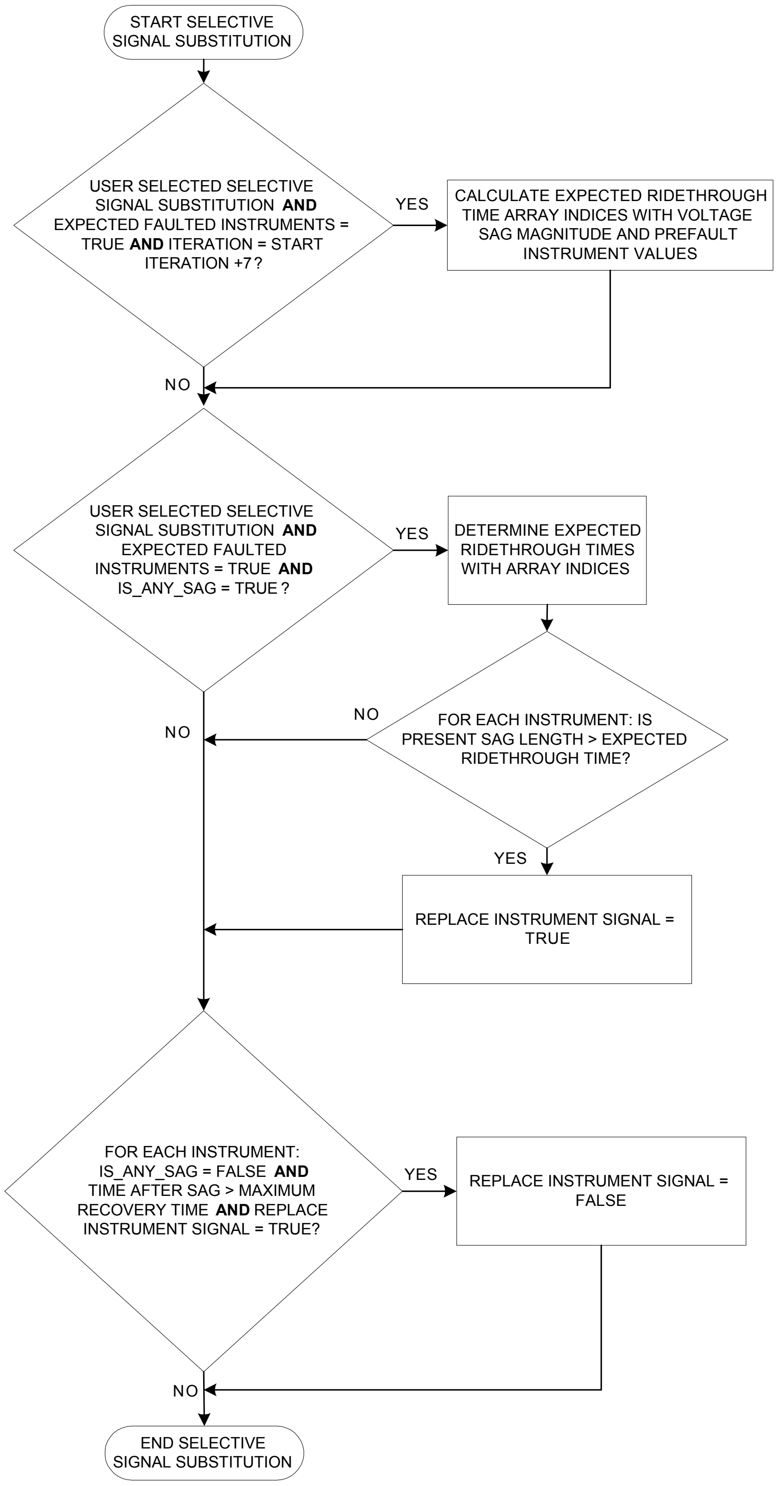

START SELECTIVE SIGNAL SUBSTITUTION
USER SELECTED SELECTIVE SIGNAL SUBSTITUTION AND EXPECTED FAULTED INSTRUMENTS = TRUE AND ITERATION = START ITERATION +7 ?
YES
CALCULATE EXPECTED RIDETHROUGH TIME ARRAY INDICES WITH VOLTAGE SAG MAGNITUDE AND PREFAULT INSTRUMENT VALUES
NO
USER SELECTED SELECTIVE SIGNAL SUBSTITUTION AND EXPECTED FAULTED INSTRUMENTS = TRUE AND IS_ANY_SAG = TRUE ?
YES
DETERMINE EXPECTED RIDETHROUGH TIMES WITH ARRAY INDICES
NO
FOR EACH INSTRUMENT: IS PRESENT SAG LENGTH > EXPECTED RIDETHROUGH TIME?
NO
YES
REPLACE INSTRUMENT SIGNAL = TRUE
FOR EACH INSTRUMENT: IS_ANY_SAG = FALSE AND TIME AFTER SAG > MAXIMUM RECOVERY TIME AND REPLACE INSTRUMENT SIGNAL = TRUE?
YES
REPLACE INSTRUMENT SIGNAL = FALSE
NO
END SELECTIVE SIGNAL SUBSTITUTION

Figure 3.22 Flowchart of selective instrument signal substitution mitigation routine.

is the need to hold the PI control integrals for a smooth transition following recovery. The theoretical response of this method as it is applied to the textile tension control stand is the same as the theoretical control signal substitution response as shown in Figure 3.21.

Summary

A software ridethrough coordination and mitigation program was developed to interface with a three phase power supply and the Profibus network used on the experimental textile tension control stand. Interfacing hardware consists of voltage isolators and a National Instruments data acquisition card for the power supply, and a Woodhead/SST Profibus interface card for the automation network. Additions to the existing PLC ladder program were minimal, consisting of an add-on subroutine to manage data transfer and a series of signal switching commands in the main ladder program. Implementation and execution of the main ridethrough program occurs in the LabView graphical programming platform.

Voltage sag detection is triggered by RMS derivative threshold crossings, with a theoretical detect time of less than one-quarter cycle. Sags are measured in an additional quarter cycle, and are either associated with historical process data corresponding to the incoming sag or applied to logic functions that determine the expected process response. The program monitors selected process data from the automation network, and in the event of a sag stores signals and status data for future reference. Voltage sag measurements are also taken and included in the event history.

Mitigation routines are executed based on preselected user input and the expected disruption region of the incoming sag. Sags that are categorized in the AC drive dropout region are mitigated by either a command driven coast routine or a contactor driven coast routine. Command driven coasting is entirely software based, but is potentially limited by data transfer rates. Contactor coasting can increase response time by functioning outside the PLC and data network, but requires the addition of motor contactors and switching relays.

Sags that are expected to cause instrumentation faults are addressed in one of two ways. In the control signal substitution method, the motor drive control signals are immediately replaced with their prefault values and the PI control integrators are held. After the sag and a fixed waiting period have expired, the PI control outputs are redirected to their original locations. In the selective substitution method, timers are set to the ridethrough times for a given instrument. This timer setting is determined by the present voltage sag magnitude and instrument physical input. Expiration of a ridethrough timer triggers the instrument signal to be replaced with its average prefault value. This is an improvement over the control signal substitution method because it takes advantage of each instrument's inherent ridethrough capabilities. Detailed sag testing is, however, required to determine the appropriate delay times. Instrument recovery times are also referenced to determine the optimal times for switching the original signal back into the control loop.

Additional flowcharts for the Ridethrough PC programming is given in Appendix I, and the LabView code is provided in Appendix J.

CHAPTER 4
EXPERIMENTAL RESULTS

The Ridethrough PC software as described in the previous chapter was applied to the experimental textile tension control stand. This chapter provides an assessment of software mitigation algorithm effectiveness, along with analysis of the interfacing, sag detection, and event analysis/association subsystems.

<u>Execution Times</u>

The latency, or stimulus-to-response timing of the Ridethrough PC software may be assessed in terms of component execution times. These components are detailed in Figure 4.1, which shows the order of operational delays for each mitigation routine. The measured times are listed in Table 4.1.

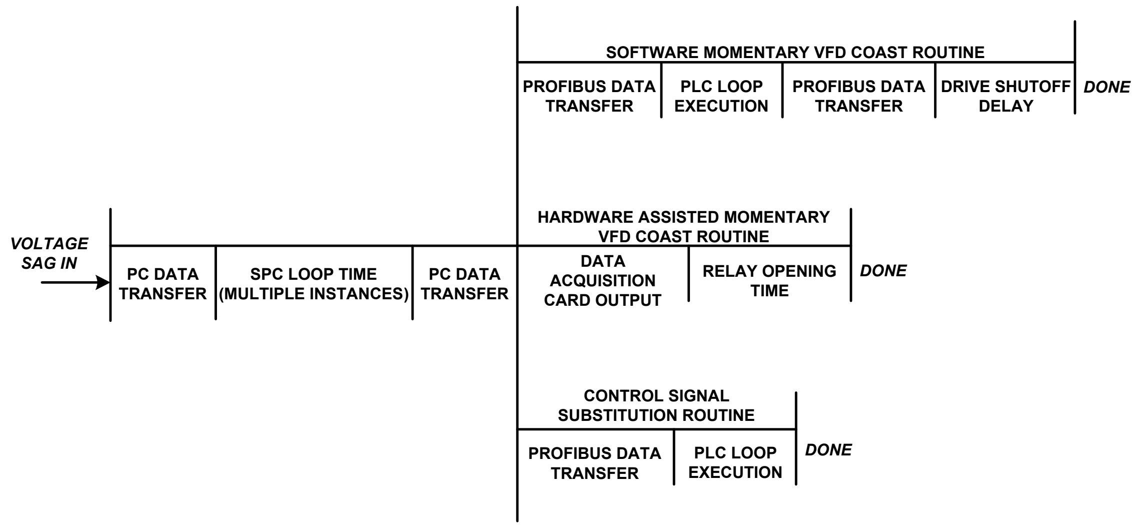


Figure 4.1 Order of operations for analysis of execution times.

Table 4.1 Individual operation execution times.

OPERATION	EXECUTION TIME
PC data transfer	$6\mu s$
Ridethrough PC loop time	$0.833ms$
Profibus data transfer	$5.912ms$
PLC loop execution	$8.7ms$
Drive shutoff delay	$25ms$
Data acquisition card output time	$204\mu s$
Drive output relay opening time	$3.61ms$

Timing Measurements

Measurement of the values in Table 4.1 were performed by the following methods:

1. PC data transfer. The delay time was determined by executing a program that sends an incremental variable to the Profibus network. Examination of the Profibus data stream for data value transitions yielded increment numbers and their corresponding time stamps. For each data point, the increment number and time increment was recorded. The average length per increment was determined to be $6\mu s$ and is attributed to PC data transfer delays from the data acquisition card or to the Profibus interface card.

2. Ridethrough PC loop time. This is defined by the sampling period for the voltage signal inputs. Five data points are sampled for each loop increment at a sampling frequency of 6kHz, yielding a loop time of $0.833ms$.

3. Profibus data transfer. Loop cycle times were measured using Scope Profibus software, and were determined to vary from an experimental low of $5.418ms$ to a high of $7.722ms$, with an average of $5.912ms$.

4. PLC loop execution. Determination of PLC execution times were made by accessing the PLC through Siemens Step 7 programming and diagnostic software. Scan cycle times varied between $6ms$ and $11ms$, with an average of $8.7ms$.

5. Drive output shutoff delay. The time required for the drive to respond to a direct (non-network issued) shutoff command was determined to lie between $18ms$ and $31ms$ with an average response time of $25ms$.

6. Data acquisition card output time. This time is determined by a executing loop containing an analog output command only. The average time is calculated using the number of loop iterations over a fixed period of time, which yielded a value of $210\mu s$. The PC data transfer delay is subtracted from this value, which results in a delay time of approximately $204\mu s$

7. Drive output relay opening time. Experimentally this delay time varied from $3.48ms$ to $3.90ms$ with an average of $3.61ms$.

Combined Routine Execution

The collective propagation and execution delays itemized in Table 4.1 may be used to estimate the execution times for the software mitigation algorithms. These estimated times are listed in Table 4.2. For each value, 12 Ridethrough PC program cycles are included to account for sag detect and measurement times. Profibus network delays and PLC loop execution times are each multiplied by 1.5 times their measured average, which accounts for the full cycle where a response is computed plus an average of one-half cycle when the stimulus arrives but is not recognized until the beginning of the next full computation cycle. The selective signal substitution execution times are not listed due to the variation caused by built-in time delays, though the time-invariant components are identical to the control signal substitution method.

Comparison of the estimated execution times in Table 4.2 with their experimental counterparts in Table 4.3 reveals several differences. In the software momentary drive coast algorithm the estimation falls above the experimental range, but the hardware assisted drive coast algorithm is predicted to be faster than the experimental values. This may be attributed to unmeasured variation in the Ridethrough PC data transfer times resulting from increased demand of system resources that the complete mitigation program has over the experimental timing measurement program. Such variations serve as an indicator that a computationally inexpensive software platform is a key component of a software-based mitigation system. When comparing the coast mitigation algorithms to one another, the improvement in execution time that the hardware assisted method offers over the software-only method emphasizes the dependence that algorithm effectiveness has on network communication throughput and execution timing. The measurements made for the control signal substitution method are made without the Profibus data transfer and PLC loop execution components, but include the data acquisition card output component. For this algorithm, if the experimental value in Table 4.3 is compensated with the average measured component values to account for the measurement method, the execution time range lies between $33.4ms$ and $35.4ms$, which is a closer match to the predicted value, but remains slightly longer.

Table 4.2 Mitigation routine predicted execution times.

MITIGATION ROUTINE	**APPROX. EXECUTION TIME**
Software momentary drive coast	$65.9ms$
Hardware assisted momentary drive coast	$13.9ms$
Control signal substitution	$32.0ms$

Table 4.3 Mitigation routine experimental execution times.

MITIGATION ROUTINE	**ACTUAL EXECUTION TIME**
Software momentary drive coast	$56ms \leq t \leq 64ms$
Hardware assisted momentary drive coast	$20ms \leq t \leq 45ms$
Control signal substitution	$19ms \leq t \leq 21ms$

Performance of Sag Detection, Sag Measurement, and Event Association

Sag Detection

Experimental performance of the voltage sag detection method is shown in Figures 4.2-4.5 and summarized in Table 4.4. In several cases, the actual detect times are one to two execution cycles of the Ridethrough PC program (multiples of $0.833ms$) longer than is predicted by MATLAB simulations (Figures 3.5-3.8). When comparing the experimental and simulated cases, the RMS voltage derivative deviates from its prefault value more rapidly in the simulated case. This result is caused by slight differences in nominal voltage magnitude and distortions from a true sine wave in the experimentally captured input waveform. The distortion's effects are evident in the experimental prefault RMS voltage derivative waveforms, where a slight ripple can be observed. Differences between actual and predicted response times are also caused by the time at which the sag is initiated relative to the execution cycles of the Ridethrough PC. In the experimental cases, the sag is initiated at a random time relative to the Ridethrough PC execution. This asynchronous behavior leads to RMS voltage derivative threshold crossings that are detected at slightly different times, which can lead to a single Ridethrough PC execution cycle difference of $0.833ms$.

Cases of 89% magnitude, 90° point in wave (Figure 4.6) and 89% magnitude, 0° point in wave voltage sags were also observed. In both examples, no detection signal is generated, which matches the simulated prediction (Figure 3.9 shows the 89%, 0° simulation). This result can be alleviated using RMS voltage derivative detection thresholds lower than the settings of $\pm 2kV/s$. This change must take into consideration the balance between fast sag measurement and low susceptibility to line noise and voltage transients. If these tolerance bands were narrowed, the resulting increase in sag detection performance would come at the cost of increased sensitivity and false sag detection signals.

Experimentally, the voltage sag detection method proves to be a satisfactory solution for the system to which it is applied. It is computationally inexpensive and will reliably detect events delivered by the voltage sag generator that supplies the textile tension control process. Detections occur within $5.0ms$ of voltage sag inception, which provides adequate time for measurement and expected response identification to occur before the process is disrupted.

Table 4.4 Actual and theoretical voltage sag detection times.

SAG MAG. / P.O.W.	**THEORETICAL TIMES**	**ACTUAL TIMES**
20% / 0°	2.50ms on, 1.67ms off	3.33ms on, 2.50ms off
20% / 90°	0.83ms on, 0.83ms off	0.83ms on, 0.83ms off
80% / 0°	3.33ms on, 3.33ms off	5.00ms on, 4.17ms off
80% / 90°	0.83ms on, 0.83ms off	2.50ms on, 1.67ms off
89% / 0°	No detections	No detections
89% / 90°	No detections	No detections

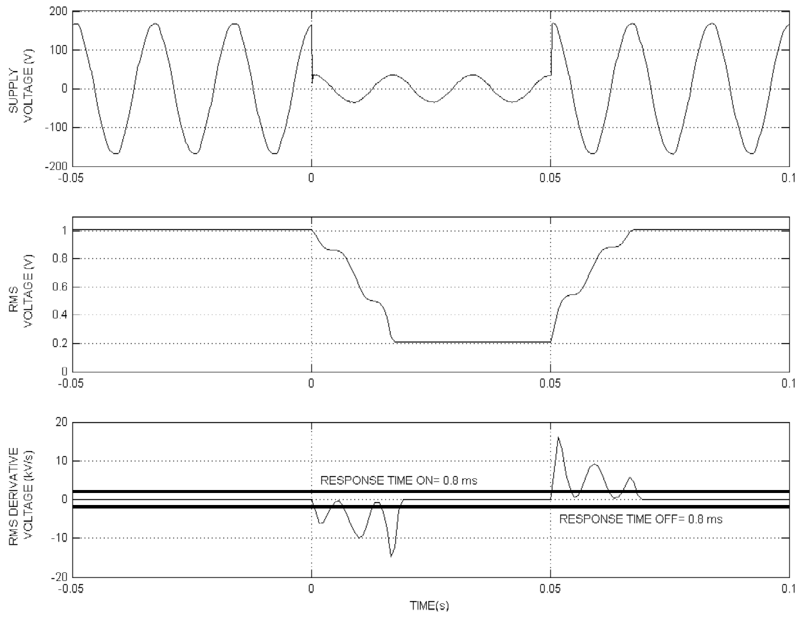


Figure 4.2 Experimental detection response to 20%, 90° point on wave voltage sag.

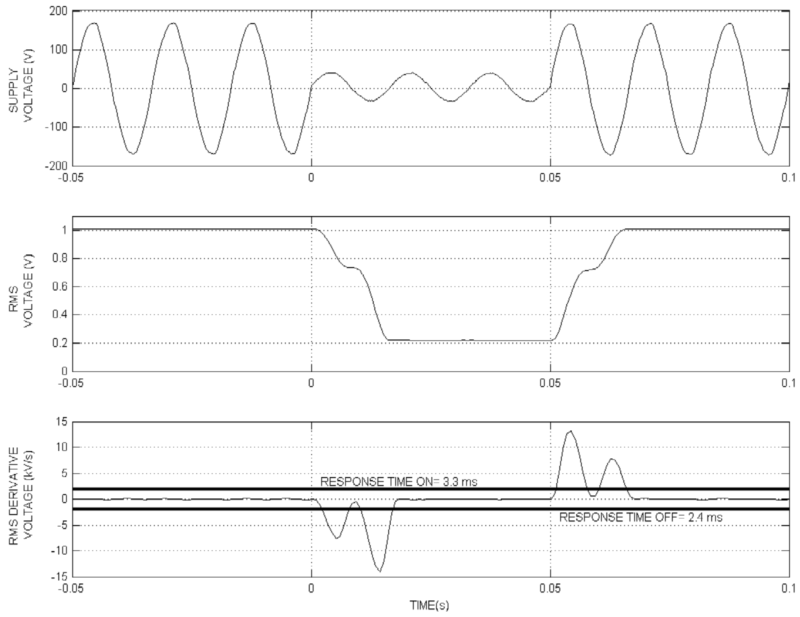


Figure 4.3 Experimental detection response to 20%, 0° point on wave voltage sag.

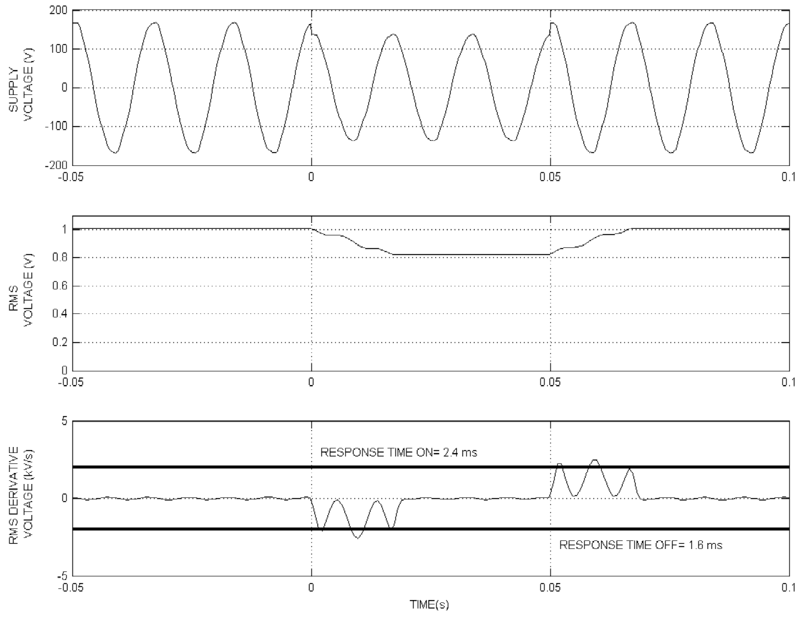


Figure 4.4 Experimental detection response to 80%, 90° point on wave voltage sag.

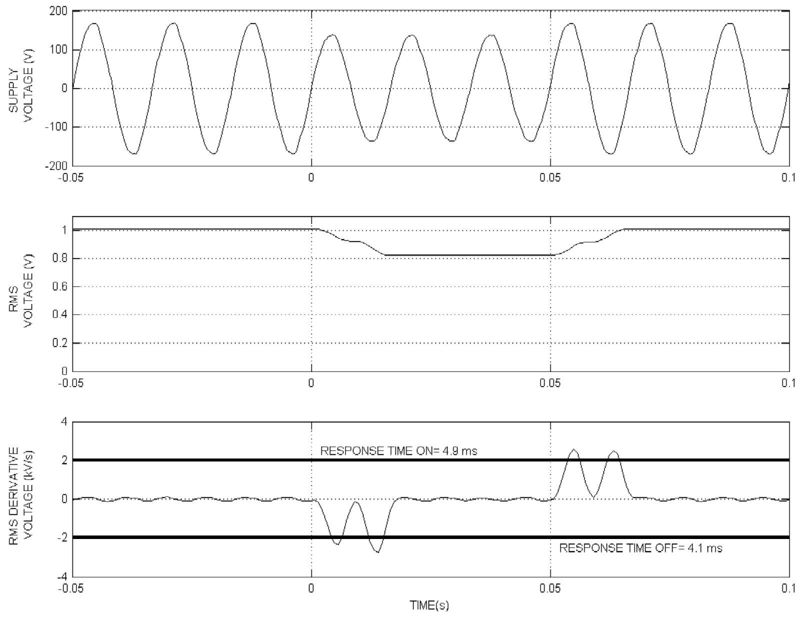


Figure 4.5 Experimental detection response to 80%, 0° point on wave voltage sag.

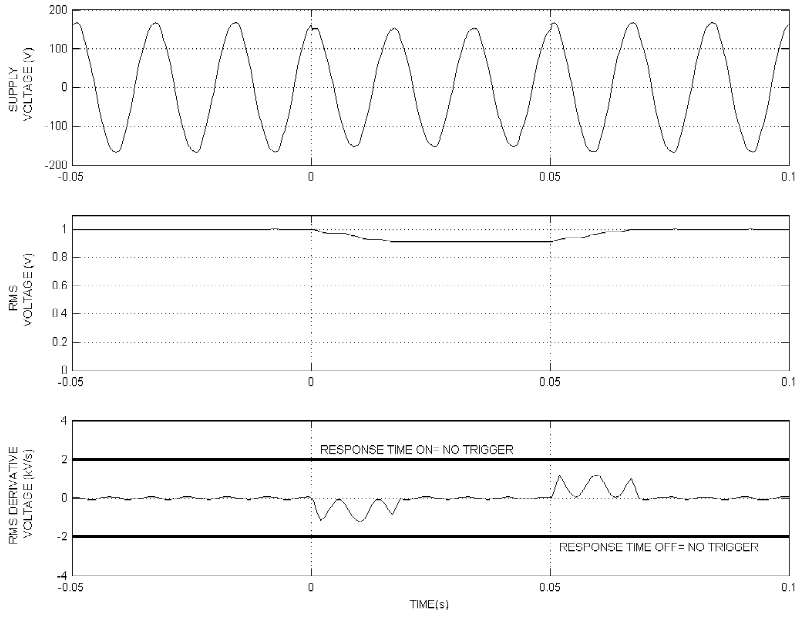


Figure 4.6 Experimental detection response to 89%, 90° point on wave voltage sag.

Sag Measurement

Graphs of the experimental voltage sag measurement output are shown in Figures 4.7 and 4.8. In the example for a 40%, 0° point in wave voltage sag, the resulting magnitude measurement possesses the expected time delay of one-quarter cycle before the measurement settles to a reliable value. The experimental case verifies the simulated case (Figure 3.11), where the sole difference between experimental and theoretical cases is a slight deviation in the experimental sag magnitude measurements at the points of inception and recovery. Despite this difference, the measurement is reliable for the application of utilizing the sag magnitude calculation after allowing the one-quarter cycle delay to elapse.

For the 40%, 90 ° point in wave case shown experimentally in Figure 4.8 and theoretically in Figure 3.12, both cases show the resulting magnitude measurement spike caused by abrupt voltage transitions applied to the calculation of Equations 3.7 and 3.8. This underscores the criticality of allowing a one-quarter cycle delay to elapse before applying the measurement towards identifying the voltage sag's likely process effects. Naturally, line noise present in the system will contribute towards errors in measurement of the voltage peak using this method. This issue may be alleviated with hardware or software filters, the application of which should involve specific consideration of the filter performance versus the delays that they introduce. Nevertheless, this measurement method proves to be a simple and effective means of determining a voltage sag's magnitude shortly after it is detected.

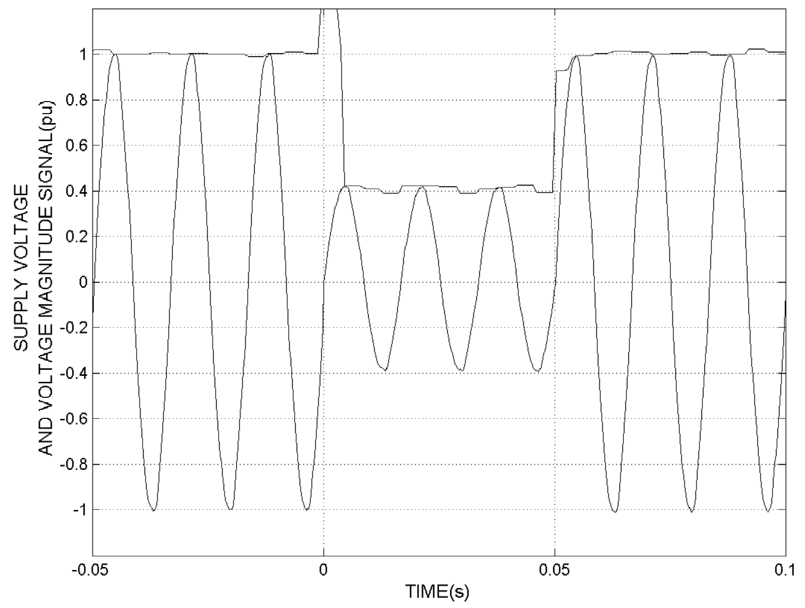


Figure 4.7 Experimental magnitude measurement response to 40%, 0° point on wave voltage sag.

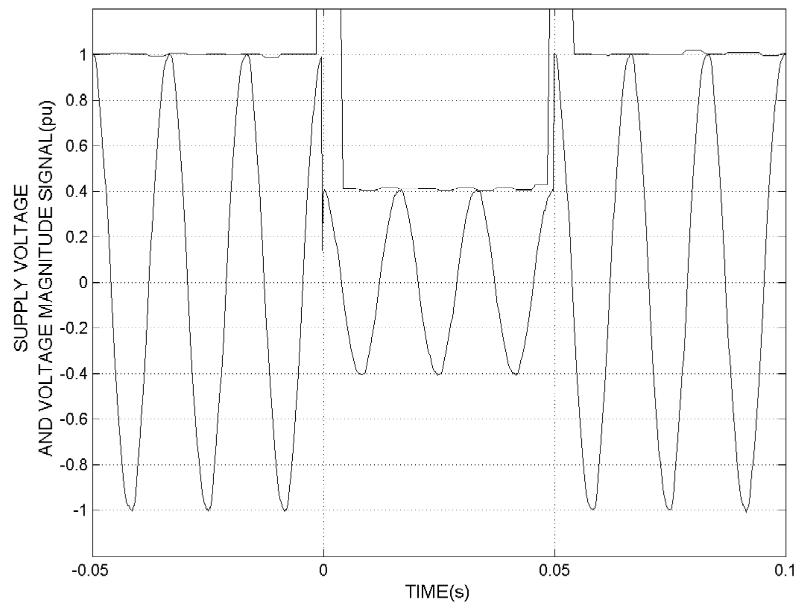


Figure 4.8 Experimental magnitude measurement response to 40%, 90° point on wave voltage sag.

Event Association

The event association methods are executed seven Ridethrough PC cycles after a detect signal is given, which allows sufficient time for the peak voltage measurement to reach its final value. The method of identifying the expected response region is preselected by user input. For the process history identification method, the history is read into memory before the program begins executing, which avoids the delays associated with opening and accessing a data file while the sag is in progress.

The method for process response identification using a recorded process history functions properly; for the magnitude and phasing combinations delivered by the voltage sag generator (in 20% increments), the detection and measurement routines' application towards identifying a recorded event using Equation 3.9 reliably calculate array indices to reference a detailed process voltage sag response history. Similarly, identification of the expected process response using Boolean response region estimations as described in Equations 2.31 and 2.32 performs properly and matches the expected responses predicted using the process history. Under both of these methods, expected response determination occurs in less than one cycle of 60Hz incoming voltage, which is significantly less than the delay time between voltage sag inception and any unmitigated process response.

With a complete process response history using voltage sag magnitude increments of 20%, the history file size is 23MB. For the identification of an expected drive dropout, much of this data is unused. Only the state of the takeup and payoff drives is scanned to determine membership in the drive dropout region and as such may be reduced if only an expected drive dropout is to be determined. However, for identification of expected faulted instrumentation, the recorded process instrumentation and control signals are also scanned to determine out of tolerance conditions and therefore require more detailed process history recording. In contrast, the memory requirement of a process history is eliminated when using Boolean response boundary estimation, but requires a clear definition of these boundaries to be effective.

Mitigation Algorithms

Each software mitigation algorithm was implemented and applied to the textile tension control stand. This section describes the performance of each routine under various test conditions.

Momentary Drive Coast Routines

The drive dropout routines aim at mitigating the dropout response as shown in Figure 4.9. This is accomplished by entering a coast state in order to prevent or delay the drive DC bus from crossing the trip level. Tests performed for these routines used equal magnitude voltage sags on all three supply phases.

Software Momentary Drive Coast

Successful execution of the software momentary drive coast algorithm is shown in Figure 4.10. Process setpoints are maintained at 1.0m/s line speed and 30N web tension. In this case, the drive DC bus voltages are held above the trip level of 200V by temporarily stopping the drive output starting approximately $56ms$ after sag inception. The DC bus decays at a slower rate, which provides additional time for sag recovery to occur. After the sag has ended, both drives resume their output, but are set to their prefault average setpoints (Figure 4.11). Product spillage and manual restart are avoided, and the process line speed and tension return to their prefault values and ultimately transition back to standard control at $t = 1.95s$. The theoretical model predicts this behavior generally, but suggests a faster recovery than in the experimental case. This difference is caused by the slack condition present in the mechanical system that is not included in the theoretical model.

Figures 4.12 and 4.13 show the behavior of the software momentary drive coast algorithm when recovery is not successful. In this case, the coast commands are executed later than in the successful case because of variation in voltage sag detection times and command execution latency. The DC buses do not remain above the dropout level of 200VDC for the entire duration of the interruption. Because of the drive undervoltage faults, the restart and restore commands are executed with no effect. The unsuccessful execution of this algorithm ultimately mirrors the unmitigated response and reveals the limitations of this method for sags of longer duration. This scenario underscores the need for fast execution, but reveals that the unsuccessful case is the inevitable result as the voltage sag duration approaches infinity, regardless of the mitigating effort applied.

Altogether this mitigation strategy meets expectations – the process sees a disturbance similar to a shutdown, but the shutdown is one that is intentionally caused so that it can be brought back online in a controlled fashion. Process variables are not continuously maintained, yet are restored without manual operator intervention. Success of the software momentary drive coast routine requires that the coast command reaches the drives before the DC bus crosses the trip threshold and that the voltage sag or interruption is of a reasonably brief duration such that the DC bus voltages do not cross over their trip thresholds, even while decaying at a slower rate.

Despite its experimental success, the response times of this mitigation routine do leave room for improvement. The effectiveness of this method is highly dependent on Profibus network latency and PLC execution time. Faster coast executions will cause DC buses to begin the de-energized rate of decay at a higher level and therefore allow more time before the trip thresholds are crossed. If this method were implemented such that its performance were less reliant on the propagation of coast commands over the Profibus network from Ridethrough PC to PLC and then back to the VFDs, then the momentary coasting routine would be tolerant to voltage sags of longer duration and hence reduce the occurrences of unsuccessful execution. This reliance on network and PLC timing gives rise to the hardware assisted momentary drive coast mitigation routine.

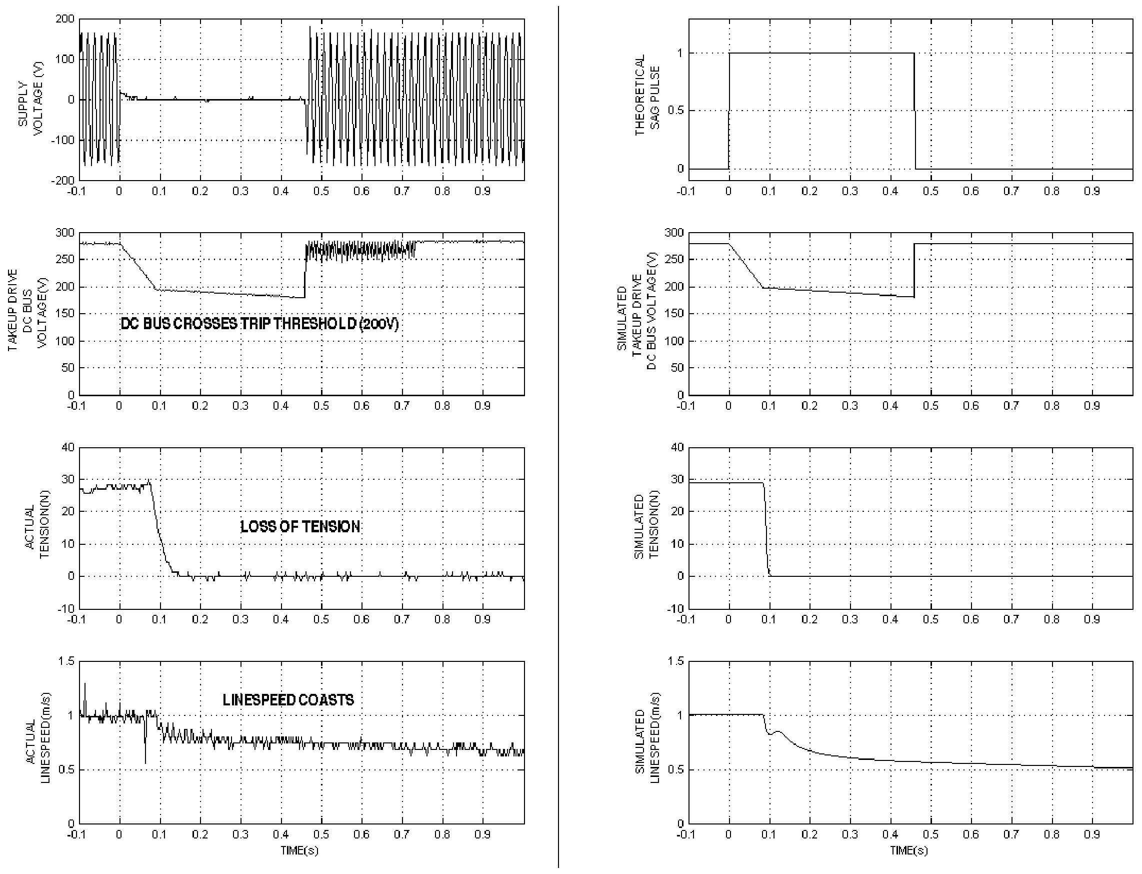


Figure 4.9 Experimental and simulated unmitigated process response resulting in drive dropout ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 3ϕ voltage interruption).

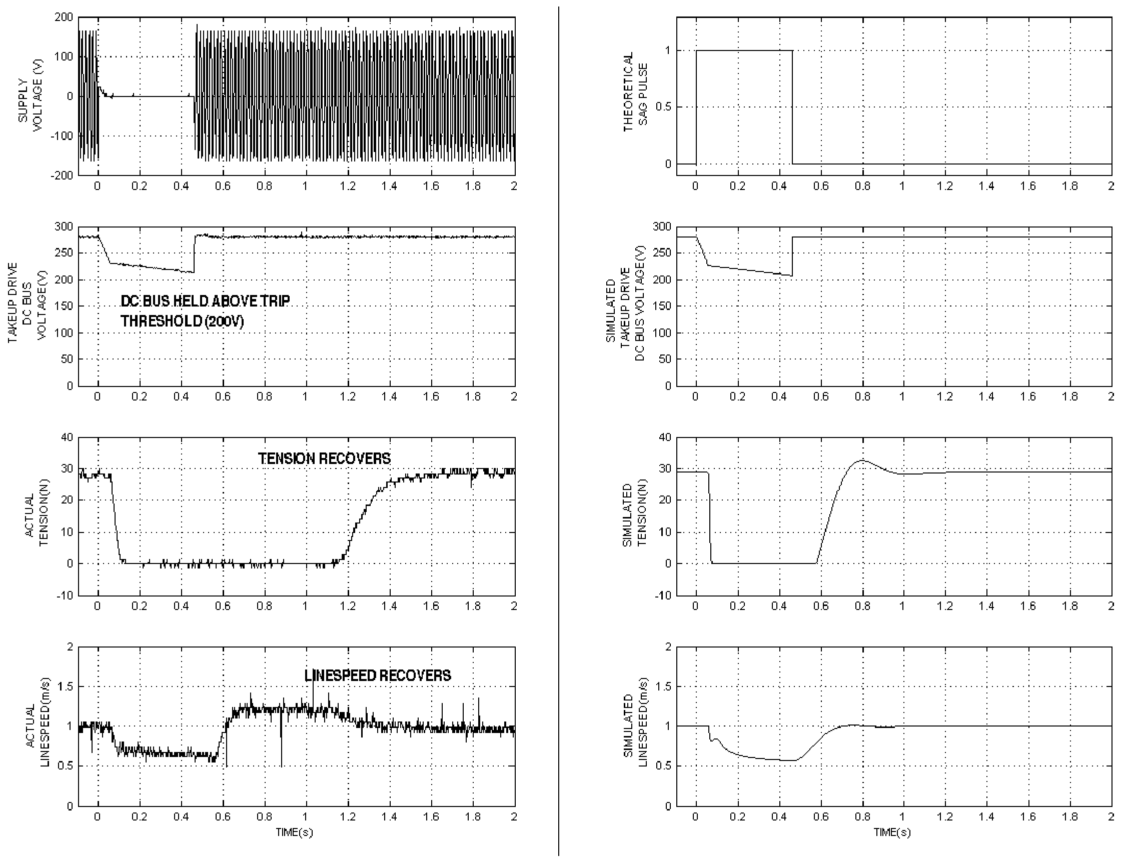


Figure 4.10 Experimental and simulated process response using software momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 3ϕ voltage interruption).

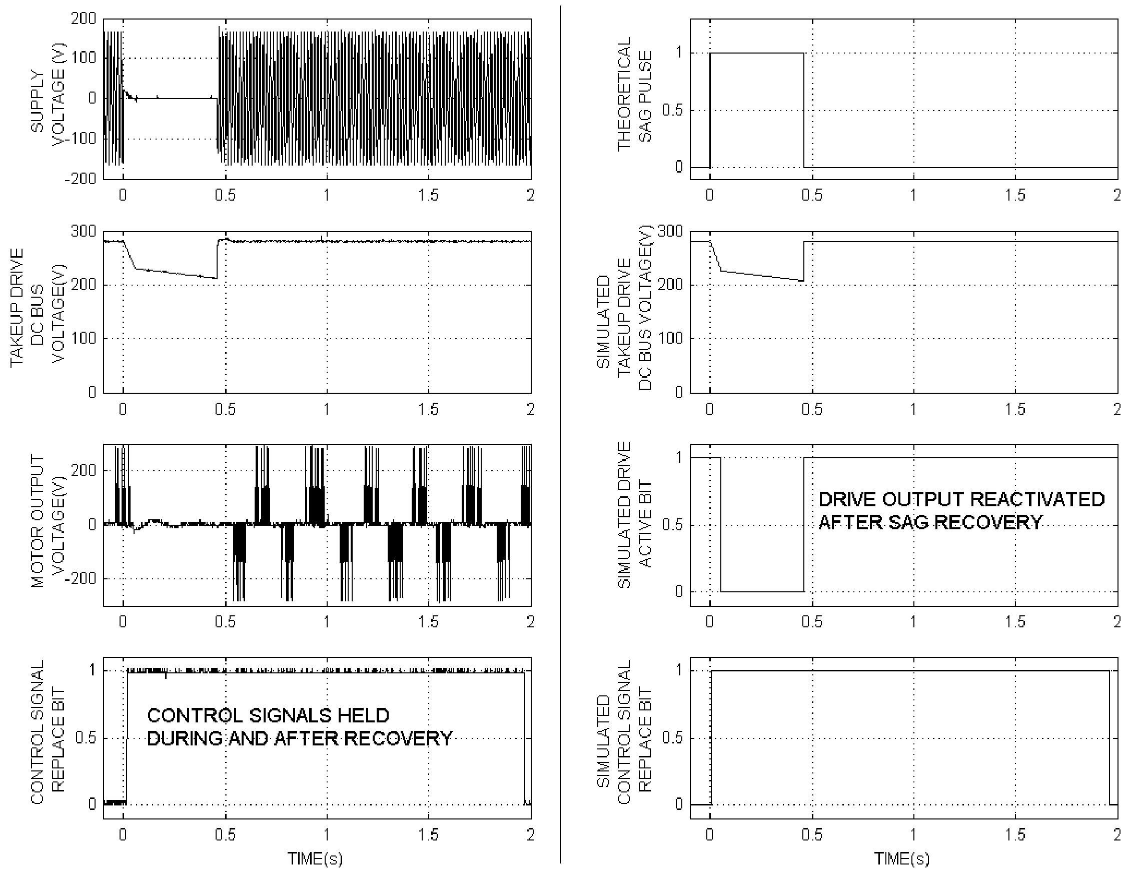


Figure 4.11 Experimental and simulated DC bus magnitude, motor output, and control signal replace command during software momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 3ϕ voltage interruption).

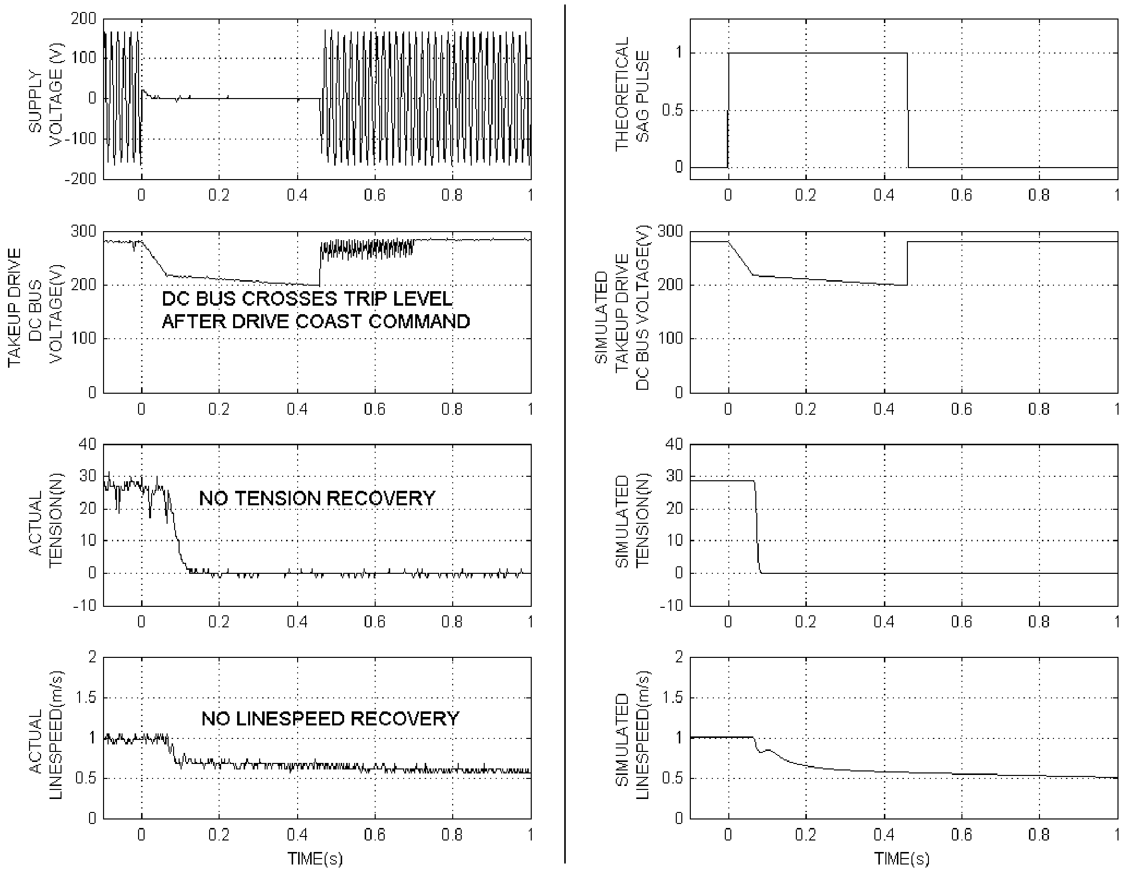


Figure 4.12 Experimental and simulated process response demonstrating unsuccessful process recovery during application of software momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 3ϕ voltage interruption).

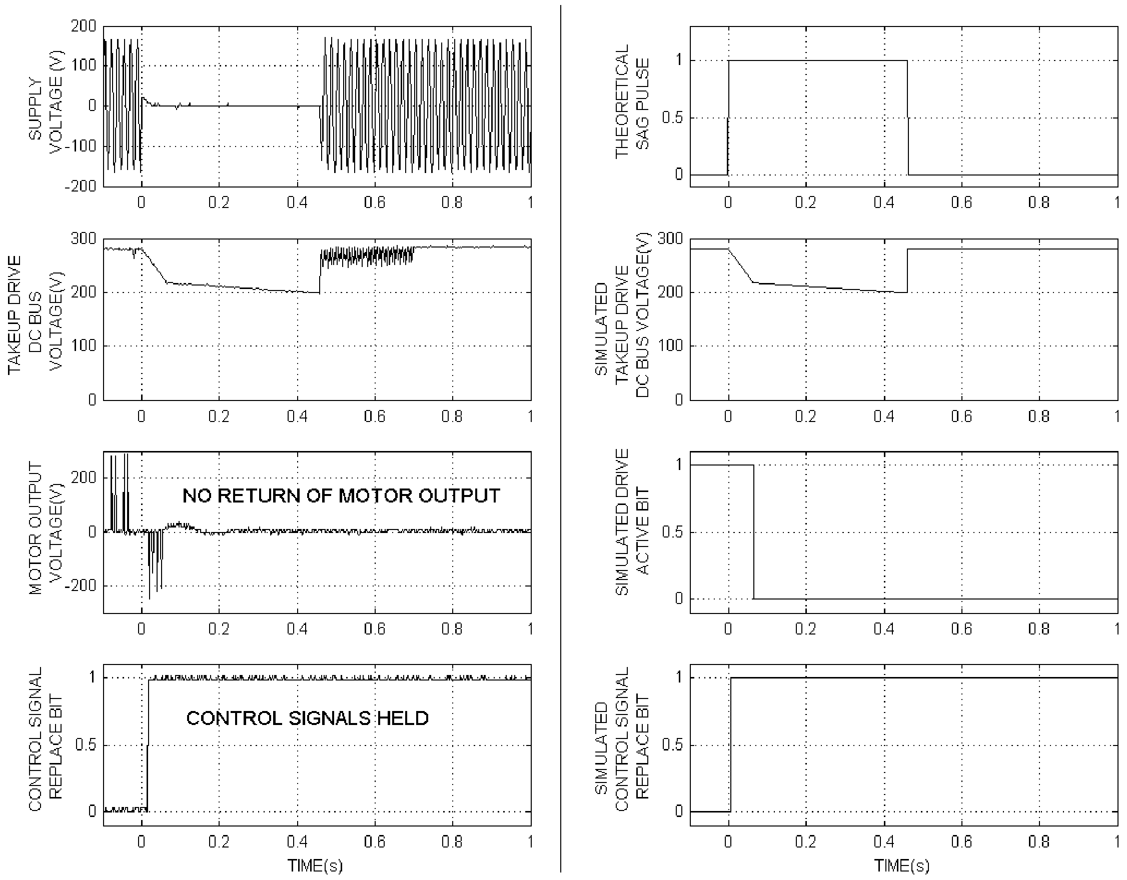


Figure 4.13 Experimental and simulated DC bus magnitude, motor output, and control signal replace commands during unsuccessful software momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 3ϕ voltage interruption).

Hardware Assisted Momentary Drive Coast

Figure 4.14 shows an experimental case of the hardware-assisted drive coast routine for a three-phase voltage interruption of 0% magnitude applied for $450ms$ duration. Both linespeed and tension responses are similar in form to the software coast routine. The slack condition again causes a difference between the experimental and simulated recovery responses. For the $450ms$ sag in Figure 4.14, the difference between the beginning of experimental and theoretical tension recovery is approximately $650ms$, whereas the difference for the shorter sag duration case of Figure 4.15 is approximately $250ms$. This indicates that for greater durations of zero tension and coasting linespeed, more slack is created between the payoff and takeup reels. Because of this, a delayed recovery occurs because the textile slack must be taken up before tension is restored.

This difference reveals noteworthy characteristics of the process dynamics while recovering from the coasting state. Substitution of prefault average control signals lead to an overshoot in measured line speed while the slack is taken up, followed by a return to the desired line speed while the web tension rises to its desired value and gradually applies load to the motors. When slack is not accounted for, the theoretical test runs will predict an overshoot in tension instead of line speed because the mechanical coupling between rollers and reels is considered to be always present.

For the hardware momentary drive coast algorithm, the coasting state response times were found to lie between $20ms$ and $45ms$. The execution of the drive halt operation is improved over the software momentary coast cases, where the successful case response time is $56ms$ and the unsuccessful case response time is $64ms$. The DC bus voltage magnitude is held higher than in the software case, thereby allowing more time to elapse before the undervoltage trip level crossover. Because of the difference in DC bus decay rates between the energized motor and de-energized motor, the improvement in response time pays off greater increases in allowable sag duration. This improvement may be quantified as

$$\Delta t_2 = \left(\frac{A_1}{A_2} - 1\right)\Delta t_1; \quad |A_1| > |A_2| \tag{4.1}$$

where $A1$ and $A2$ are linearly approximated slopes of the DC bus decay rates for the energized and de-energized motor, and Δt_1 and Δt_2 are time delay intervals as shown in Figure 4.16. When the software coast algorithm is used, the improvement in voltage sag duration susceptibility increases by Δt_2. Additional improvements gained using the hardware coast algorithm over the software coast algorithm may also be quantified with Equation 4.1, where the value of Δt_1 instead represents the additional improvement in response time over the software coast method.

Another remarkable outcome is that of the unnecessary operation case, as shown in Figure 4.17. In this scenario, the momentary drive dropout mitigation routines (both software and hardware-assisted) cause a brief coast condition, but the incoming voltage sag recovers before the DC buses would cross the undervoltage trip level if the routine were not applied. The observed process disturbance is therefore caused entirely by the mitigation algorithm and creates a less desirable result than a process voltage sag ride through. Though this condition will occur for a small set of possible voltage sag durations, potential fixes such as building delay times into the mitigation algorithms to minimize occurrences of the unnecessary case would be done at the expense of any gains in sag duration tolerance that the mitigation routines seek to achieve.

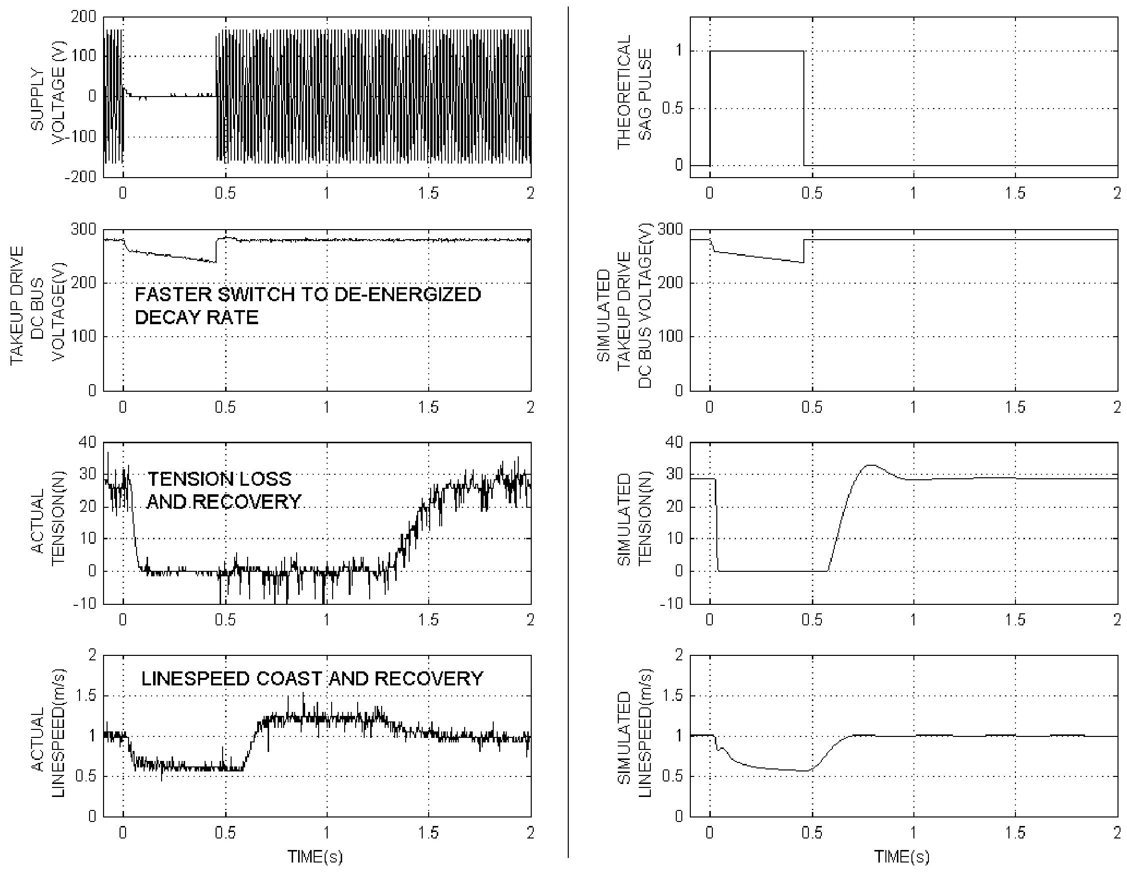


Figure 4.14 Experimental and simulated process response using hardware assisted momentary drive coast algorithm (1.0m/s, 30N process setpoints; 0%, 450ms, 3ϕ voltage interruption).

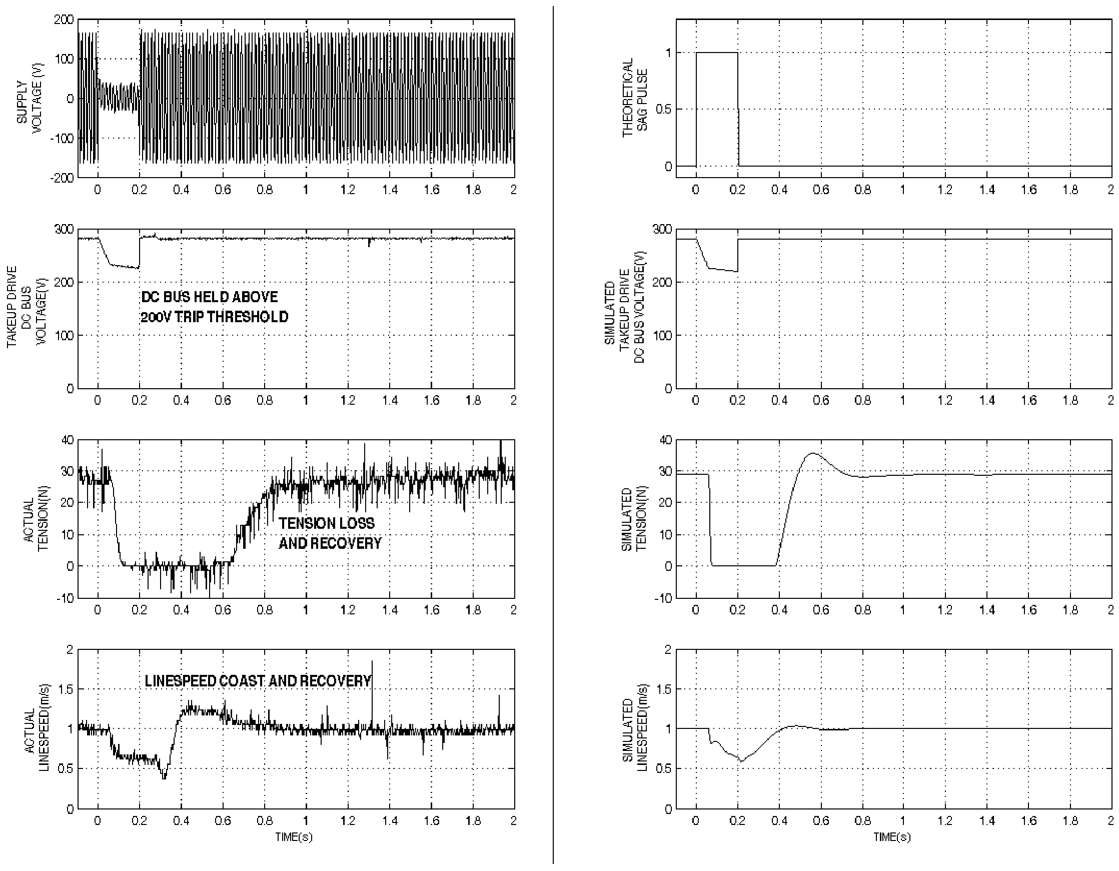


Figure 4.15 Experimental and simulated process response using hardware assisted momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 20%, $200ms$, 3ϕ voltage sag).

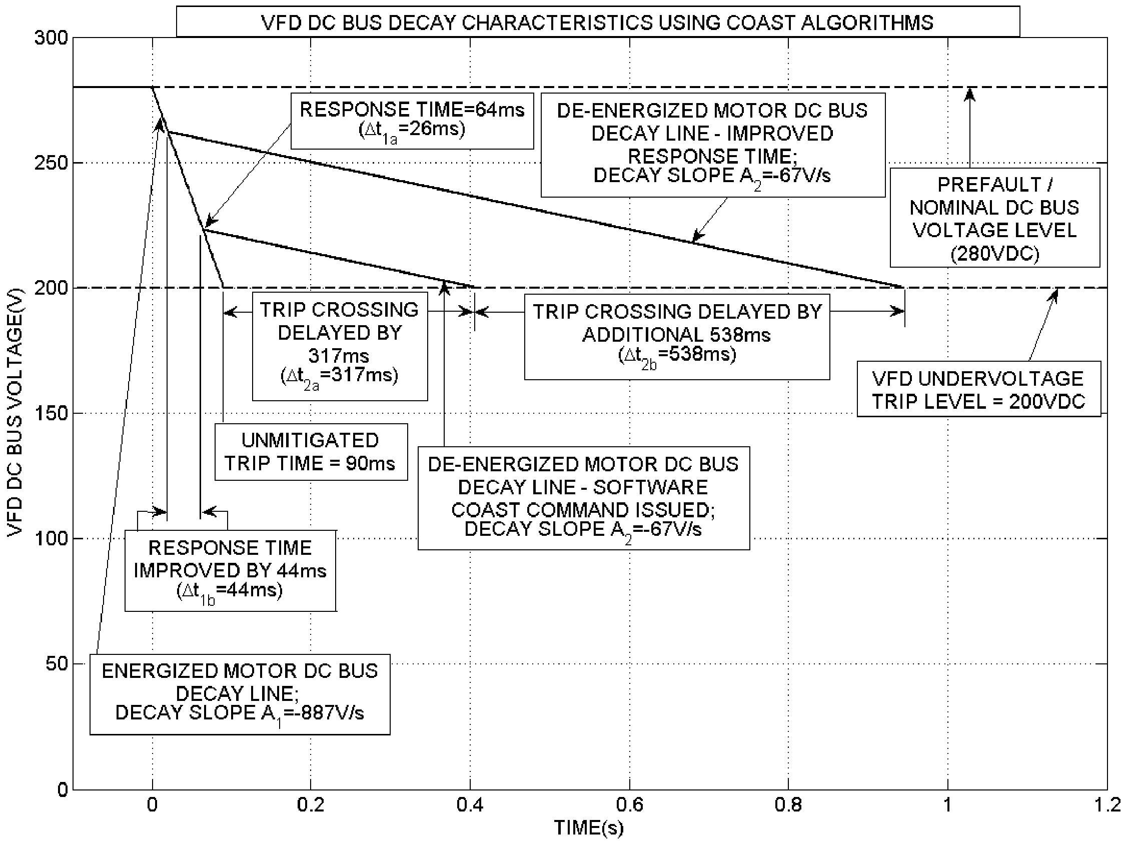


Figure 4.16 Comparison of DC bus decay rates during momentary drive coasting algorithm executions.

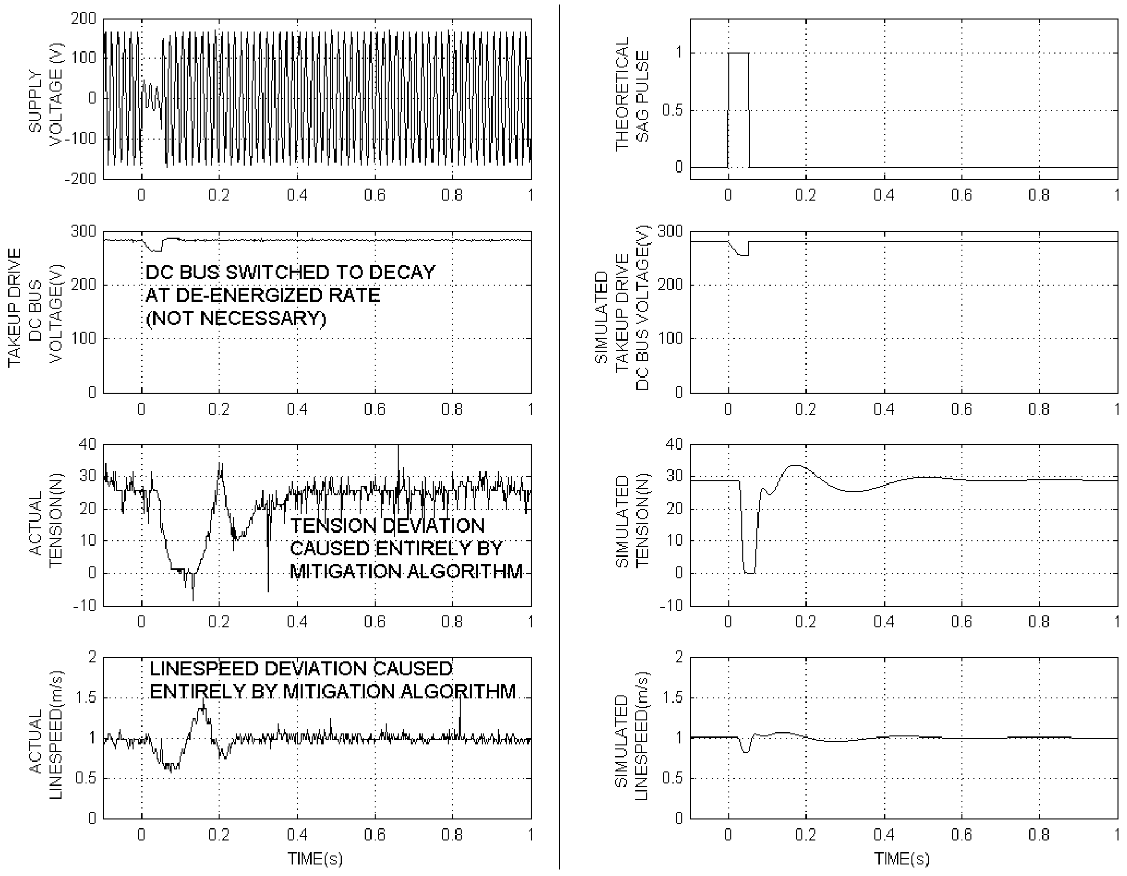


Figure 4.17 Experimental and simulated process response unnecessarily using hardware assisted momentary drive coast algorithm ($1.0m/s$, $30N$ process setpoints; 20%, $200ms$, 3ϕ voltage sag).

Faulted Instrumentation Routines

The faulted instrumentation routines mitigate the process deviation response by interrupting and substituting feedback and control signals in the PLC controller. The characteristic unmitigated faulted instrumentation response is shown in Figure 4.18, where the applied fault is a single phase voltage interruption on Phase C of the power system.

Control Signal Substitution

The control signal substitution routine replaces the drive setpoint control signals with their prefault averages when faulted instrumentation is expected. Integral components of the PI controllers are held to avoid accumulation of controller error. The experimental and theoretical responses of the textile tension control stand using this algorithm are shown in Figure 4.19. In this case, the AC motor drives do not drop offline, but instead the tension sensor and tachometer produce faulted, erratic signals. In response, the control signal replace bits are set (Figure 4.20) 19ms into the event and for a delay of 333ms following the sag. The transition back to standard control occurs at t=805ms. Algorithm execution results in the maintenance of both linespeed and textile tension for the voltage sag event and hence a process ride through response. Substituted signal transitions occur without any resulting disturbance in the process outputs. However, during the open loop/signal substitution period, it is important that no mechanical disturbance requiring closed loop control occurs in the process. If this condition is met, the control signal substitution algorithm demonstrates an effective means of improving process ridethrough in cases of faulted instrumentation.

As Figure 4.21 illustrates with a 50ms, 1ϕ interruption, there exists an unnecessary case response when applying this method. In this scenario, all instrumentation rides through the event because the incoming voltage sag is not of sufficient duration to exhaust the instruments' ridethrough for the applied voltage sag depth and process physical conditions. No misoperation occurs in either the tachometer or the tension cell, yet detection and measurement of the sag triggers the algorithms's execution. Signal substitution occurs between $t = 21ms$ and $t = 395ms$. Unlike the unnecessary case seen with momentary drive coasting routines, no resulting process disturbance occurs. While occurrences of such a scenario are limited to voltage sags of brief duration, they demonstrate the absence of customization with regard to individual instrument voltage sag characteristics. Therefore, while the functionality of this routine is sound, room for improvement exists to the extent that variations in instrument voltage sag responses may be accounted for when determining the need to substitute process signals.

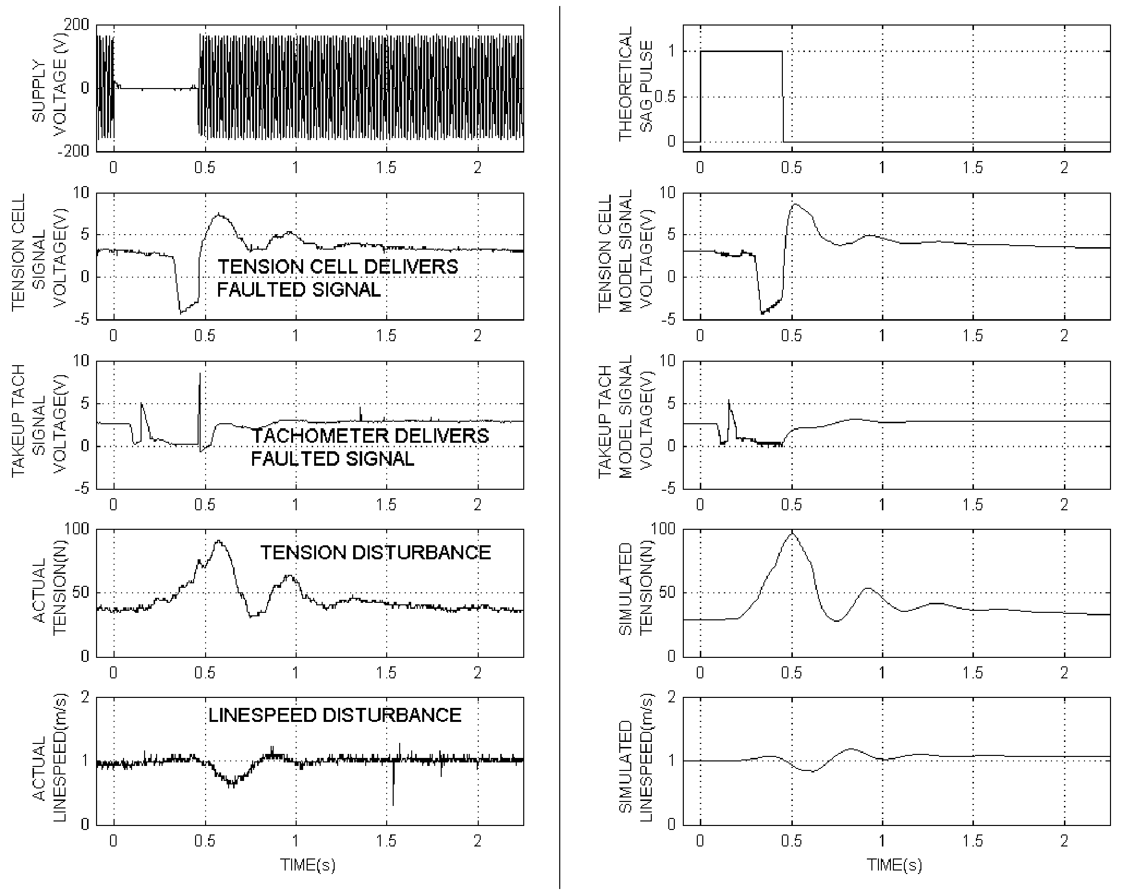


Figure 4.18 Experimental and simulated unmitigated process response resulting in faulted instrumentation ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

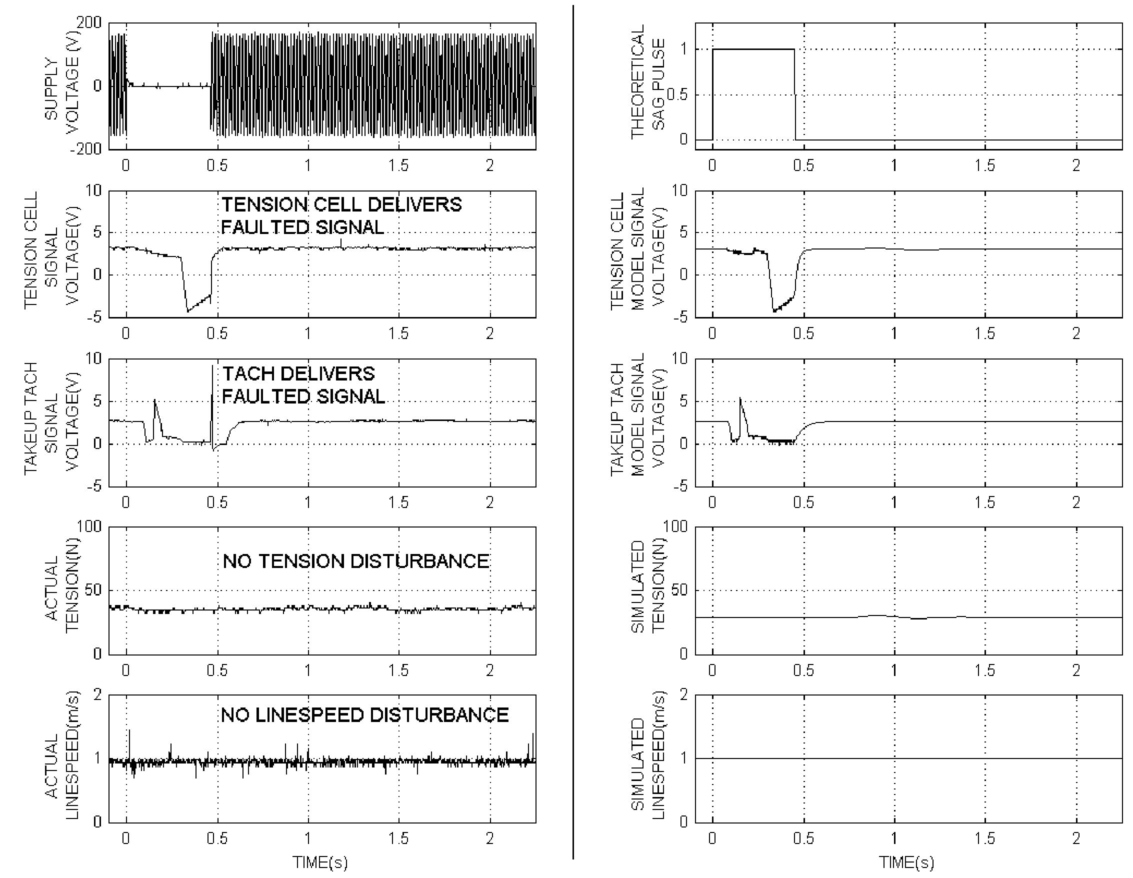


Figure 4.19 Experimental and simulated process response using control signal substitution algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

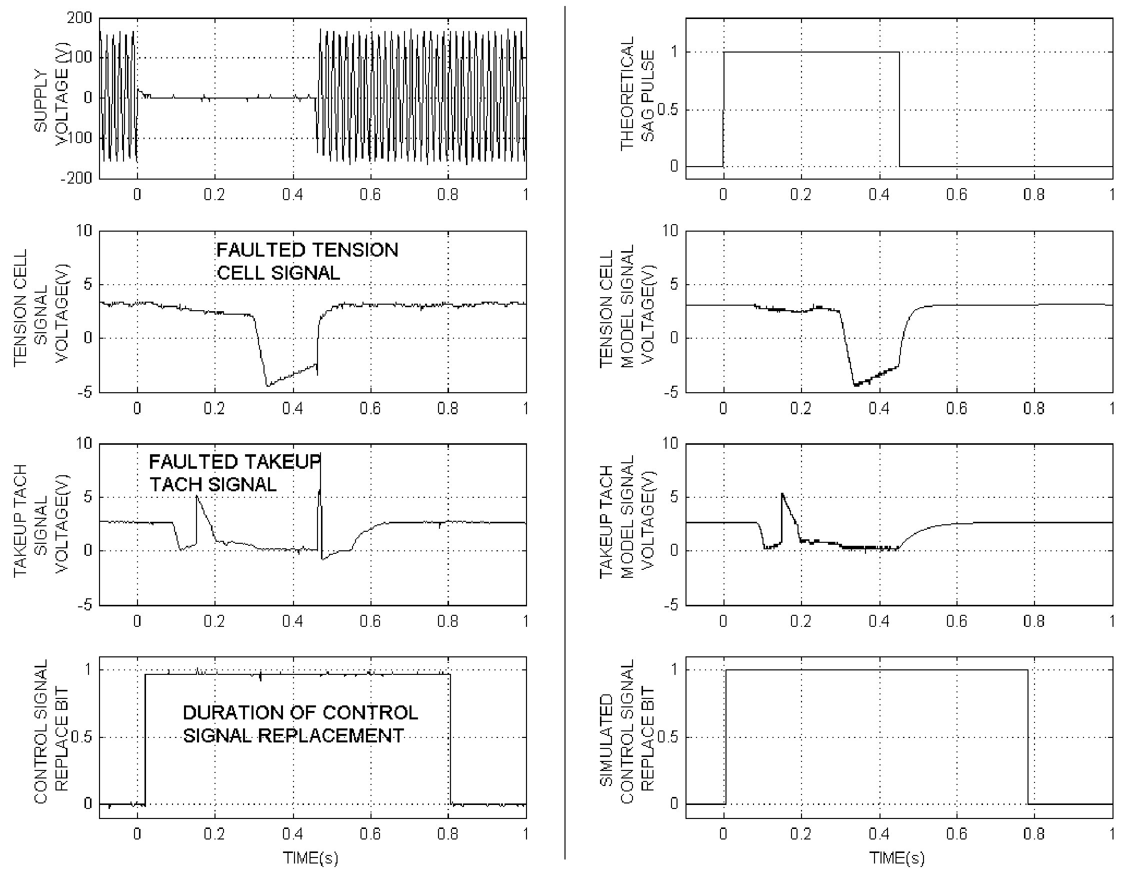


Figure 4.20 Experimental and simulated faulted instrument signals and control replace bits ($1.0 m/s$, $30N$ process setpoints; 0%, $450 ms$, 1ϕ voltage interruption).

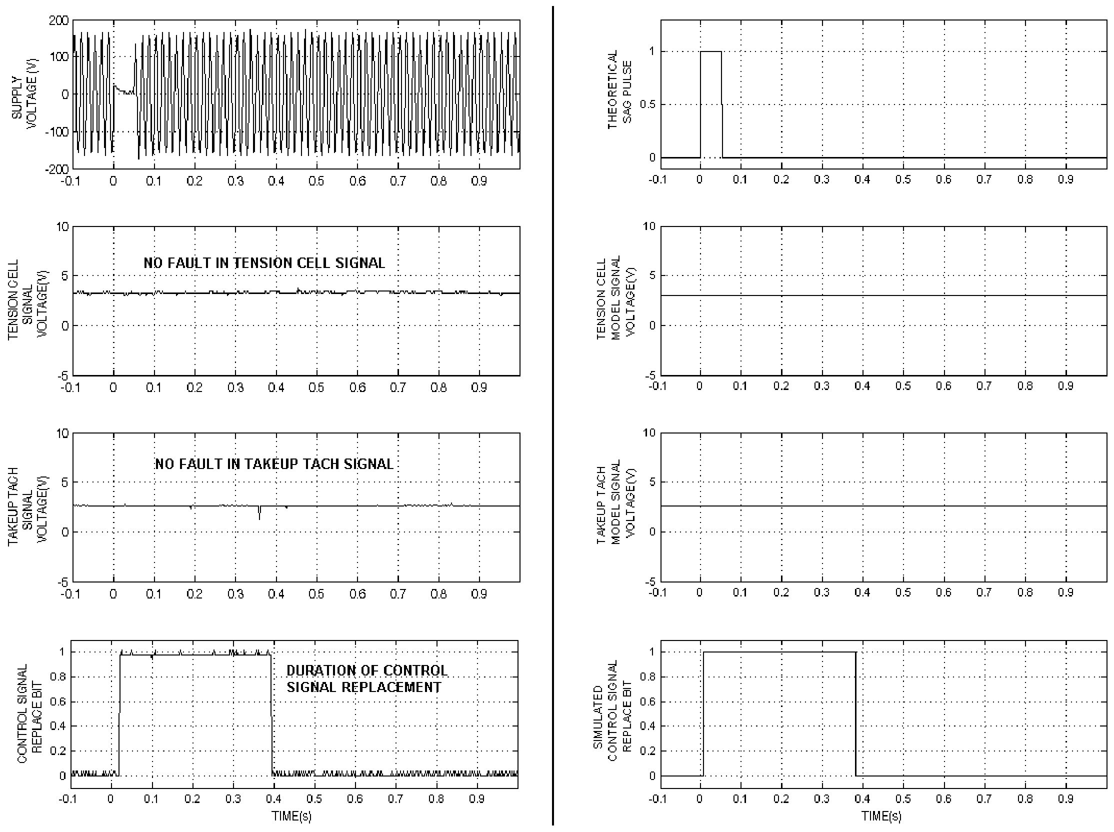


Figure 4.21 Experimental and simulated process response unnecessarily using control signal substitution algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $50ms$, 1ϕ voltage interruption).

Selective Signal Substitution

The selective signal substitution algorithm is designed to minimize the replacement of instrument feedback signals in cases where fault and process conditions will not cause an instrument error. The input variables of physical instrument inputs and voltage sag magnitude are used to index arrays where the appropriate ridethrough time is found and used as a signal switching delay.

The textile tension controller unmitigated case for $1.0m/s$, $10N$ process inputs was shown in Figure 4.18, where the faulted tension and tachometer inputs are shown to produce a process disturbance for both linespeed and tension. Using the selective signal substitution algorithm, the mitigated response for these conditions is shown in Figure 4.22. Here both the experimental and theoretical responses indicate faulted instrumentation signals, yet no appreciable linespeed or tension disturbances result. The signal replace intervals are shown in Figure 4.23 and illustrate different delay times for the initial signal substitutions as well as different delay times for restoring the original signal feeds.

A test case for different physical input levels of $2.5m/s$ linespeed and $50N$ tension is shown in Figures 4.24 through 4.26. The unmitigated response of Figure 4.24 shows a significant disturbance for a single phase voltage interruption of duration $450ms$. Figures 4.25 and 4.26 show the selective signal substitution method functioning as predicted, with no resulting process disturbance and signal transition times that are customized to the input disturbance and process prefault characteristics.

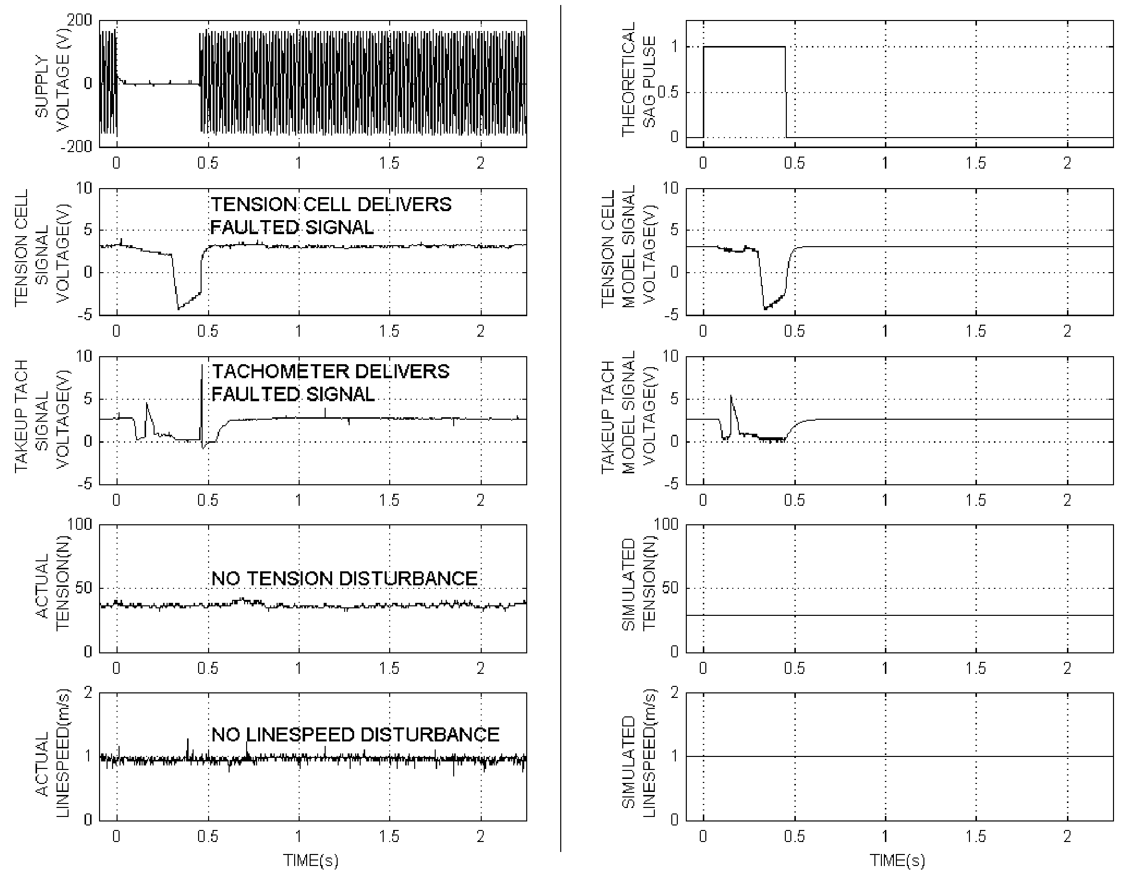


Figure 4.22 Experimental and simulated process response using selective signal substitution algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

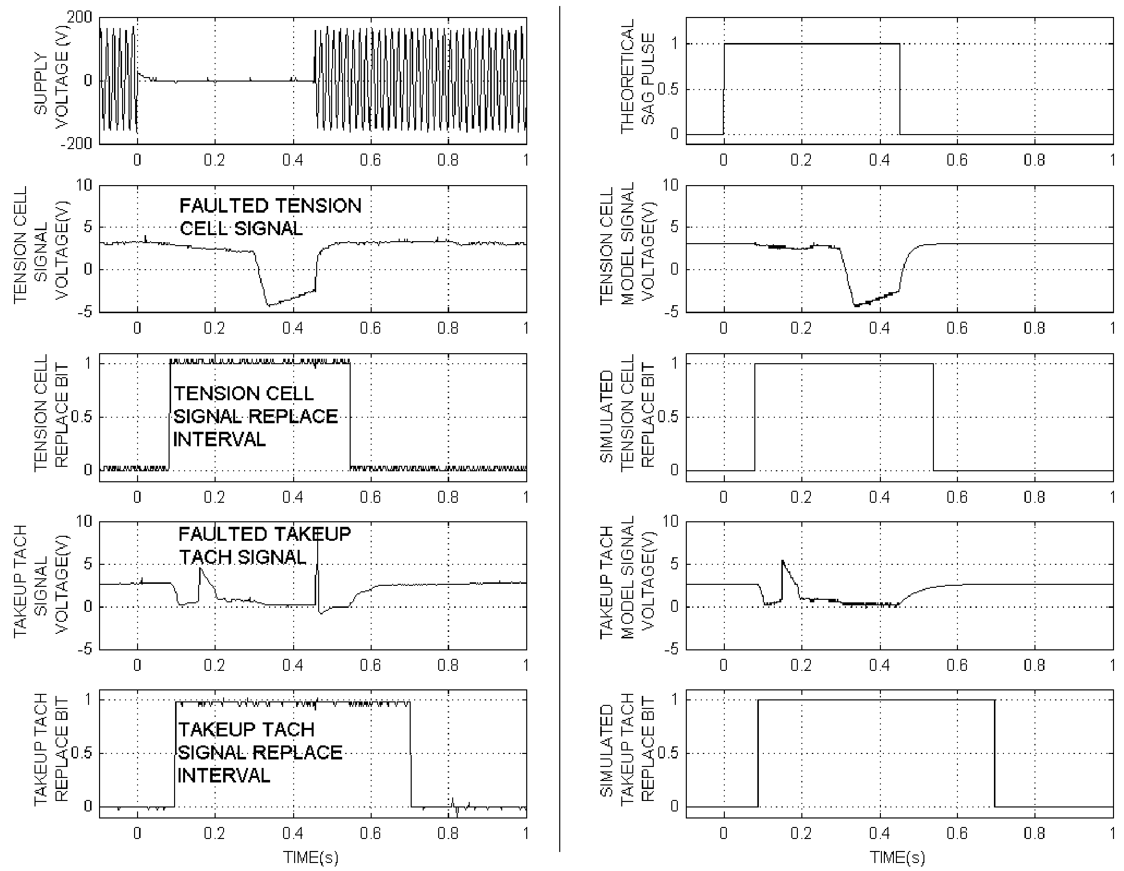


Figure 4.23 Experimental and simulated faulted instrument signals and replace bits for selective signal substitution algorithm ($1.0m/s$, $30N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

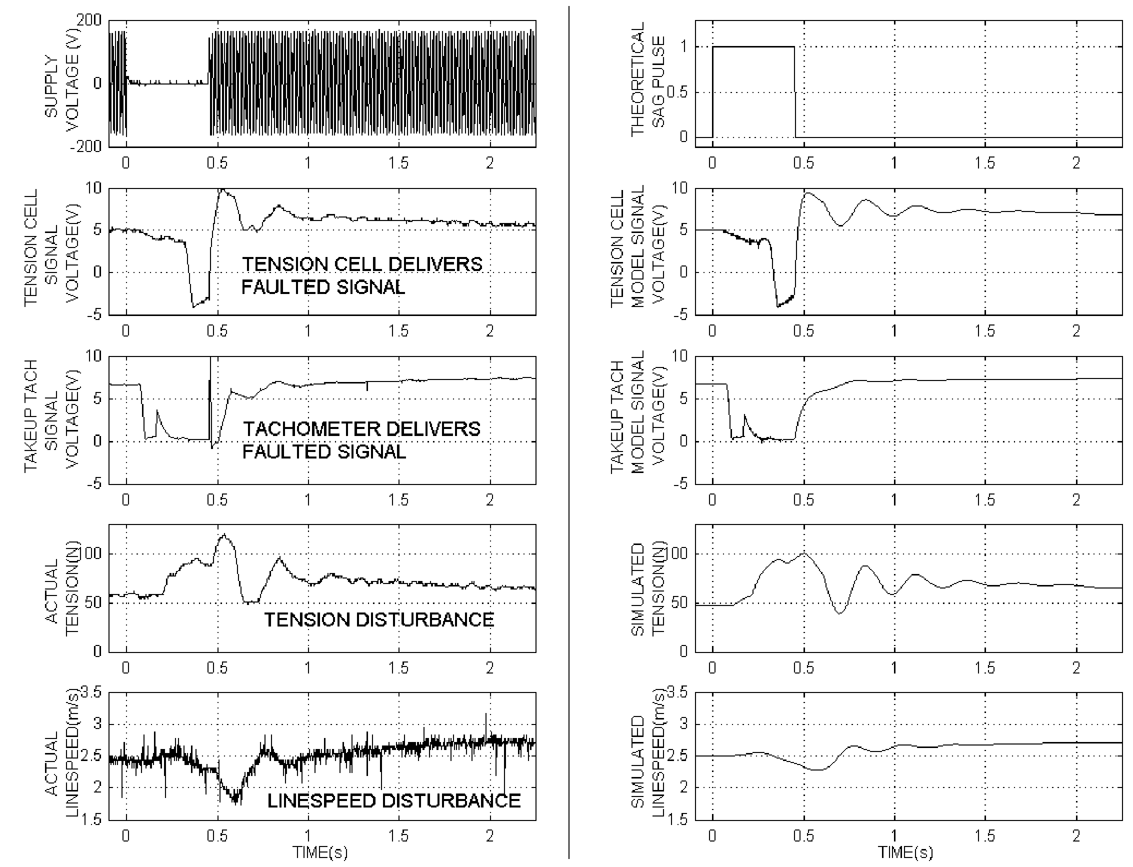


Figure 4.24 Experimental and simulated unmitigated process response resulting in faulted instrumentation ($2.5m/s$, $50N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

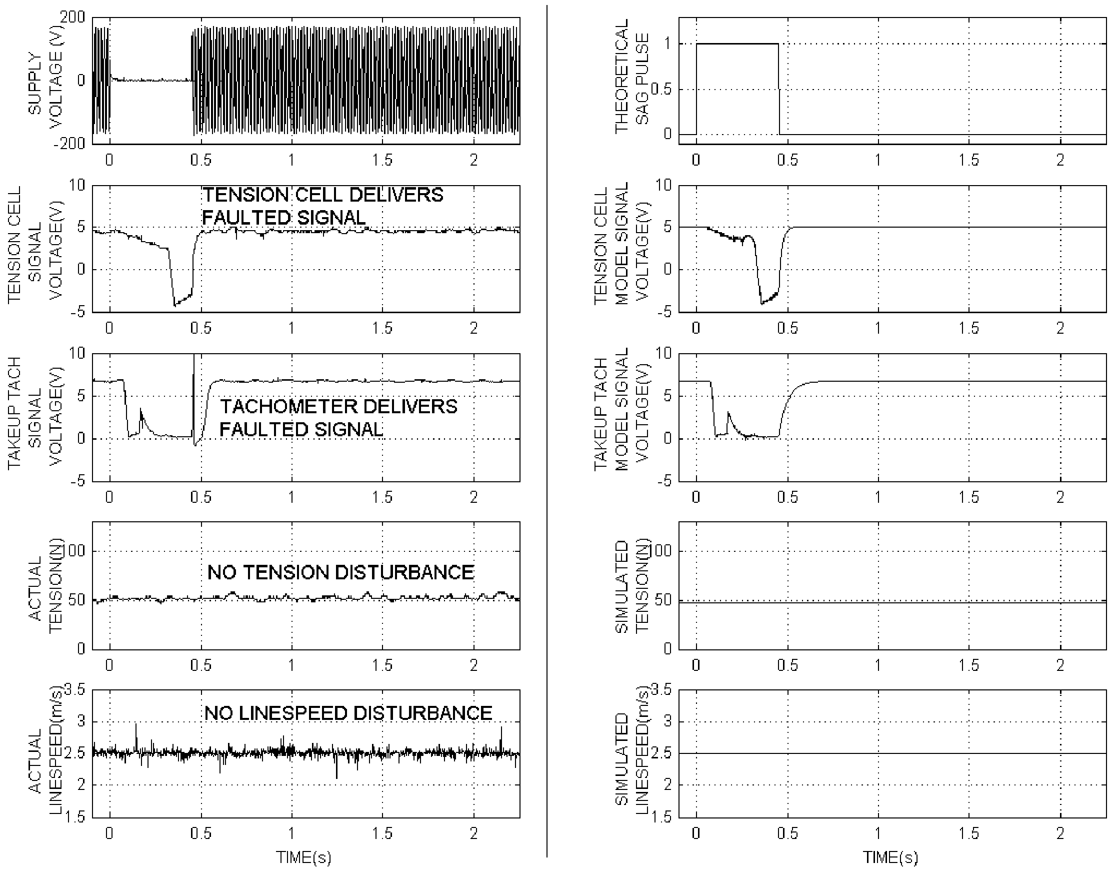


Figure 4.25 Experimental and simulated process response using selective signal substitution algorithm ($2.5m/s$, $50N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

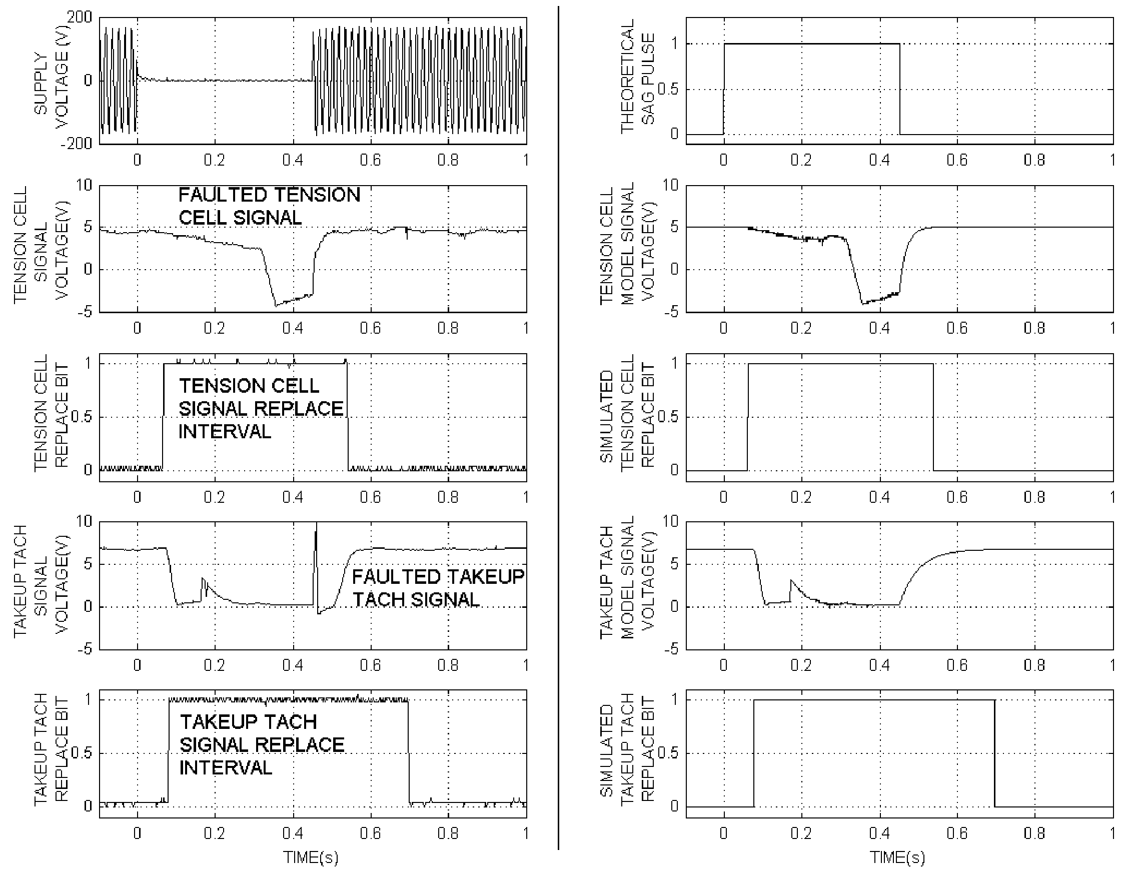


Figure 4.26 Experimental and simulated faulted instrument signals and replace bits for selective signal substitution algorithm ($2.5m/s$, $50N$ process setpoints; 0%, $450ms$, 1ϕ voltage interruption).

A case where only a single instrument is substituted is shown in Figures 4.27 through 4.29. In the unmitigated case, the tension cell measures a web tension of $10N$ and throughout the event provides a reasonable measurement that does not contribute to the process disturbances. The tachometer, however, does deliver a faulted signal and hence calls for replacement during the 40%, $450ms$, 1ϕ supply voltage sag. Upon applying the selective signal substitution algorithm, the linespeed and tension disturbances are eliminated, as shown in Figure 4.28. The instrument signals and switching signals shown in Figure 4.29 indicate that this particular execution of the selective signal substitution algorithm did not include substitution of the tension cell signal, thereby allowing it to function in the control loop in its normal capacity. A noteworthy aspect of this test case is that at first inspection of the unmitigated response it might seem that the tension cell misoperates and contributes to the process disturbance. However, the instrument ridethrough time arrays indicate a ridethrough condition for the given fault scenario and therefore prove themselves a more reliable means of determining the appropriate signals to replace.

Experimentation indicates that the selective signal substitution method functions adequately and is an improvement over the control signal substitution method because the process controller is not bypassed unless absolutely necessary. The inherent ridethrough that exists in each device is used to its full advantage, which optimizes the amount of time the PLC controller may be left unassisted before intervention. The three salient test cases are compared in Figure 4.30, which illustrates the variation in signal substitution intervals for each process response scenario.

Trivial cases of selective signal substitution algorithm execution also exist. When the incoming voltage sag is determined to cause a possible faulted instrumentation response, but the voltage sag duration and process variable conditions do not lead to any signal replacement, the routine nonetheless executes, begins timing for replacement commands, but never sends them. In these trivial case executions, closed loop control is maintained throughout the duration of the incoming voltage sag event. This ultimately leads to no mitigating action, no process disturbance, and maintenance of all instrument signal feedback paths into the process controller.

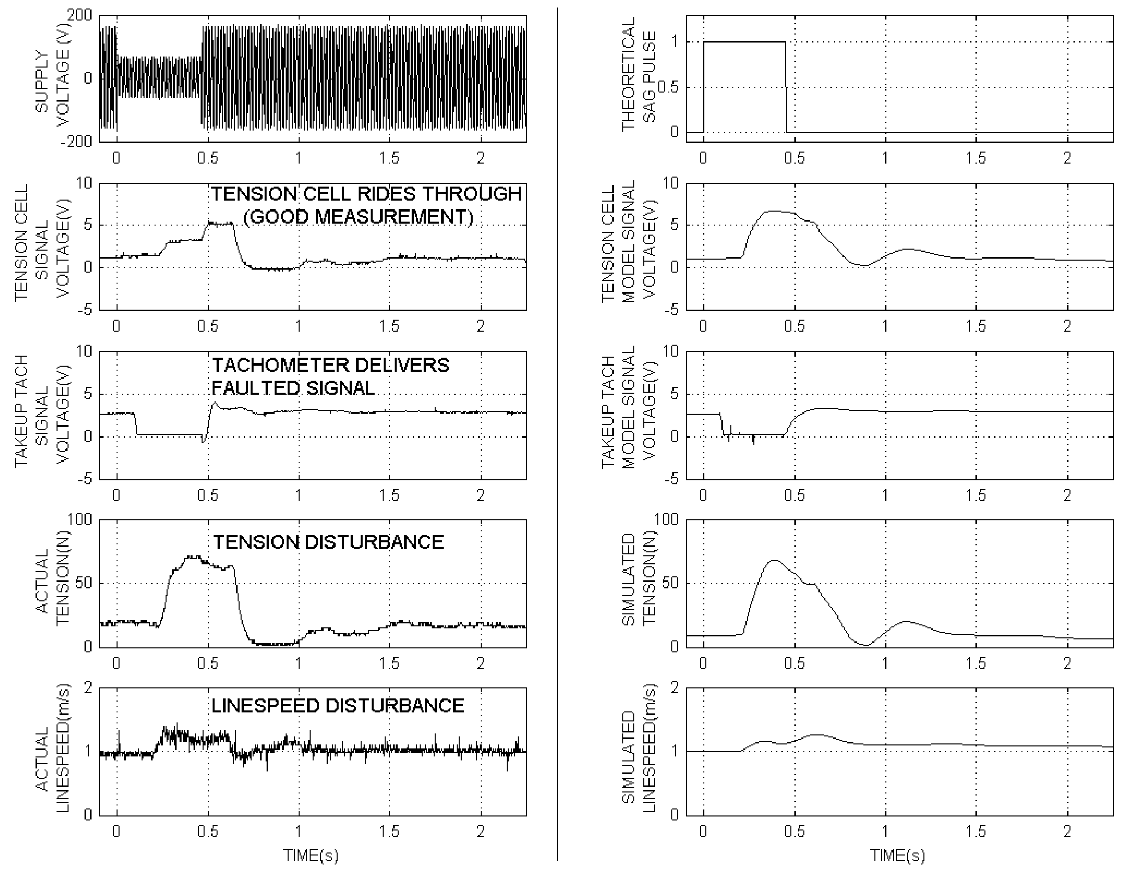


Figure 4.27 Experimental and simulated unmitigated process response resulting in faulted instrumentation ($1.0m/s$, $10N$ process setpoints; 40%, $450ms$, 1ϕ voltage sag).

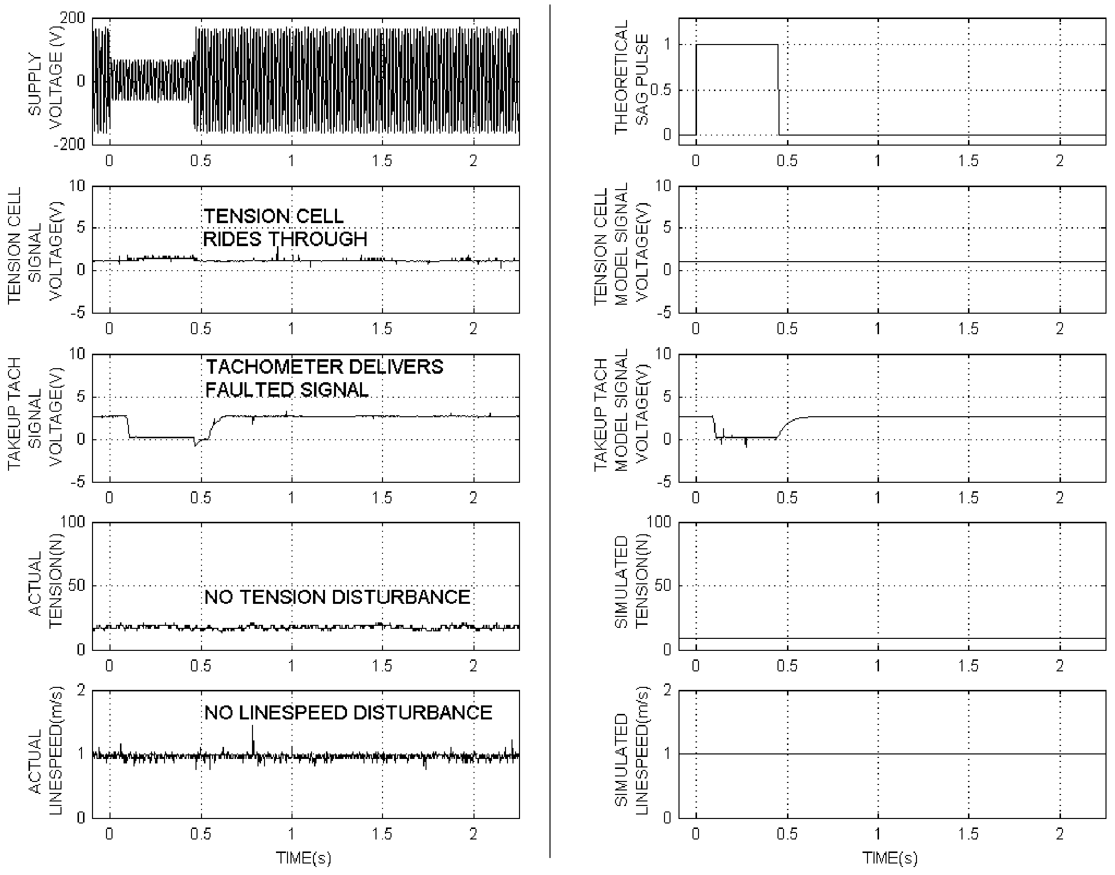


Figure 4.28 Experimental and simulated process response using selective signal substitution algorithm ($1.0m/s$, $10N$ process setpoints; 40%, $450ms$, 1ϕ voltage sag).

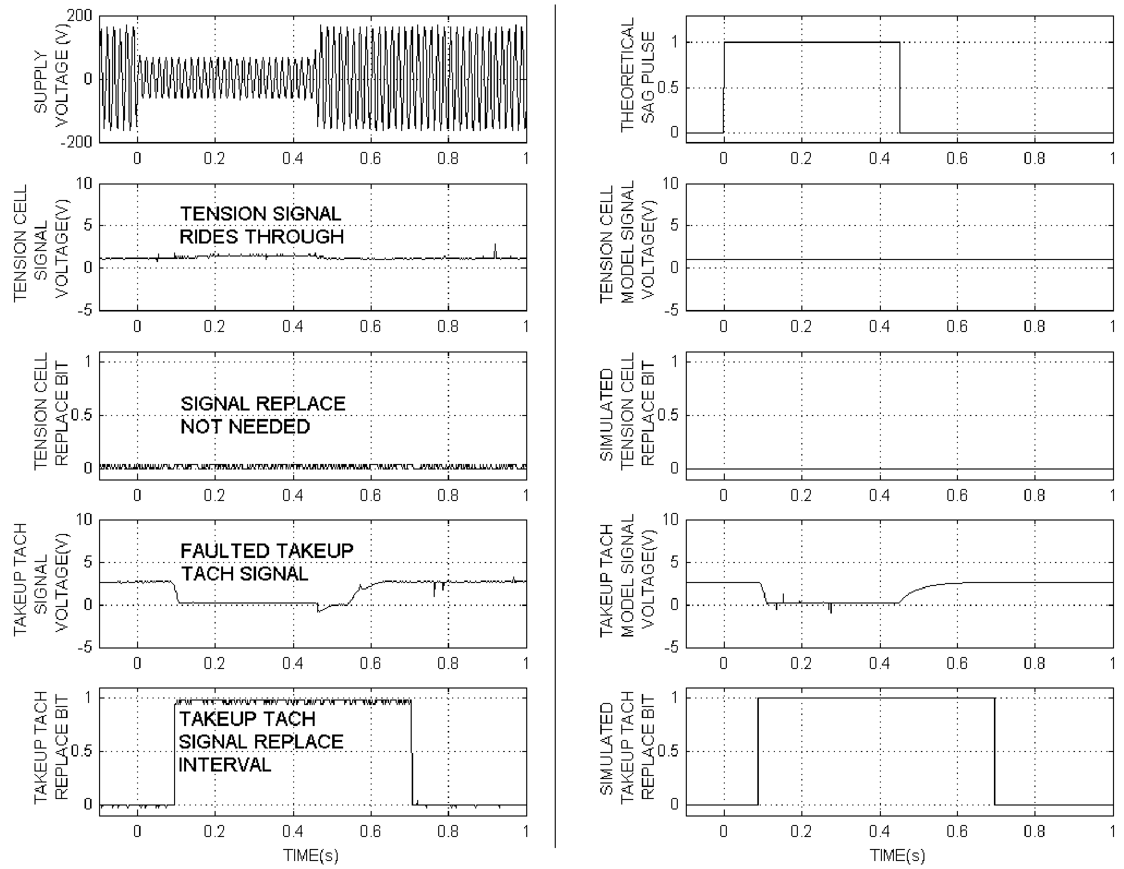


Figure 4.29 Experimental and simulated faulted instrument signals and replace bits for selective signal substitution algorithm ($1.0m/s$, $10N$ process setpoints; 40%, $450ms$, 1ϕ voltage sag).

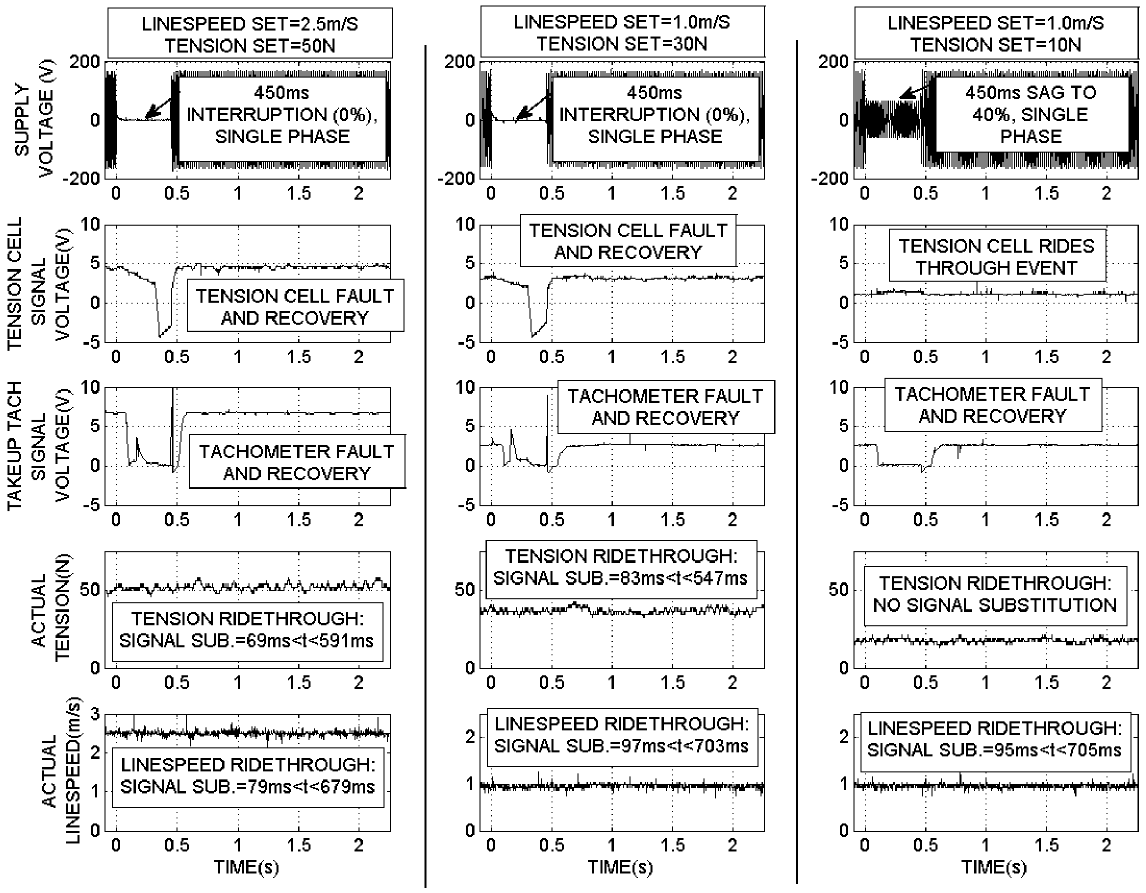


Figure 4.30 Comparison of process responses and signal substitution intervals for selective signal substitution algorithm executions.

Summary

An analysis of the Ridethrough PC operation was performed as it applies to the test bed textile tension control system. Execution times were evaluated and found to be strongly dependent on network delay and process controller cycle times. Detection methods were found to trigger slightly longer than predicted, yet still prove effective when used for the Ridethrough PC application. Measurement methods were found to function as predicted, yet show sensitivity to waveform distortions and voltage transients.

The software and hardware coast mitigation routines perform properly in their mitigation of voltage sags and interruptions that would cause AC motor drive dropout. The software momentary drive coast method shows a strong dependence on network latency and processing delays. Execution times for the hardware coast mitigation method showed a significant improvement over the software coast method, but at the cost of minor hardware additions. Both coast methods halt the motor outputs in order to prevent the decaying DC bus from crossing its trip threshold, and cause a brief loss of tension and linespeed decay. Recovery occurs in a gradual fashion and eliminates extreme overcompensation caused by the accumulation of integral control error. In cases of extended sag duration, a trip threshold crossover will occur even if the mitigation routine is applied. Therefore, the effectiveness of the coast routines' mitigating efforts may be quantified by the additional time the algorithms provide before a drive DC bus undervoltage fault occurs. Unnecessary uses of the drive coast routines were shown to exist, but their presence is restricted to sags that are of shorter duration than the unmitigated dropout time.

When applied to the textile tension control system, the signal substitution routines were shown to improve the faulted instrumentation process response and agree with their predicted outcomes. In tests of the control signal substitution method, faulted instrument signals were blocked from propagating to the process drives at the output stage. Arbitrarily substituting the output controls proves effective, but reveals an unnecessary case of operation at times when a voltage sag recovers before instrument ridethrough times elapse, or when physical input and voltage sag magnitude conditions do not produce instrument faults. The unnecessary case is minimized by the case of the selective signal substitution algorithm, which shows variation in the use of and timing of switching prefault average instrument outputs into the process controllers. This method is an improvement over control signal substitution for all expected faulted instrumentation cases, as it allows the instrumentation to remain in service when its misoperation is not anticipated. Trivial cases of selective signal substitution were found to occur, but do not result in any corrective action.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

Investigation Conclusions

This dissertation presents novel software based means of mitigating the effects of voltage sags on network connected industrial processes. Underlying design criteria are to avoid the use of conventional mitigation hardware and to minimize disturbing the existing system to which the solution is applied.

A dedicated textile tension control process is used as a test bed for applying the software algorithms. Voltage sag testing on this process indicated that one of three distinct responses can occur: AC drive dropout leading to process shutdown, faulted instrumentation leading to process variable deviation, or no observed response (ride through). Expressions for mathematically predicting the process response are derived, where inputs are voltage sag magnitudes for three input power phases and outputs are Boolean variables indicating an expected response.

The mitigation algorithms are applied using an add-on PC, or 'Ridethrough PC', that interfaces with the supply voltages and the Profibus automation network. Through monitoring and recording of network traffic during a voltage sag, a detailed process history is established as an additional means of identifying the expected process response. This history may be used independently of the mathematical expected process identification.

Using the Ridethrough PC, voltage sags are detected using an analysis of RMS voltage derivatives and their magnitudes are measured using modified peak detection. The expected process response is determined either through analysis of the process history or Boolean calculation of the expected response.

Four voltage sag mitigation algorithms are proposed. A coast algorithm that is exclusively software controlled addresses the AC drive dropout response. This method preemptively forces the motor drives to coast in order to delay an undervoltage trip condition. An anticipated temporary loss of tension and drop in linespeed results, but the process recovers after the sag has ended, thus avoiding significant downtime, product spillage, and user restart/reset of the process.

The software coast algorithm provides improved ridethrough for the process, but is nevertheless susceptible to incoming sags of longer duration. Decreasing susceptibility to longer duration voltage sags is achieved with the hardware assisted momentary coast algorithm, which forces motor drive output contactors to an open circuit state as a means of forcing the process to coast. Additional ridethrough beyond the software coast algorithm is provided by an increased response time that stems from eliminating the automation network and PLC delay times.

The faulted instrumentation process response is mitigated by one of two algorithms. In the control signal substitution method, the output controls are held to their prefault average values for the duration of the sag plus a postfault delay time. During this time, the PI controller integrals are held to prevent an unnecessary accumulation of integral error before the signal routing is restored. This algorithm performs well in maintaining process conditions during expected faulted instrumentation voltage sags, but often performs unnecessary executions when the combination of voltage sag magnitude, sag duration, and device physical input conditions will not cause an instrument fault.

Improvements in faulted instrumentation methods are made in the selective signal substitution mitigation algorithm, which accesses stored ridethrough times as a function of voltage sag magnitude and instrument physical input for use as signal substitution switching delays. Maximum recovery times are also used as times to switch the signals back to their original routing. Experimentation with this algorithm proved successful; signal substitutions are made after an instrument's ridethrough time has expired and are not made when instrument disturbances are not anticipated by the event conditions. This algorithm effectively mitigates cases of expected faulted instrumentation up to the maximum test sag length of $450ms$, maximizes the amount of time an instrument feedback signal may remain in a feedback path, and reduces instances of unnecessary signal substitution.

Development of the Ridethrough PC and the software-based voltage sag mitigation algorithms combined with experimental verification applied to a test bed textile tension control workstation yields the following conclusions:

1. Use of software algorithms that execute in the presence of voltage sags can be an effective means of mitigating process disturbances caused by voltage sags.

2. A recorded process response history or a mathematical response estimation may be used to determine the appropriate course of action for software based voltage sag mitigation.

3. AC drive dropout failures may be mitigated by application of software issued coast commands. This solution increases the allowable time that an AC drive dropout sag may last. Maintenance of process parameters during the coast state is dependent on the specific process mechanics, but the ability to automatically restore these parameters is enhanced by application of the coast routines.

4. Use of external contactors to force the coast state, as in the hardware assisted momentary drive coast algorithm, improves the sag duration tolerance of the process by eliminating the coast command delays of the automation network, PLC program, and AC drive. This would not be the case if the latency of the command path was less than the operating time of the external contactor.

5. Cases of unnecessary drive coast algorithm execution will occur when incoming voltage sags are of a magnitude that will ultimately cause a dropout condition, but are of a sufficiently brief duration such that the undervoltage fault would not result. These cases produce process disturbances rather than mitigate them.

6. Arbitrary substitution of control signals during an incoming voltage sag that is anticipated to cause a faulted instrumentation condition may lead to unnecessary substitution of control signals and excessive instances of forced open loop control.

7. Implementation of predetermined instrument ridethrough and recovery times in the decision making process for signal substitution can decrease the instances where feedback signals are unnecessarily ignored and help preserve closed loop process control in the presence of supply voltage sags.

8. Ridethrough improvements made with these strategies are constrained by communication network latency and controller execution delays. These delays may be reduced if execution bottleneck paths are bypassed.

Research Contribution

This investigation has focused on several topics that to date remain unexplored:

1. Existing software ridethrough solutions use only the presence of a voltage sag as a means of triggering alternate control. The methods herein use measurable sag characteristics as criteria for determining a required action.

2. A process history is utilized for the purposes of voltage sag mitigation. Process variables and voltage sag event data are recorded and stored together. The unmitigated process response is analyzed during an incoming event to determine an expected response.

3. Process voltage sag response boundaries are expressed mathematically. These expressions are then evaluated during in-progress voltage sags to determine the expected process response for mitigation purposes.

4. Measured variations in instrumentation responses are used in determining the need for momentary open loop control. Sag magnitude and physical input levels serve as inputs in determining the allowable delay time before substituting feedback signals. This approach utilizes the available ridethrough in each device and minimizes the time open loop control is used.

Future Research Opportunities

System Implementation

Future work in this area should initially aim at improvements in implementation. Programming the Ridethrough PC in a high-level programming language can reduce the computational expense of the graphical LabView software. Faster and higher resolution data acquisition hardware may be used to improve voltage sag detection and sag measurement. To the extent that process controllers will permit, the software may reside primarily in a process PLC to reduce the dependency on supervisory systems.

More advancements may be found in the application of fuzzy logic or neural network control that functions in a secondary or backup controller capacity in the presence of a supply voltage sag. It is likely that these solutions are more feasible in a process that does not include a PLC and exclusively uses PC control. In this modified architecture, the additional Ridethrough PC may be unnecessary, making application of software mitigation algorithms less reliant on fieldbus network traffic.

Application of Commercially Available Equipment

Complexity of the Ridethrough PC can be reduced by using available measurements from commercial metering or protective relaying equipment. This modification would potentially eliminate the requirement for data aquisition hardware in the Ridethrough PC and the task of constructing custom voltage sag detection and measurement subsystems. Use of equipment with advanced sag detection and measurement capabilities would provide additional benefits by increasing the ability to respond to sags greater than the $450ms$ boundary of this study, sags of varying magnitude, and successive sags such as those seen during recloser operation. Implementation of advanced monitoring functionality may also allow the Ridethrough PC to perform multiple decision-making operations rather than respond to a single event at a time.

For this modification to be successfully introduced, several equipment characteristics must be considered. Voltage sag detection and measurement functionality must be comparable or superior to the methods shown in this study in their sensitivity, accuracy, and response times. Effects of device placement on voltage measurements must also be considered. A general preference should exist for devices in close proximity to the process under analysis to minimize compensation related to power system configuration and connection schemes. Additionally, locally placed devices are favorable because they are more likely to respond to locally generated voltage sags (as with high current motor starting) and represent fewer challenges in establishing a communication link to a software mitigation system. Network communication capabilities are also of concern. In order for detection and measurement data to initiate response identification and mitigation routines in a timely fashion, communication latency between the metering/protective relaying device and the mitigation system should be minimal. Other specifications such as network protocol compatibility, network traffic burdens, and communication methods (master/slave or peer-to-peer) should also be assessed to determine the feasibility of applying commercially available hardware.

Microprocessor-based power quality or system protection relays offered by manufacturers such as Schweitzer Engineering Laboratories, GE, ABB, or Siemens are potentially suitable for such an application. For some equipment offerings, system power quality information, including voltage sag data, is gathered and may be quickly accessed from system data registers [74,75]. These devices are typically furnished with network interfaces such as DNP, Modbus, Modbus Plus, Ethernet, or Profibus (Siemens) that can facilitate communications with external systems. If supported by the device, time synchronization technologies such as synchrophasor measurements [76,77] may be employed as a means of accounting for data transfer delays. With these features available, application of commercially available protection or metering equipment may be realized in a software-based process voltage sag mitigation system.

Response Identification and Mitigation Routines

Both methods of identifying an expected process response are applied in an environment that assumes either a completed process voltage sag history or knowledge of the expected process response boundaries. Future development of response identification should focus on determining an expected response when only a sparse data set, either in the historical record or in response region boundary estimation, is available. In this scenario, a combination of process history and boundary estimation can be used, where a more clearly defined outcome is given a greater weight in suggesting the expected response. Improved process response history management may be implemented in an effort to fill the historical record with sags that are of the greatest available duration, so that for equal magnitude and phasing combinations, recorded effects under longer sags overwrite those of shorter ones. Combining these methods may also permit automatic generation and adaptation of Boolean response identification equations from the process history. On a system database level, traditional SCADA process historian systems may be implemented in future studies. By accessing a global process event history, a condensed voltage sag event history may be assembled for use in anticipated response identification. This approach can ultimately eliminate the task of creating a separate process sag response history.

Specific mitigation algorithms may be further developed in several ways. Coast algorithms may be improved by further optimizing algorithm response times or eliminating the coast state altogether by incorporating intelligent management of stored energy injection or kinetic buffering/inertia ridethrough methods. Future signal substitution methods should account for variables that change over the course of a process run such as package accumulation or setpoint profiling. Hybrid routines that simultaneously address AC drive dropout and faulted instrumentation are also possible, and are likely to be necessary in cases where the AC drive dropout response is completely mitigated but a faulted instrumentation response remains.

Simulation and Experimentation

Improvements in process simulation methods may involve more detailed modeling of hardware sag responses, with the goal of entering sag characteristics or recorded fault waveforms as inputs to create a simulated process response. By doing so, simulated results may be added to a process sag response history. If computational expense will not affect system performance, the process dynamic model may also be used to predict the voltage sag response to eliminate reliance on event history access.

Experimental verification may also be enhanced by the application of the methods proposed in this study to a full scale industrial process. In its present or supplemented form, implementation and application to a variety of automation networks, equipment, and processes can further demonstrate the advantages and limitations of software-based voltage sag mitigation solutions.

APPENDICES

Appendix A

Textile Tension Controller Equipment List

This appendix contains tables of hardware used on the textile tension control test stand and software used for programming and interface purposes. Ridethrough PC equipment is not included in this Appendix.

Table A.1 Tension control system hardware information.

ITEM DESCRIPTION	**MANUFAC.**	**PART OR ID NO.**
Load cell tension transducer	Electromatic	FLN-14-50-6
Tension signal conditioner	Electromatic	TI-17A-800
Pulse tach transducer	Electro-Sensors	255, 906
Pulse tach signal conditioner	Electro-Sensors	SA420
Tachometer generators	Servo-Tech	SB740A-7
Discrete photo sensors	Banner	Q45BB6D
Profibus I/O base	Siemens	IM151-1
Analog input block	Siemens	2AI U
Discrete input block	Siemens	2DI DC24V
I/O power supply	Siemens	PM-E DC24V
24VDC power supply	Siemens	PS307 5A
24VDC power supply	Siemens	PS307 2A
S7-300 PLC/Profibus master	Siemens	CPU 315-2DP
ASI bus master	Siemens	CP 342-2
ASI power converter	Siemens	6EP1632-1AL01
ASI interface block	Siemens	ZU. NR. 17101
Micromaster Vector AC drive	Siemens	6SE3213-6CA40
Profibus drive interface	Siemens	CB15
1HP 3ph. duty master ind. motor	Reliance	S2000 - P56H0441P
Serial PC/PLC interface	Siemens	SVPL 0303891
Profibus/PC interface card	Applicom	PCI 1500S7

Table A.2 Programming and monitoring software.

SOFTWARE	**VERSION**
Microsoft Windows 98	Second Ed.
National Instruments LabView	5.1
Siemens Step 7	NA
National Instruments Applicom drivers	1.1
TMG-Itec Scope Profibus	5.1

Appendix B
Textile Tension Controller Mechanical Analysis

The mechanical portion of the tension control system consists of two drive reels and three idle reels (Figure B.1). For analysis, each reel is treated as a rolling mass system, and each connecting segment of textile is treated as a spring-damper system that exerts forces on the reels it connects to. Of critical importance in this analysis is the assumption that no idler reel slippage or web slack conditions occur in the system. If slack or slippage were to occur, the spring-damper and rolling mass systems would decouple, causing the system equations to become nonlinear. The constants and variables used in this analysis are graphically shown in Figure B.2, and listed in Table B.1. The analysis will determine the matrices for the state space relationship

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{B.1}$$

and

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{B.2}$$

where

$$\mathbf{u} = \begin{bmatrix} \tau_{app1} \\ \tau_{app5} \end{bmatrix} \tag{B.3}$$

and

$$\mathbf{y} = \begin{bmatrix} v_t \\ v_p \\ f_p \\ f_s \end{bmatrix}. \tag{B.4}$$

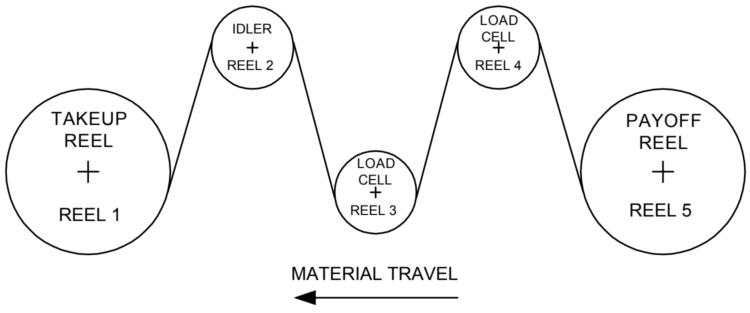


Figure B.1 Mechanical configuration of web tension control system.

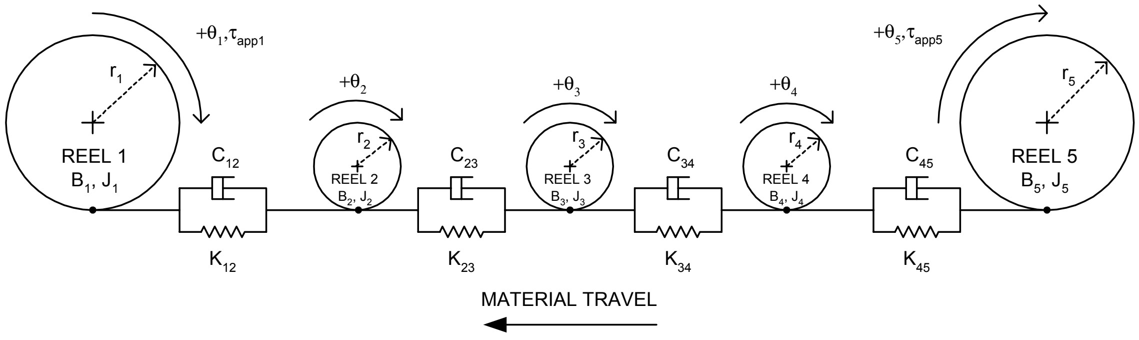


Figure B.2 Mechanical constants for web tension control system.

Table B.1 Tension control system mechanical constants.

CONSTANTS	**DESCRIPTION**	**UNITS**
$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$	reel angles	*unitless*
$\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5$	reel angular velocities	$1/\sec$
$\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3, \ddot{\theta}_4, \ddot{\theta}_5$	reel angular accelerations	$1/\sec^2$
τ_{app1}, τ_{app5}	drive reel applied torques	$N \cdot m$
B_1, B_2, B_3, B_4, B_5	reel rolling frictions	$(kg \cdot m^2)/\sec$
$C_{12}, C_{23}, C_{34}, C_{45}$	textile damping constants	$(N \cdot \sec)/m$
f_p, f_s	textile tensions	N
J_1, J_2, J_3, J_4, J_5	reel inertias	$kg \cdot m^2$
$K_{12}, K_{23}, K_{34}, K_{45}$	textile spring constants	N/m
r_1, r_2, r_3, r_4, r_5	reel radii	m
v_1, v_5	line speeds	$m/\sec$

Dynamic Analysis

We begin by evaluating the dynamic conditions for the takeup reel. Newton's Second Law is

$$Force = mass \cdot acceleration \tag{B.5}$$

and for a system in dynamic equilibrium [70],

$$\sum Forces = 0. \tag{B.6}$$

Since these are rolling mass systems, we sum the torques on each reel and set the sum equal to zero. Figure B.3 graphically shows all of the torques for the takeup reel (#1). The dynamic equation is therefore

$$0 = \tau_{app1} - B_1\dot{\theta}_1 - J_1\ddot{\theta}_1 - r_1K_{12}(r_1\theta_1 - r_2\theta_2) - r_1C_{12}(r_1\dot{\theta}_1 - r_2\dot{\theta}_2). \tag{B.7}$$

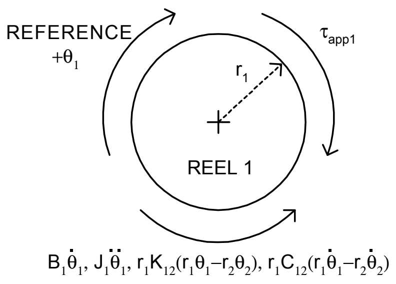

REFERENCE
$+\theta_1$
τ_{app1}
r_1
REEL 1
$B_1\dot{\theta}_1$, $J_1\ddot{\theta}_1$, $r_1K_{12}(r_1\theta_1-r_2\theta_2)$, $r_1C_{12}(r_1\dot{\theta}_1-r_2\dot{\theta}_2)$

Figure B.3 Free body diagram of forces on reel #1 (takeup).

The idler reels are similar in their form. Figure B.4 graphically shows the torques that contribute to the dynamic behavior for reel #2. Again we sum the torques to obtain

$$\begin{aligned} 0 &= r_2K_{12}(r_1\theta_1 - r_2\theta_2) + r_2C_{12}(r_1\dot{\theta}_1 - r_2\dot{\theta}_2) - B_2\dot{\theta}_2 - J_2\ddot{\theta}_2 \\ &\quad -r_2K_{23}(r_2\theta_2 - r_3\theta_3) - r_2C_{23}(r_2\dot{\theta}_2 - r_3\dot{\theta}_3). \end{aligned} \tag{B.8}$$

The equations for rollers #3 and #4 are the same as Equation B.8, with the exception of subscript differences, which gives

$$\begin{aligned} 0 &= r_3K_{23}(r_2\theta_2 - r_3\theta_3) + r_3C_{23}(r_2\dot{\theta}_2 - r_3\dot{\theta}_3) - B_3\dot{\theta}_3 - J_3\ddot{\theta}_3 \\ &\quad -r_3K_{34}(r_3\theta_3 - r_4\theta_4) - r_3C_{34}(r_3\dot{\theta}_3 - r_4\dot{\theta}_4) \end{aligned} \tag{B.9}$$

and

$$\begin{aligned} 0 &= r_4K_{34}(r_3\theta_3 - r_4\theta_4) + r_4C_{34}(r_3\dot{\theta}_3 - r_4\dot{\theta}_4) - B_4\dot{\theta}_4 - J_4\ddot{\theta}_4 \\ &\quad -r_4K_{45}(r_4\theta_4 - r_5\theta_5) - r_4C_{45}(r_4\dot{\theta}_4 - r_5\dot{\theta}_5). \end{aligned} \tag{B.10}$$

The payoff reel (#5) free body diagram is shown in Figure B.5. Summing these torques yields the equation

$$0 = r_5K_{45}(r_4\theta_4 - r_5\theta_5) + r_5C_{45}(r_4\dot{\theta}_4 - r_5\dot{\theta}_5) - B_5\dot{\theta}_5 - J_5\ddot{\theta}_5 + \tau_{app5}. \tag{B.11}$$

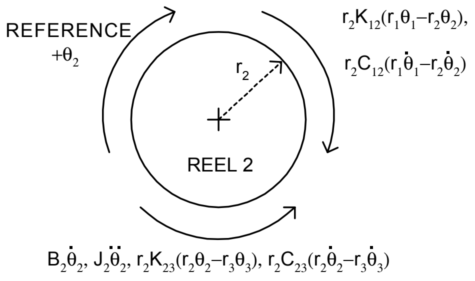

REFERENCE
$+\theta_2$
$r_2K_{12}(r_1\theta_1-r_2\theta_2)$,
$r_2C_{12}(r_1\dot{\theta}_1-r_2\dot{\theta}_2)$
r_2
REEL 2
$B_2\dot{\theta}_2$, $J_2\ddot{\theta}_2$, $r_2K_{23}(r_2\theta_2-r_3\theta_3)$, $r_2C_{23}(r_2\dot{\theta}_2-r_3\dot{\theta}_3)$

Figure B.4 Free body diagram of forces on reel #2 (idler).

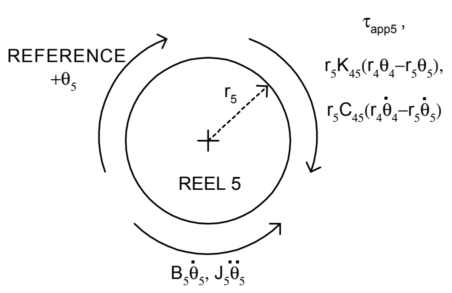

REFERENCE
$+\theta_5$
r_5
REEL 5
τ_{app5},
$r_5K_{45}(r_4\theta_4-r_5\theta_5)$,
$r_5C_{45}(r_4\dot{\theta}_4-r_5\dot{\theta}_5)$
$B_5\dot{\theta}_5$, $J_5\ddot{\theta}_5$

Figure B.5 Free body diagram of forces on reel #5 (payoff).

Forming the State Equations

The state equations are formed by rearranging Equations B.7 through B.11 for the highest order derivative, which yields

$$\begin{aligned}\ddot{\theta}_1 &= \left(\frac{-r_1^2K_{12}}{J_1}\right)\theta_1 + \left(\frac{-r_1^2C_{12}-B_1}{J_1}\right)\dot{\theta}_1 \\ &\quad + \left(\frac{r_1r_2K_{12}}{J_1}\right)\theta_2 + \left(\frac{r_1r_2C_{12}}{J_1}\right)\dot{\theta}_2 + \left(\frac{1}{J_1}\right)\tau_{app1}, \end{aligned} \tag{B.12}$$

$$\begin{aligned}\ddot{\theta}_2 &= \left(\frac{r_1r_2K_{12}}{J_2}\right)\theta_1 + \left(\frac{r_1r_2C_{12}}{J_2}\right)\dot{\theta}_1 + \left(\frac{-r_2^2K_{12}-r_2^2K_{23}}{J_2}\right)\theta_2 \\ &\quad + \left(\frac{-r_2^2C_{12}-r_2^2C_{23}-B_2}{J_2}\right)\dot{\theta}_2 + \left(\frac{r_2r_3K_{23}}{J_2}\right)\theta_3 + \left(\frac{r_2r_3C_{23}}{J_2}\right)\dot{\theta}_3, \end{aligned} \tag{B.13}$$

$$\begin{aligned}\ddot{\theta}_3 &= \left(\frac{r_2r_3K_{23}}{J_3}\right)\theta_2 + \left(\frac{r_2r_3C_{23}}{J_3}\right)\dot{\theta}_2 + \left(\frac{-r_3^2K_{23}-r_3^2K_{34}}{J_3}\right)\theta_3 \\ &\quad + \left(\frac{-r_3^2C_{23}-r_3^2C_{34}-B_3}{J_3}\right)\dot{\theta}_3 + \left(\frac{r_3r_4K_{34}}{J_3}\right)\theta_4 + \left(\frac{r_3r_4C_{34}}{J_3}\right)\dot{\theta}_4, \end{aligned} \tag{B.14}$$

$$\begin{aligned}\ddot{\theta}_4 &= \left(\frac{r_3r_4K_{34}}{J_4}\right)\theta_3 + \left(\frac{r_3r_4C_{34}}{J_4}\right)\dot{\theta}_3 + \left(\frac{-r_4^2K_{34}-r_4^2K_{45}}{J_4}\right)\theta_4 \\ &\quad + \left(\frac{-r_4^2C_{34}-r_4^2C_{45}-B_4}{J_4}\right)\dot{\theta}_4 + \left(\frac{r_4r_5K_{45}}{J_4}\right)\theta_5 + \left(\frac{r_4r_5C_{45}}{J_4}\right)\dot{\theta}_5, \end{aligned} \tag{B.15}$$

and

$$\begin{aligned}\ddot{\theta}_5 &= \left(\frac{r_4r_5K_{12}}{J_5}\right)\theta_4 + \left(\frac{r_4r_5C_{45}}{J_5}\right)\dot{\theta}_4 \\ &\quad + \left(\frac{-r_5^2K_{45}}{J_5}\right)\theta_5 + \left(\frac{-r_5^2C_{45}-B_5}{J_5}\right)\dot{\theta}_5 + \left(\frac{1}{J_5}\right)\tau_{app5}. \end{aligned} \tag{B.16}$$

These equations may be represented in matrix form, where the state vector is

$$\mathbf{x} = \left[\begin{array}{cccccccccc} \theta_1 & \dot{\theta}_1 & \theta_2 & \dot{\theta}_2 & \theta_3 & \dot{\theta}_3 & \theta_4 & \dot{\theta}_4 & \theta_5 & \dot{\theta}_5 \end{array} \right]^T, \tag{B.17}$$

the state derivative vector is

$$\dot{\mathbf{x}} = \left[\begin{array}{cccccccccc} \dot{\theta}_1 & \ddot{\theta}_1 & \dot{\theta}_2 & \ddot{\theta}_2 & \dot{\theta}_3 & \ddot{\theta}_3 & \dot{\theta}_4 & \ddot{\theta}_4 & \dot{\theta}_5 & \ddot{\theta}_5 \end{array} \right]^T, \tag{B.18}$$

and the input vector is

$$\mathbf{u} = \left[\begin{array}{c} \tau_{app1} \\ \tau_{app5} \end{array} \right]. \tag{B.19}$$

The state matrix is formed from the state variable coefficients of equations B.12 through B.16, and by recognizing the derivative equalities amongst the elements of Equations B.17 and B.18. Assembling this matrix gives us

$$\mathbf{A} = \left[\begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ \left(\frac{-r_1^2 K_{12}}{J_1}\right) & \left(\frac{-r_1^2 C_{12} - B_1}{J_1}\right) & \left(\frac{r_1 r_2 K_{12}}{J_1}\right) & \left(\frac{r_1 r_2 C_{12}}{J_1}\right) & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \left(\frac{r_1 r_2 K_{12}}{J_2}\right) & \left(\frac{r_1 r_2 C_{12}}{J_2}\right) & \left(\frac{-r_2^2 K_{12} - r_2^2 K_{23}}{J_2}\right) & \left(\frac{-r_2^2 C_{12} - r_2^2 C_{23} - B_2}{J_2}\right) & \left(\frac{r_2 r_3 K_{23}}{J_2}\right) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \left(\frac{r_2 r_3 K_{23}}{J_3}\right) & \left(\frac{r_2 r_3 C_{23}}{J_3}\right) & \left(\frac{-r_3^2 K_{23} - r_3^2 K_{34}}{J_3}\right) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{r_3 r_4 K_{34}}{J_4}\right) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right. \cdots$$

$$\cdots \left. \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \left(\frac{r_2 r_3 C_{23}}{J_2}\right) & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \left(\frac{-r_3^2 C_{23} - r_3^2 C_{34} - B_3}{J_3}\right) & \left(\frac{r_3 r_4 K_{34}}{J_3}\right) & \left(\frac{r_3 r_4 C_{34}}{J_3}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \left(\frac{r_3 r_4 C_{34}}{J_4}\right) & \left(\frac{-r_4^2 K_{34} - r_4^2 K_{45}}{J_4}\right) & \left(\frac{-r_4^2 C_{34} - r_4^2 C_{45} - B_4}{J_4}\right) & \left(\frac{r_4 r_5 K_{45}}{J_4}\right) & \left(\frac{r_4 r_5 C_{45}}{J_4}\right) \\ 0 & 0 & 0 & 0 & 1 \\ 0 & \left(\frac{r_4 r_5 K_{45}}{J_5}\right) & \left(\frac{r_4 r_5 C_{45}}{J_5}\right) & \left(\frac{-r_5^2 K_{45}}{J_5}\right) & \left(\frac{-r_5^2 C_{45} - B_5}{J_5}\right) \end{array} \right]. \tag{B.20}$$

The input matrix is formed from the applied torque coefficients of Equations A.12 and A.16. All other values are zero, which when assembled is

$$\mathbf{B} = \begin{bmatrix} 0 & \left(\frac{1}{J_1}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{1}{J_5}\right) \end{bmatrix}^T . \tag{B.21}$$

Next we form an expression to describe the process outputs of line speed and tension in state space terms. Each process output will be computed by two avenues to accurately reflect the redundant instrumentation found on the tension controller. The output matrix contains line speed measured from both payoff and takeup reels, and web tension measured by idler reels #3 and #4. Again, this matrix is

$$\mathbf{y} = \begin{bmatrix} v_t \\ v_p \\ f_p \\ f_s \end{bmatrix} . \tag{B.22}$$

Line speed may be expressed as the product of a reel's angular velocity times the radius. For the takeup and payoff reels, this is

$$v_t = r_1\dot{\theta}_1 \tag{B.23}$$

and

$$v_p = r_5\dot{\theta}_5 . \tag{B.24}$$

Each tension cell measures the average of the tension in the approaching and retreating web segments. In terms of the state variables, this is

$$f_p = \frac{[K_{23}(r_2\theta_2 - r_3\theta_3) + C_{23}(r_2\dot{\theta}_2 - r_3\dot{\theta}_3)] + [K_{34}(r_3\theta_3 - r_4\theta_4) + C_{34}(r_3\dot{\theta}_3 - r_4\dot{\theta}_4)]}{2} \tag{B.25}$$

and

$$f_s = \frac{[K_{34}(r_3\theta_3 - r_4\theta_4) + C_{34}(r_3\dot{\theta}_3 - r_4\dot{\theta}_4)] + [K_{45}(r_4\theta_4 - r_5\theta_5) + C_{45}(r_4\dot{\theta}_4 - r_5\dot{\theta}_5)]}{2} \tag{B.26}$$

Again we combine state variables, giving us

$$\begin{aligned} f_p \;=\; & \left(\frac{r_2K_{23}}{2}\right)\theta_2 + \left(\frac{r_2C_{23}}{2}\right)\dot{\theta}_2 + \left(\frac{-r_3K_{23} + r_3K_{34}}{2}\right)\theta_3 \\ & + \left(\frac{-r_3C_{23} + r_3C_{34}}{2}\right)\dot{\theta}_3 + \left(\frac{-r_4K_{34}}{2}\right)\theta_4 + \left(\frac{-r_4C_{34}}{2}\right)\dot{\theta}_4 \end{aligned} \tag{B.27}$$

and

$$f_s = \left(\frac{r_3 K_{34}}{2}\right)\theta_3 + \left(\frac{r_3 C_{34}}{2}\right)\dot{\theta}_3 + \left(\frac{-r_4 K_{34} + r_4 K_{45}}{2}\right)\theta_4 + \left(\frac{-r_4 C_{34} + r_4 C_{45}}{2}\right)\dot{\theta}_4 + \left(\frac{-r_5 K_{45}}{2}\right)\theta_5 + \left(\frac{-r_5 C_{45}}{2}\right)\dot{\theta}_5. \tag{B.28}$$

Finally, Equations B.23, B.24, B.27, and B.28 are combined into the matrix form

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{B.29}$$

where

$$\mathbf{C} = \begin{bmatrix} 0 & r_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_5 \\ 0 & 0 & \left(\frac{r_2 K_{23}}{2}\right) & \left(\frac{r_2 C_{23}}{2}\right) & \left(\frac{-r_3 K_{23}+r_3 K_{34}}{2}\right) & \left(\frac{-r_3 C_{23}+r_3 C_{34}}{2}\right) & \left(\frac{-r_4 K_{34}}{2}\right) & \left(\frac{-r_4 C_{34}}{2}\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{r_3 K_{34}}{2}\right) & \left(\frac{r_3 C_{34}}{2}\right) & \left(\frac{-r_4 K_{34}+r_4 K_{45}}{2}\right) & \left(\frac{-r_4 C_{34}+r_4 C_{45}}{2}\right) & \left(\frac{-r_5 K_{45}}{2}\right) & \left(\frac{-r_5 C_{45}}{2}\right) \end{bmatrix}, \tag{B.30}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{B.31}$$

and the matrices **y**, **x**, and **u** are defined in Equations B.22, B.17, and B.19, respectively.

The state matrices that include values of r_1 and r_5 are time variant, since r_1 and r_5 vary throughout the unwinding and winding process. For the purpose of our analysis, however, their values are considered to change negligibly during the timespan of sag simulation. Therefore they are held at fixed values during simulation, causing the state space model to remain linear and time invariant. Experimentation with the tension control stand accounts for this as well. Voltage sags are applied during the same point in the wind/unwind process, keeping the payoff and takeup radii consistent for all tests.

Appendix C
Determination of Physical Constants

This appendix details the procedures used to determine the physical constants in the textile control stand.

Reel Radii

Simple calipers were used to measure the reel diameters. Table C.1 shows the resulting radii measurements.

Table C.1 Measured roller radii.

CONSTANT	**DESCRIPTION**	**VALUE**
r_1	takeup reel radius with accumulated web	0.072 m
r_{1base}	takeup reel core radius	0.038 m
r_2	idler reel radius	0.038 m
r_3	primary tension cell radius	0.044 m
r_4	secondary tension cell radius	0.044 m
r_5	payoff reel radius with accumulated web	0.072 m
r_{5base}	payoff reel core radius	0.038 m

Reel Inertia

To measure reel inertia, we first refer to the standard motor-load rotating system equation

$$\tau_{applied} = J\frac{d\omega}{dt} + B\omega + \tau_{load}. \tag{C.1}$$

We may experimentally isolate the inertia component from the system by removing the load torque, τ_{load}, and by experimenting at low rotational speeds to keep the rolling friction component, $B\omega$, negligible. This gives us

$$\tau_{applied} = J\frac{d\omega}{dt}. \tag{C.2}$$

By using calibrated weights to apply a constant torque of known value, we may measure the rate of change of the system's rotational speed. With these values known, the value of J is easily solved for.

The final value of J is determined by assembling the results of n test runs into a set of matrices in the form

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{C.3}$$

where

$$\mathbf{y} = \begin{bmatrix} \tau_{applied,1} \\ \tau_{applied,2} \\ \tau_{applied,3} \\ \vdots \\ \tau_{applied,n} \end{bmatrix}, \tag{C.4}$$

$$\mathbf{A} = \begin{bmatrix} \frac{d\omega}{dt}_{,1} \\ \frac{d\omega}{dt}_{,2} \\ \frac{d\omega}{dt}_{,3} \\ \vdots \\ \frac{d\omega}{dt}_{,n} \end{bmatrix}, \tag{C.5}$$

and

$$\mathbf{x} = [J]\,. \tag{C.6}$$

The value of $\mathbf{x} = [J]$ is found using an overdetermined case solution, as outlined in [71]. This solution is found with the equation

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{C.7}$$

The idler and tension reels do not have speed measurement instruments directly attached to them, and may be referred to as being 'unmeasurable'. Therefore, to determine the inertia of an unmeasurable reel, a belt was attached between the unmeasurable reel and the nearest measurable (takeup or payoff) reel. Constant torques were again applied to the measurable reel, and the two-roller system inertia was calculated using the overdetermined case solution. Since the measurable reel inertia is known from previous experiments, the unmeasurable reel inertia is the only unknown quantity in the expression

$$J_{2rollersystem} = J_{measurable} + J_{unmeasurable}\left(\frac{r_{measurable}}{r_{unmeasurable}}\right)^2, \tag{C.8}$$

which is rearranged to give

$$J_{unmeasurable} = \frac{J_{2rollersystem} - J_{measurable}}{\left(\frac{r_{measurable}}{r_{unmeasurable}}\right)^2}. \tag{C.9}$$

Table C.2 summarizes the results obtained from these procedures.

Table C.2 Bare roller inertias.

CONSTANT	**DESCRIPTION**	**VALUE**
J_{1base}	takeup reel inertia without accumulated web	0.0140 $kg \cdot m^2$
J_2	idler reel inertia	0.0040 $kg \cdot m^2$
J_3	primary tension cell inertia	0.0082 $kg \cdot m^2$
J_4	secondary tension cell inertia	0.0089 $kg \cdot m^2$
J_{5base}	payoff reel inertia without accumulated web	0.0126 $kg \cdot m^2$

Additive Inertia of Coiled Web

Added to the base inertia of the payoff and takeup reels is the inertia of the coiled web. This may be expressed as a function of the reel radius, including the accumulated web. The inertia of the accumulated web is the difference between the inertia of a coiled web of radius r_{web} and the inertia of the theoretical center of the coiled web whose space is take up by the reel core. This is expressed by

$$J_{accumulatedweb,r=r_{web}} = J_{coiledweb,r=r_{web}} - J_{coiledweb,r=r_{base}}. \tag{C.10}$$

To determine an expression for Equation C.10 in terms of radius, we begin with the mass, m of a tightly coiled web, which is

$$m = \rho \pi r_{web}^2 w, \tag{C.11}$$

where the density ρ, and width w are

$$\rho = 216.6 \frac{kg}{m^3} \tag{C.12}$$

and

$$w = 0.185\ m \tag{C.13}$$

respectively. Using these values, we calculate the inertia of the theoretical center to be

$$J_{coiledweb,r=r_{base}} = \frac{1}{2}mr^2 = \frac{1}{2}\rho\pi r_{base}^2 w r_{base}^2 = 0.000262\ kg \cdot m^2 \tag{C.14}$$

Similarly, the mass of the solid web cylinder of radius r_{web} is

$$J_{coiledweb,r=r_{web}} = \frac{1}{2}mr^2 = \frac{1}{2}\rho\pi r_{web}^2 w r_{web}^2 = 62.943 r_{web}^4\ kg \cdot m^2 \tag{C.15}$$

Substituting values from C.14 and C.15 into C.10 yields

$$J_{accumulatedweb,r=r_{web}} = (62.943 r_{web}^4 - 0.000262)\ kg \cdot m^2 \tag{C.16}$$

Equation C.16 is evaluated for r_1, then added to J_{1base} to obtain the total takeup reel inertia, J_1. The same operation is performed for the payoff reel to determine J_5.

Rolling Friction

Having calculated the inertia J for each reel, we again refer to Equation C.1 for the rolling mass system. For higher speeds the component $B\omega$ cannot be neglected. To determine B, we observe the deceleration characteristics from an appreciable speed under no load and no applied torque conditions. The time response takes on the form

$$\omega(t) = \omega_0 e^{-\frac{B}{J}t} \tag{C.17}$$

where the value B/J is the inverse of the decay time constant, and ω_0 is the initial speed. The time constant τ was measured by curve fitting the deceleration response, and B was calculated using

$$B = \frac{J}{\tau}. \tag{C.18}$$

For the idler and tension reels, a belt was again attached between an 'unmeasurable' roller and a roller with measurable speed. The two reel system's rolling friction constant was determined, and the unknown rolling friction constant was calculated with

$$B_{unmeasurable} = \frac{B_{2rollersystem} - B_{measurable}}{\left(\frac{r_{measurable}}{r_{unmeasurable}}\right)^2} \tag{C.19}$$

which is similar in form to Equation C.9. Table C.3 shows the measured rolling friction constants.

Table C.3 Rolling friction constants.

CONSTANT	**DESCRIPTION**	**VALUE**
B_1	takeup reel rolling friction	0.0018 $(kg \cdot m^2)/\sec$
B_2	idler reel rolling friction	0.0008 $(kg \cdot m^2)/\sec$
B_3	primary tension cell rolling friction	0.0022 $(kg \cdot m^2)/\sec$
B_4	secondary tension cell rolling friction	0.0023 $(kg \cdot m^2)/\sec$
B_5	payoff reel rolling friction	0.0015 $(kg \cdot m^2)/\sec$

Textile Constants

The textile spring and damper constants were measured using an Instron tensile test machine in accordance with ASTM 5035 [72]. The tensile test machine grips onto a segment of textile and slowly pulls it apart at a constant crosshead speed. During this operation, the force exerted by the textile on the grips is measured and recorded as a force vs. elongation curve. From this curve, the spring constant K can be determined as the rate of change of force per unit elongation. The value of the spring constant was determined for several test runs at varying crosshead speeds, and the results averaged.

The damper constant, however, was not clearly evident. If an appreciable damper constant were present, then an appreciable initial force would be exerted at the time of initial extension. This was not the case. Since the damper constant offers a negligible contribution to the force exerted by the textile, it would seem reasonable to neglect it in modeling and simulation. If it is neglected, then a high frequency component exists in tension waveforms during simulation. Therefore, the damper constant C was assigned a value of 1% of the spring constant, which is negligible when compared to the spring constant, but nonzero to damp high frequency tension oscillations in simulation. Table C.4 lists the textile spring and damper constants.

Table C.4 Textile constants.

CONSTANT	**DESCRIPTION**	**VALUE**
$K_{12} = K_{23} = K_{34} = K_{45}$	textile spring constants	79010 N/m
$C_{12} = C_{23} = C_{34} = C_{45}$	textile damper constants	790.10 $N/(m \cdot \sec)$

First Order Device Responses

The first order input-output relationships were determined for the analog instrumentation, A/D converter, and drive frequency setpoint. These relationships are described in Equations 2.1 through 2.3. For each device, a series of steady state outputs were determined for corresponding input, and formed into the matrices

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{C.20}$$

where

$$\mathbf{y} = \begin{bmatrix} output_1 \\ output_2 \\ output_3 \\ \vdots \\ output_n \end{bmatrix}, \tag{C.21}$$

$$\mathbf{A} = \begin{bmatrix} input_1 & 1 \\ input_2 & 1 \\ input_3 & 1 \\ \vdots & \vdots \\ input_n & 1 \end{bmatrix}, \tag{C.22}$$

and

$$\mathbf{x} = \begin{bmatrix} K_{device} \\ B_{device} \end{bmatrix}. \tag{C.23}$$

The overdetermined case solution in Equation C.7 was then calculated for each device to yield the values for the device gain, K and offset, B. Device time constants were determined from manufacturer step response or construction specifications. Table C.5 lists the first order gain, time constant, and offset results.

Table C.5 First order device constants.

CONSTANT	**DESCRIPTION**	**VALUE**
K_{tach1}	tachometer 1 gain	0.1966 $V/(rad/\sec)$
τ_{tach1}	tachometer 1 time constant	0.0266 sec
B_{tach1}	tachometer 1 offset	-0.0804 V
K_{tach5}	tachometer 5 gain	0.1918 $V/(rad/\sec)$
τ_{tach5}	tachometer 5 time constant	0.0266 sec
B_{tach5}	tachometer 5 offset	0.0216 V
K_{btach1}	backup tachometer 1 gain	0.0250 $V/(rad/\sec)$
τ_{btach1}	backup tachometer 1 time constant	0.0006 sec
B_{btach1}	backup tachometer 1 offset	-0.0016 V
K_{btach5}	backup tachometer 5 gain	0.0253 $V/(rad/\sec)$
τ_{btach5}	backup tachometer 5 time constant	0.0006 sec
B_{btach5}	backup tachometer 5 offset	-0.0220 V
K_{pten}	primary tension cell gain	0.0989 V/N
τ_{pten}	primary tension cell time constant	0.0028 sec
B_{pten}	primary tension cell offset	0.0746 V
K_{sten}	secondary tension cell gain	0.1023 V/N
τ_{sten}	secondary tension cell time constant	0.0028 sec
B_{sten}	secondary tension cell offset	0.2829 V
K_{AD}	A/D converter gain	2765.1 $bits/V$
τ_{AD}	A/D converter time constant	0.120 sec
B_{AD}	A/D converter offset	-4.4997 $bits$
K_{DR1}	drive 1 setpoint gain	0.01943 $(rad/\sec)/bit$
τ_{DR1}	drive 1 setpoint time constant	0.015 sec
B_{DR1}	drive 1 setpoint offset	0 $(rad/\sec)$
K_{DR5}	drive 5 setpoint gain	0.01943 $(rad/\sec)/bit$
τ_{DR5}	drive 5 setpoint time constant	0.015 sec
B_{DR5}	drive 5 setpoint offset	0 $(rad/\sec)$

Induction Motor Tests

To determine their equivalent circuit parameters, the induction motors were tested with a DC stator resistance test, no-load test, and locked-rotor test, as described in [73]. The determined equivalent circuit parameters are listed in Table C.6.

Table C.6 Induction motor equivalent circuit parameters.

CONSTANT	**DESCRIPTION**	**VALUE**
R_{s1}	stator resistance, motor 1	3.02 Ω
R_{r1}	rotor resistance, motor 1	1.88 Ω
L_{ls1}	stator leakage inductance, motor 1	0.006477 H
L_{lr1}	rotor leakage inductance, motor 1	0.009713 H
L_{m1}	magnetizing inductance, motor 1	0.181 H
$L_{s1} = L_{ls1} + L_{m1}$	total stator inductance, motor 1	0.187477 H
$L_{r1} = L_{lr1} + L_{m1}$	total rotor inductance, motor 1	0.190713 H
R_{s5}	stator resistance, motor 5	2.85 Ω
R_{r5}	rotor resistance, motor 5	1.92 Ω
L_{ls5}	stator leakage inductance, motor 5	0.007207 H
L_{lr5}	rotor leakage inductance, motor 5	0.010810 H
L_{m5}	magnetizing inductance, motor 5	0.181 H
$L_{s5} = L_{ls5} + L_{m5}$	total stator inductance, motor 5	0.188207 H
$L_{r5} = L_{lr5} + L_{m5}$	total rotor inductance, motor 5	0.191810 H

V/Hz Drive Behavior

The V/Hz output relationship of the motor drives may be modeled in two segments by the functions

$$V_{out} = V_{ebase} e^{m_e f}, \ (f < 6 \ Hz) \tag{C.24}$$

$$V_{out} = m_l f + V_{offset}, \ (f \geq 6 \ Hz) \tag{C.25}$$

which accounts for the low frequency voltage boost, followed by linear V/Hz output. The relationship of output voltage versus output frequency was measured and plotted for each drive, and the resulting curves fitted to the model in Equation C.24 using the values listed in Table C.7.

Table C.7 Volts/Hertz drive constants.

CONSTANT	**DESCRIPTION**	**VALUE**
V_{base1}	low freq. base voltage, drive 1	26.0 V_{RMS}
m_{e1}	low freq. exponential multiplier, drive 1	0.0541 $1/Hz$
$V_{offset1}$	linear segment voltage offset, drive 1	3.4799 V_{RMS}
m_{l1}	linear segment V/Hz gain, drive 1	15.0871 V/Hz
V_{base5}	low freq. base voltage, drive 5	22.5 V_{RMS}
m_{e5}	low freq. exponential multiplier, drive 5	0.0728 $1/Hz$
$V_{offset5}$	linear segment voltage offset, drive 5	3.3971 V_{RMS}
m_{l5}	linear segment V/Hz gain, drive 5	14.4466 V/Hz

Appendix D

Matlab Process Response Simulation Code

This appendix lists the Matlab code used to simulate the textile tension controller's behavior in the presence of voltage sags.

```
%-------------------------------------------------------------------%
%sagsim.m
%
%Simulates the output response of a textile tension controller when
%subjected to voltage sags.
%
%Written by Owen Parks
%Clemson University PQIA Laboratories
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%initialize memory
clear all;
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%load sag data
load instsagdata;%101000;

%sag recovery data
tau_rec=rectime/5;
tau_rec2=rectime2/5;
tau_rec3=rectime3/5;
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%define time parameters

%evaluation points/sec
ptspersec=10000;

%simulation duration
numsecs=3;

%total calculation points (+1 for t=0)
numpts=(ptspersec*numsecs)+1;

%time array
time=linspace(0,numsecs,numpts);

%time differential
dt=time(3)-time(2);

%sag inception time
t_sagstart=0.5;
```

```
%sag extinction time
t_sagend=0.5+0.050;

%dt for sampled waveform used in sag simulation
dt2=0.1/250;
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%define measured physical constants

%radii in meters
r1=0.072;
r2=0.038;
r3=0.044;
r4=0.044;
r5=0.072;
r1base=0.038;
r5base=0.038;

%additive inertia due to accumulated web
J1web=62.943*(r1^4)-0.000262;
J5web=62.943*(r5^4)-0.000262;

%inertia in kg*m^2
J1=0.0140+J1web;
J2=0.0040;
J3=0.0082;
J4=0.0089;
J5=0.0126+J5web;

%rolling friction in (kg*m^2)/sec
B1=0.0018;
B2=0.0008;
B3=0.0022;
B4=0.0023;
B5=0.0015;

%textile spring constant in N/m
K12=79010;
K23=79010;
K34=79010;
K45=79010;

%textile damping constant in N*sec/m
C12=K12*0.01;
C23=K23*0.01;
C34=K34*0.01;
C45=K45*0.01;
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%assemble state space matrices
```

```
%state matrix
A(1,:)=[0 1 0 0 0 0 0 0 0 0];
A(2,1)=(-r1*r1*K12)/(J1);
A(2,2)=(-r1*r1*C12-B1)/(J1);
A(2,3)=(r1*r2*K12)/(J1);
A(2,4)=(r1*r2*C12)/(J1);
A(2,5)=0;
A(2,6)=0;
A(2,7)=0;
A(2,8)=0;
A(2,9)=0;
A(2,10)=0;
A(3,:)=[0 0 0 1 0 0 0 0 0 0];
A(4,1)=(r1*r2*K12)/(J2);
A(4,2)=(r1*r2*C12)/(J2);
A(4,3)=(-r2*r2*K12-r2*r2*K23)/(J2);
A(4,4)=(-r2*r2*C12-r2*r2*C23-B2)/(J2);
A(4,5)=(r2*r3*K23)/(J2);
A(4,6)=(r2*r3*C23)/(J2);
A(4,7)=0;
A(4,8)=0;
A(4,9)=0;
A(4,10)=0;
A(5,:)=[0 0 0 0 0 1 0 0 0 0];
A(6,1)=0;
A(6,2)=0;
A(6,3)=(r2*r3*K23)/(J3);
A(6,4)=(r2*r3*C23)/(J3);
A(6,5)=(-r3*r3*K23-r3*r3*K34)/(J3);
A(6,6)=(-r3*r3*C23-r3*r3*C34-B3)/(J3);
A(6,7)=(r3*r4*K34)/(J3);
A(6,8)=(r3*r4*C34)/(J3);
A(6,9)=0;
A(6,10)=0;
A(7,:)=[0 0 0 0 0 0 0 1 0 0];
A(8,1)=0;
A(8,2)=0;
A(8,3)=0;
A(8,4)=0;
A(8,5)=(r3*r4*K34)/(J4);
A(8,6)=(r3*r4*C34)/(J4);
A(8,7)=(-r4*r4*K34-r4*r4*K45)/(J4);
A(8,8)=(-r4*r4*C34-r4*r4*C45-B4)/(J4);
A(8,9)=(r4*r5*K45)/(J4);
A(8,10)=(r4*r5*C45)/(J4);
A(9,:)=[0 0 0 0 0 0 0 0 0 1];
A(10,1)=0;
A(10,2)=0;
A(10,3)=0;
A(10,4)=0;
A(10,5)=0;
A(10,6)=0;
A(10,7)=(r4*r5*K45)/(J5);
A(10,8)=(r4*r5*C45)/(J5);
```

```
A(10,9)=(-r5*r5*K45)/(J5);
A(10,10)=(-r5*r5*C45-B5)/(J5);

%input matrix
B=[0 0; (1/J1) 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 (1/J5)];

%output matrix
C(1,:)=[0 r1 0 0 0 0 0 0 0 0];
C(2,:)=[0 0 0 0 0 0 0 0 0 r5];
C(3,1)=0;
C(3,2)=0;
C(3,3)=(r2*K23)/(2);
C(3,4)=(r2*C23)/(2);
C(3,5)=(-r3*K23+r3*K34)/(2);
C(3,6)=(-r3*C23+r3*C34)/(2);
C(3,7)=(-r4*K34)/(2);
C(3,8)=(-r4*C34)/(2);
C(3,9)=0;
C(3,10)=0;
C(4,1)=0;
C(4,2)=0;
C(4,3)=0;
C(4,4)=0;
C(4,5)=(r3*K34)/(2);
C(4,6)=(r3*C34)/(2);
C(4,7)=(-r4*K34+r4*K45)/(2);
C(4,8)=(-r4*C34+r4*C45)/(2);
C(4,9)=(-r5*K45)/(2);
C(4,10)=(-r5*C45)/(2);

%feedthrough matrix
D=zeros(4,2);
%----------------------------------------------------------------------%

%----------------------------------------------------------------------%
%define control constants

%proportional, integral, and state gain for linespeed
P_linespeed=0.3025;
I_linespeed=1/0.566;
K_state_linespeed=0.005;

%proportional, integral, and state gain for tension
P_tension=0.3;
I_tension=1/0.090;
K_state_tension=0.02;
%----------------------------------------------------------------------%

%----------------------------------------------------------------------%
%setpoints for speed and tension - held constant

%linespeed set in m/s
linespeed_set=1.0;
```

```
%tension set in N
tension_set=30;
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%define gain and time constant for first order hardware

%takeup speed sensor
K_tach1=0.1966;
tau_tach1=0.0266;
offset_tach1=-0.0804;

%payoff speed sensor
K_tach5=0.1918;
tau_tach5=0.0266;
offset_tach5=0.0216;

%backup takeup speed sensor
K_btach1=0.0250;
tau_btach1=0.0006;
offset_btach1=-0.0016;

%backup payoff speed sensor
K_btach5=0.0253;
tau_btach5=0.0006;
offset_btach5=0.0220;

%primary tension sensor
K_pten=0.0989;
tau_pten=0.0028;
offset_pten=0.0746;

%secondary tension sensor
K_sten=0.1023;
tau_sten=0.0028;
offset_sten=0.2829;

%input A/D block
K_ad=2765.1;
tau_ad=0.120;
offset_ad=-4.4997;

%drive 1 output frequency - convert to rad/sec, not Hz
K_DR1=0.01943;
tau_DR1=0.015;
%no drive 1 output offset

%drive 5 output frequency - convert to rad/sec, not Hz
K_DR5=0.01943;
tau_DR5=0.015;
%no drive 5 output offset
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
```

```
%define AC induction motor constants

%motor 1
Rs1=2.77;
Rr1=1.999;
Lls1=0.00617;
Llr1=0.00925;
Lm1=0.1731;
Ls1=Lls1+Lm1;
Lr1=Llr1+Lm1;
Lmat1=[(Lls1+Lm1) 0 Lm1 0; 0 (Lls1+Lm1) 0 Lm1;...
        Lm1 0 (Llr1+Lm1) 0; 0 Lm1 0 (Llr1+Lm1)];

%motor 5
Rs5=2.756;
Rr5=1.831;
Lls5=0.006903;
Llr5=0.010354;
Lm5=0.17070;
Ls5=Lls5+Lm5;
Lr5=Llr5+Lm5;
Lmat5=[(Lls5+Lm5) 0 Lm5 0; 0 (Lls5+Lm5) 0 Lm5;...
        Lm5 0 (Llr5+Lm5) 0; 0 Lm5 0 (Llr5+Lm5)];
%---------------------------------------------------------------------%

%---------------------------------------------------------------------%
%define constants for motor drives

%volts per hertz constants >= 6Hz
VpHzgain1=3.4799;
VpHzgain5=3.3971;
VpHzoffset1=15.0871;
VpHzoffset5=14.4466;

%volts per hertz constants < 6Hz
VpHzegain1=26.0;
VpHzegain5=22.5;
VpHzexp1=log((VpHzgain1*6+VpHzoffset1)/VpHzegain1)/6;
VpHzexp5=log((VpHzgain5*6+VpHzoffset5)/VpHzegain5)/6;

%rotor voltages - shorted for squirrel cage induction motor
vqr1=0;
vdr1=0;
vqr5=0;
vdr5=0;
%---------------------------------------------------------------------%

%---------------------------------------------------------------------%
%initialize integration sums and output signals at t=0

%initialize linespeed and tension outputs
initialspeed=linespeed_set;
initialtension=tension_set;
```

```
%assign array inital values
linespeed(1)=initialspeed;
pten_actual(1)=initialtension;

%calculate initial matrix states - x and u
xdot=[linespeed(1)/r1; 0; linespeed(1)/r2; 0; linespeed(1)/r3; 0;...
        linespeed(1)/r4; 0; linespeed(1)/r5; 0];
x=[0; linespeed(1)/r1; 0; linespeed(1)/r2; 0; linespeed(1)/r3;...
        0; linespeed(1)/r4; 0; linespeed(1)/r5];

%calculate angles theta1-theta5
findthetamatrix=zeros(5,5);
findthetamatrix(1,2)=r2*K23;
findthetamatrix(1,3)=-r3*K23+r3*K34;
findthetamatrix(1,4)=-r4*K34;
findthetamatrix(2,1)=r1*r2*K12;
findthetamatrix(2,2)=-r2*r2*K12-r2*r2*K23;
findthetamatrix(2,3)=r2*r3*K23;
findthetamatrix(3,2)=r2*r3*K23;
findthetamatrix(3,3)=-r3*r3*K23-r3*r3*K34;
findthetamatrix(3,4)=r3*r4*K34;
findthetamatrix(4,3)=r3*r4*K34;
findthetamatrix(4,4)=-r4*r4*K34-r4*r4*K45;
findthetamatrix(4,5)=r4*r5*K45;
findthetamatrix(5,3)=1;
findthetavalues=[2*initialtension; B2*x(4); B3*x(6); B4*x(8); 10*pi];
findthetas=inv(findthetamatrix)*findthetavalues;

x(1)=findthetas(1);
x(3)=findthetas(2);
x(5)=findthetas(3);
x(7)=findthetas(4);
x(9)=findthetas(5);

x_intsum=x;

initialtorques=(xdot-(A*x));
torque_1(1)=initialtorques(2)*J1;
torque_5(1)=initialtorques(10)*J5;

yinit=C*x;

%instrument output signals at t=0; w=1
thetadot_1(1)=linespeed_set/r1;
out_tach1(1)=thetadot_1(1)*K_tach1+offset_tach1;
out_tach1B(1)=thetadot_1(1)*K_tach1+offset_tach1;
thetadot_5(1)=linespeed_set/r5;
out_tach5(1)=thetadot_5(1)*K_tach5+offset_tach5;
out_tach5B(1)=thetadot_5(1)*K_tach5+offset_tach5;
out_btach1(1)=thetadot_1(1)*K_btach1+offset_btach1;
out_btach5(1)=thetadot_5(1)*K_btach5+offset_btach5;
pten_actual(1)=yinit(3);
out_pten(1)=pten_actual(1)*K_pten+offset_pten;
out_pten2(1)=out_pten(1);
```

```
sten_actual(1)=yinit(4);
out_sten(1)=sten_actual(1)*K_sten+offset_sten;
out_ad1(1)=out_tach1(1)*K_ad+offset_ad;
out_ad2(1)=out_tach5(1)*K_ad+offset_ad;
out_ad3(1)=out_pten(1)*K_ad+offset_ad;
out_ad1B(1)=out_ad1(1);
out_ad2B(1)=out_ad2(1);
out_ad3B(1)=out_ad3(1);
out_ad1A(1)=out_ad1(1);
out_ad2A(1)=out_ad2(1);
out_ad3A(1)=out_ad3(1);

%find linespeed integral control initial value
%iterative application of dq theory required

%search above thetadot for motoring
sweeppts=100000;
omegaesweep=linspace(2*thetadot_1(1),2*thetadot_1(1)+20,sweeppts);
q=1;

for z=1:sweeppts
    if abs(omegaesweep(z))<(6*2*pi)
        llvolts=VpHzegain1*exp(VpHzexp1*(omegaesweep(z)/(2*pi)));
        voltmag=llvolts/sqrt(3);
    elseif abs(omegaesweep(z))>=(6*2*pi)
        llvolts=VpHzgain1*(omegaesweep(z)/(2*pi))+VpHzoffset1;
        voltmag=llvolts/sqrt(3);
    end

    vas=voltmag*sqrt(2)*cos(0);
    vbs=voltmag*sqrt(2)*cos(0-2*pi/3);
    vcs=voltmag*sqrt(2)*cos(0+2*pi/3);

    vqss=(2/3)*vas-(1/3)*vbs-(1/3)*vcs;
    vdss=(-1/sqrt(3))*vbs+(1/sqrt(3))*vcs;

    vqs=vqss*cos(0)-vdss*sin(0);
    vds=vqss*sin(0)+vdss*cos(0);

    vmat=[vqs; vds; 0; 0];
    initmat=[Rs1 omegaesweep(z)*Ls1 0 omegaesweep(z)*Lm1;...
            -1*omegaesweep(z)*Ls1 Rs1 -1*omegaesweep(z)*Lm1 0;...
            0 (omegaesweep(z)-2*thetadot_1(1))*Lm1 ...
            Rr1 (omegaesweep(z)-2*thetadot_1(1))*Lr1;...
            -1*(omegaesweep(z)-2*thetadot_1(1))*Lm1 0 ...
            -1*(omegaesweep(z)-2*thetadot_1(1))*Lr1 Rr1];
    i_init=inv(initmat)*vmat;
    iqs=i_init(1);
    ids=i_init(2);
    iqr=i_init(3);
    idr=i_init(4);
    testtorque=3*Lm1*(iqs*idr-ids*iqr);

    if abs(testtorque-torque_1(1))<0.001
```

```
        initialomega1(q)=omegaesweep(z);
        q=q+1;
    end

end

%electrical frequency that yields the desired speed-torque point
omegaefound=mean(initialomega1);

%back calculate to integral output
intsum_Ilinespeed=(((omegaefound/(K_DR1*276.4))...
    -K_state_linespeed*r5*yinit(3))/P_linespeed)/I_linespeed;

%find tension integral control initial value
%iterative application of dq theory required

%search below thetadot for braking
sweeppts=100000;
omegaesweep=linspace(2*thetadot_5(1),2*thetadot_5(1)-20,sweeppts);
q=1;

for z=1:sweeppts
    if abs(omegaesweep(z))<(6*2*pi)
        llvolts=VpHzegain5*exp(VpHzexp5*(omegaesweep(z)/(2*pi)));
        voltmag=llvolts/sqrt(3);
    elseif abs(omegaesweep(z))>=(6*2*pi)
        llvolts=VpHzgain5*(omegaesweep(z)/(2*pi))+VpHzoffset5;
        voltmag=llvolts/sqrt(3);
    end

    vas=voltmag*sqrt(2)*cos(0);
    vbs=voltmag*sqrt(2)*cos(0-2*pi/3);
    vcs=voltmag*sqrt(2)*cos(0+2*pi/3);

    vqss=(2/3)*vas-(1/3)*vbs-(1/3)*vcs;
    vdss=(-1/sqrt(3))*vbs+(1/sqrt(3))*vcs;

    vqs=vqss*cos(0)-vdss*sin(0);
    vds=vqss*sin(0)+vdss*cos(0);

    vmat=[vqs; vds; 0; 0];
    initmat=[Rs5 omegaesweep(z)*Ls5 0 omegaesweep(z)*Lm5;...
            -1*omegaesweep(z)*Ls5 Rs5 -1*omegaesweep(z)*Lm5 0;...
            0 (omegaesweep(z)-2*thetadot_5(1))*Lm5 ...
            Rr5 (omegaesweep(z)-2*thetadot_5(1))*Lr5;...
            -1*(omegaesweep(z)-2*thetadot_5(1))*Lm5 0 ...
            -1*(omegaesweep(z)-2*thetadot_5(1))*Lr5 Rr5];
    i_init=inv(initmat)*vmat;
    iqs=i_init(1);
    ids=i_init(2);
    iqr=i_init(3);
    idr=i_init(4);
    testtorque=3*Lm5*(iqs*idr-ids*iqr);
    if abs(testtorque-torque_5(1))<0.001
        initialomega5(q)=omegaesweep(z);
        q=q+1;
    end

end

%electrical frequency that yields the desired speed-torque point
omegaefound=mean(initialomega5);

%back calculate to integral output
intsum_Itension=(((omegaefound/(K_DR5*276.4))...
    -K_state_tension*x(2))/P_tension)/I_tension;

%control signals at t=0; w=1
linespeed_err(1)=0;
tension_err(1)=0;
linespeed_ctrl_out(1)=(P_linespeed*I_linespeed*intsum_Ilinespeed...
    +K_state_linespeed*r5*yinit(3))*276.4;
tension_ctrl_out(1)=(P_tension*I_tension*intsum_Itension...
    +K_state_tension*x(2))*276.4;

%control integrator initialization
w_integral=1;
linespeed_I_cont=I_linespeed*intsum_Ilinespeed;
tension_I_cont=I_tension*intsum_Itension;

%motor 1 and drive 1 values at t=0; w=1
omegae1(1)=linespeed_ctrl_out(1)*K_DR1;
drive1_thetas(1)=0;

if abs(omegae1(1))<(6*2*pi)
    llvolts=VpHzegain1*exp(VpHzexp1*(omegae1(1)/(2*pi)));
    voltmag_1=llvolts/sqrt(3);
elseif abs(omegae1(1))>=(6*2*pi)
    llvolts=VpHzgain1*(omegae1(1)/(2*pi))+VpHzoffset1;
    voltmag_1=llvolts/sqrt(3);
end

vas1=voltmag_1*sqrt(2)*cos(0);
vbs1=voltmag_1*sqrt(2)*cos(0-2*pi/3);
vcs1=voltmag_1*sqrt(2)*cos(0+2*pi/3);

vqss1=(2/3)*vas1-(1/3)*vbs1-(1/3)*vcs1;
vdss1=(-1/sqrt(3))*vbs1+(1/sqrt(3))*vcs1;

vqs1(1)=vqss1*cos(0)-vdss1*sin(0);
vds1(1)=vqss1*sin(0)+vdss1*cos(0);

vmat1=[vqs1(1); vds1(1); vqr1; vdr1];
initmat_1=[Rs1 omegae1(1)*Ls1 0 omegae1(1)*Lm1;...
        -1*omegae1(1)*Ls1 Rs1 -1*omegae1(1)*Lm1 0;...
        0 (omegae1(1)-2*thetadot_1(1))*Lm1 ...
        Rr1 (omegae1(1)-2*thetadot_1(1))*Lr1;...
        -1*(omegae1(1)-2*thetadot_1(1))*Lm1 0 ...
        -1*(omegae1(1)-2*thetadot_1(1))*Lr1 Rr1];

i_init1=inv(initmat_1)*vmat1;
iqs1(1)=i_init1(1);
ids1(1)=i_init1(2);
iqr1(1)=i_init1(3);
idr1(1)=i_init1(4);

psi_init1=Lmat1*i_init1;
psi_qs1(1)=psi_init1(1);
psi_ds1(1)=psi_init1(2);
psi_qr1(1)=psi_init1(3);
psi_dr1(1)=psi_init1(4);

%motor 5 and drive 5 values at t=0; w=1
omegae5(1)=tension_ctrl_out(1)*K_DR5;
drive5_thetas(1)=0;

if abs(omegae5(1))<(6*2*pi)
    llvolts=VpHzegain5*exp(VpHzexp5*(omegae5(1)/(2*pi)));
    voltmag_5=llvolts/sqrt(3);
elseif abs(omegae5(1))>=(6*2*pi)
    llvolts=VpHzgain5*(omegae5(1)/(2*pi))+VpHzoffset5;
    voltmag_5=llvolts/sqrt(3);
end

vas5=voltmag_5*sqrt(2)*cos(0);
vbs5=voltmag_5*sqrt(2)*cos(0-2*pi/3);
vcs5=voltmag_5*sqrt(2)*cos(0+2*pi/3);

vqss5=(2/3)*vas5-(1/3)*vbs5-(1/3)*vcs5;
vdss5=(-1/sqrt(3))*vbs5+(1/sqrt(3))*vcs5;

vqs5(1)=vqss5*cos(0)-vdss5*sin(0);
vds5(1)=vqss5*sin(0)+vdss5*cos(0);

vmat5=[vqs5(1); vds5(1); vqr5; vdr5];
initmat_5=[Rs5 omegae5(1)*Ls5 0 omegae5(1)*Lm5;...
        -1*omegae5(1)*Ls5 Rs5 -1*omegae5(1)*Lm5 0;...
        0 (omegae5(1)-2*thetadot_5(1))*Lm5 ...
        Rr5 (omegae5(1)-2*thetadot_5(1))*Lr5;...
        -1*(omegae5(1)-2*thetadot_5(1))*Lm5 0 ...
        -1*(omegae5(1)-2*thetadot_5(1))*Lr5 Rr5];

i_init5=inv(initmat_5)*vmat5;
iqs5(1)=i_init5(1);
ids5(1)=i_init5(2);
iqr5(1)=i_init5(3);
idr5(1)=i_init5(4);

psi_init5=Lmat5*i_init5;
psi_qs5(1)=psi_init5(1);
psi_ds5(1)=psi_init5(2);
```

```
psi_qr5(1)=psi_init5(3);
psi_dr5(1)=psi_init5(4);

%back calculate initial integral values - several use w=1 values
intsum_tach1=out_tach1(1);
intsum_tach5=out_tach5(1);
intsum_btach1=out_btach1(1);
intsum_btach5=out_btach5(1);
intsum_pten=out_pten(1);
intsum_sten=out_sten(1);
intsum_ad1=out_ad1(1);
intsum_ad2=out_ad2(1);
intsum_ad3=out_ad3(1);

intsum_omegae1=omegae1(1);
drive1_thetas_int=0;
psi_qs1_int=psi_qs1(1);
psi_ds1_int=psi_ds1(1);
psi_qr1_int=psi_qr1(1);
psi_dr1_int=psi_dr1(1);

intsum_omegae5=omegae5(1);
drive5_thetas_int=0;
psi_qs5_int=psi_qs5(1);
psi_ds5_int=psi_ds5(1);
psi_qr5_int=psi_qr5(1);
psi_dr5_int=psi_dr5(1);

%create arrays for signal substitution and control substitution
tension_replace(1:numpts)=0;
tach1_replace(1:numpts)=0;
tach5_replace(1:numpts)=0;
control_substitution(1:numpts)=0;

display('INITIAL CONDITIONS SET')
clock
%--------------------------------------------------------------------%

%--------------------------------------------------------------------%
%instrument sag simulation - reset variables for recovery
intreset=1;
intreset2=1;
intreset3=1;
%--------------------------------------------------------------------%

%--------------------------------------------------------------------%
%indices for referencing sampled waves
q=0;
r=1;
q2=0;
r2=1;
q3=0;
r3=1;
%--------------------------------------------------------------------%
```

```
%-----------------------------------------------------------------------%
%drive variables for trip / no trip
dcbus1(1:numpts)=280;
dcbus5(1:numpts)=280;
dcdecayA=975; %v/sec dc bus decay rate: energized motor
dcdecayB=46.6; %v/sec dc bus decay rate: nonenergized motor
drivetrip1=199.5;
drivetrip5=199.5;
trip1=0;
trip5=0;
driveout1(1:numpts)=1;
driveout5(1:numpts)=1;
coastcommand1(1:numpts)=0;
coastcommand5(1:numpts)=0;
%-----------------------------------------------------------------------%

%-----------------------------------------------------------------------%
%enter loop for each point in time and run calculations
for w=2:numpts         %loop for number of evaluation points
    %-------------------------------------------------------------------%
    %feedback instruments - all first order
    %-------------------------------------------------------------------%
    %main tachometer - roller 1
    %input: thetadot_1(w-1)
    %output: out_tach1B(w)

    %standard operation - prefault, ridethrough, and postfault
    %standard first order for instrument
    tach1_integral=dt*((K_tach1/tau_tach1)*thetadot_1(w-1))...
        -dt*((1/tau_tach1)*(out_tach1(w-1)-offset_tach1));
    intsum_tach1=tach1_integral+intsum_tach1;
    out_tach1(w)=intsum_tach1;
    out_tach1B(w)=out_tach1(w);

    %sagged operation - deviation segment
    if time(w)>t_sagstart+ridetime2 & time(w)<=t_sagend
        %routine to handle dissimilar sampling
        timeindex=time(w)-time(w-q2);
        if timeindex < (time(w)-time(w-4))
            q2=q2+1;
            out_tach1B(w)=wave2(r2);
        elseif timeindex >= (time(w)-time(w-4))
            q2=1;
            r2=r2+1;
            out_tach1B(w)=wave2(r2);
        end
    end

    %sagged operation - recovery segment
    if time(w)>t_sagend & time(w)<t_sagend+10*tau_rec2
        %initial value set
        if intreset2==1
           intreset2=0;
```

```
       initval2=out_tach1B(w-1);
    end

    %calculate gradual sweep multipliers
    dec_multiplier2=exp(-(time(w)-t_sagend)/tau_rec2);
    inc_multiplier2=1-exp(-(time(w)-t_sagend)/tau_rec2);

    %signal definition - equation 2.29
    out_tach1B(w)=initval2*dec_multiplier2+out_tach1(w)*inc_multiplier2;
end

%------------------------------------------------------------------%
%main tachometer - roller 5
%input: thetadot_5(w-1)
%output: out_tach5B(w)

%standard operation - prefault, ridethrough, and postfault
%standard first order for instrument
tach5_integral=dt*((K_tach5/tau_tach5)*thetadot_5(w-1))...
    -dt*((1/tau_tach5)*(out_tach5(w-1)-offset_tach5));
intsum_tach5=tach5_integral+intsum_tach5;
out_tach5(w)=intsum_tach5;
out_tach5B(w)=out_tach5(w);

%sagged operation - deviation segment
if time(w)>t_sagstart+ridetime3 & time(w)<=t_sagend
    %routine to handle dissimilar sampling
    timeindex=time(w)-time(w-q3);
    if timeindex < (time(w)-time(w-4))
        q3=q3+1;
        out_tach5B(w)=wave3(r3);
    elseif timeindex >= (time(w)-time(w-4))
        q3=1;
        r3=r3+1;
        out_tach5B(w)=wave3(r3);
    end
end

%sagged operation - recovery segment
if time(w)>t_sagend & time(w)<t_sagend+10*tau_rec3
    %initial value set
    if intreset3==1
       intreset3=0;
       initval3=out_tach5B(w-1);
    end

    %calculate gradual sweep multipliers
    dec_multiplier3=exp(-(time(w)-t_sagend)/tau_rec3);
    inc_multiplier3=1-exp(-(time(w)-t_sagend)/tau_rec3);

    %signal definition - equation 2.29
    out_tach5B(w)=initval3*dec_multiplier3+out_tach5(w)*inc_multiplier3;
end
```

```
%------------------------------------------------------------------%
%backup tachometer - roller 1
%input: thetadot_1(w-1)
%output: out_btach1(w)
btach1_integral=dt*((K_btach1/tau_btach1)*thetadot_1(w-1))...
    -dt*((1/tau_btach1)*(out_btach1(w-1)-offset_btach1));
intsum_btach1=btach1_integral+intsum_btach1;
out_btach1(w)=intsum_btach1;

%backup tachometer - roller 5
%input: thetadot_5(w-1)
%output: out_btach5(w)
btach5_integral=dt*((K_btach5/tau_btach5)*thetadot_5(w-1))...
    -dt*((1/tau_btach5)*(out_btach5(w-1)-offset_btach5));
intsum_btach5=btach5_integral+intsum_btach5;
out_btach5(w)=intsum_btach5;

%------------------------------------------------------------------%
%primary tension cell
%input: pten_actual(w-1)
%output: out_pten2(w)

%standard operation - prefault, ridethrough, and postfault
%standard first order for instrument
pten_integral=dt*((K_pten/tau_pten)*pten_actual(w-1))...
-dt*((1/tau_pten)*(out_pten(w-1)-offset_pten));
intsum_pten=pten_integral+intsum_pten;
out_pten(w)=intsum_pten;
out_pten2(w)=out_pten(w);

%sagged operation - deviation segment
if time(w)>t_sagstart+ridetime & time(w)<=t_sagend
    %routine to handle dissimilar sampling
    timeindex=time(w)-time(w-q);
    if timeindex < (time(w)-time(w-4))
        q=q+1;
        out_pten2(w)=wave(r);
    elseif timeindex >= (time(w)-time(w-4))
        q=1;
        r=r+1;
        out_pten2(w)=wave(r);
    end
end

%sagged operation - recovery segment
if time(w)>t_sagend & time(w)<t_sagend+10*tau_rec
    %initial value set
    if intreset==1
       intreset=0;
       initval=out_pten2(w-1);
    end

    %calculate gradual sweep multipliers
    dec_multiplier=exp(-(time(w)-t_sagend)/tau_rec);
    inc_multiplier=1-exp(-(time(w)-t_sagend)/tau_rec);

    %signal definition - equation 2.29
    out_pten2(w)=initval*dec_multiplier+out_pten(w)*inc_multiplier;
end

%-----------------------------------------------------------------%
%secondary tension cell
%input: sten_actual(w-1)
%output: out_sten(w)
sten_integral=dt*((K_sten/tau_sten)*sten_actual(w-1))...
    -dt*((1/tau_sten)*(out_sten(w-1)-offset_sten));
intsum_sten=sten_integral+intsum_sten;
out_sten(w)=intsum_sten;
%-----------------------------------------------------------------%

%-----------------------------------------------------------------%
%instrument input block - three instances

%zero order hold for sampling
timeindex1=0.1*(floor(time(w)/0.1));
oldw1=floor((timeindex1/(1/10000))+1);

%tach1 input signal
%input: out_tach1(w-1)
%output: out_ad1(w)
ad_integral1=dt*((K_ad/tau_ad)*out_tach1B(w-1))...
    -dt*((1/tau_ad)*(out_ad1(w-1)-offset_ad));
intsum_ad1=ad_integral1+intsum_ad1;
out_ad1A(w)=intsum_ad1;
out_ad1(w)=out_ad1A(oldw1);
out_ad1B(w)=out_ad1(w);

%tach5 input signal
%input: out_tach5(w-1)
%output: out_ad2(w)
ad_integral2=dt*((K_ad/tau_ad)*out_tach5B(w-1))...
    -dt*((1/tau_ad)*(out_ad2(w-1)-offset_ad));
intsum_ad2=ad_integral2+intsum_ad2;
out_ad2A(w)=intsum_ad2;
out_ad2(w)=out_ad2A(oldw1);
out_ad2B(w)=out_ad2(w);

%primary tension input signal
%input: out_pten2(w-1)
%output: out_ad3(w)
ad_integral3=dt*((K_ad/tau_ad)*out_pten2(w-1))...
    -dt*((1/tau_ad)*(out_ad3(w-1)-offset_ad));
intsum_ad3=ad_integral3+intsum_ad3;
out_ad3A(w)=intsum_ad3;
out_ad3(w)=out_ad3A(oldw1);
out_ad3B(w)=out_ad3(w);

%INTERRUPT SIGNALS IF RIDETIMES EXCEEDED - SELECTIVE SUBSTITUTION
```

```
    %3 IF STATEMENTS REASSIGN OUT_AD*B VALUES
%     if time(w)>=0.5+ridetime2 & time(w)<=0.5+0.450+rectime2
%         out_ad1B(w)=mean(out_ad1B(10:4000));
%         intsum_ad1=out_ad1B(w);
%         tach1_replace(w)=1;
%     end
%     if time(w)>=0.5+ridetime3 & time(w)<=0.5+0.450+rectime3
%         out_ad2B(w)=mean(out_ad2B(10:4000));
%         intsum_ad2=out_ad2B(w);
%         tach5_replace(w)=1;
%     end
%     if time(w)>=0.5+ridetime & time(w)<=0.5+0.450+rectime
%         out_ad3B(w)=mean(out_ad3B(10:4000));
%         intsum_ad3=out_ad3B(w);
%         tension_replace(w)=1;
%     end

    %-----------------------------------------------------------------%

    %-----------------------------------------------------------------%
    %control calculations

    %scale feedback signals

    %tach1 signal
    tach1_sig=(((out_ad1B(w)-offset_ad)/K_ad)-offset_tach1)/K_tach1;

    %tach5 signal
    tach5_sig=(((out_ad2B(w)-offset_ad)/K_ad)-offset_tach5)/K_tach5;

    %primary tension signal
    pten_sig=(((out_ad3B(w)-offset_ad)/K_ad)-offset_pten)/K_pten;

    %calculate PVs from scaled feed back signals
    linespeed_PV=tach1_sig;
    tension_PV=pten_sig*r5;

    %calculate control SPs from scaled feedback signals and
    %desired values of linespeed and tension
    linespeed_SP=linespeed_set/r1;
    tension_SP=tension_set*r5;

    %calculate control signal errors
    linespeed_err(w)=linespeed_SP-linespeed_PV;
    tension_err(w)=tension_PV-tension_SP;

    %calculate integral contribution to control
    %calculated every PLC cycle
    dt_PLC=0.01; %PLC cycle interval

    %IF STATEMENT MANAGES INTEGRAL HOLD, ELSE IS FOR NORMAL OPERATION
    if time(w)>=t_sagstart+0.006 & time(w)<=t_sagend+1.5
        intsum_Ilinespeed=intsum_Ilinespeed;
        intsum_Itension=intsum_Itension;
```

```
else
    if time(w)-time(w_integral)>=dt_PLC-0.000001
        w_integral=w;
        intsum_Ilinespeed=intsum_Ilinespeed+linespeed_err(w)*dt_PLC;
        linespeed_I_cont=I_linespeed*intsum_Ilinespeed;
        intsum_Itension=intsum_Itension+tension_err(w)*dt_PLC;
        tension_I_cont=I_tension*intsum_Itension;
    end
end

%record integral sums for reference
I_line(w)=intsum_Ilinespeed;
I_ten(w)=intsum_Itension;

%calculate proportional contribution to control
linespeed_P_cont=P_linespeed*linespeed_err(w);
tension_P_cont=P_tension*tension_err(w);

%sum proportional and integral terms
linespeed_PIsum(w)=linespeed_P_cont+P_linespeed*linespeed_I_cont;
tension_PIsum(w)=tension_P_cont+P_tension*tension_I_cont;

%calculate state feedback terms
linespeed_state_cont(w)=r5*pten_sig*K_state_linespeed;
tension_state_cont(w)=tach5_sig*K_state_tension;

%sum state feedback term and PI sum
linespeed_control(w)=linespeed_PIsum(w)+linespeed_state_cont(w);
tension_control(w)=tension_PIsum(w)+tension_state_cont(w);

%multiply by intrinsic constant used in Siemens PLC
linespeed_ctrl_out(w)=linespeed_control(w)*276.4;
tension_ctrl_out(w)=tension_control(w)*276.4;
%-----------------------------------------------------------------%

%-----------------------------------------------------------------%
%drive maximum and minimum saturation at +-15Hz
if linespeed_ctrl_out(w)>((15*2*pi)/K_DR1)
    linespeed_ctrl_out(w)=((15*2*pi)/K_DR1);
elseif linespeed_ctrl_out(w)<((-15*2*pi)/K_DR1)
    linespeed_ctrl_out(w)=((-15*2*pi)/K_DR1);
end

if tension_ctrl_out(w)>((15*2*pi)/K_DR5)
    tension_ctrl_out(w)=((15*2*pi)/K_DR5);
elseif tension_ctrl_out(w)<((-15*2*pi)/K_DR5)
    tension_ctrl_out(w)=((-15*2*pi)/K_DR5);
end

%CALCULATE AND OUTPUT PREFAULT AVERAGE VALUES FOR
%CONTROL OUTPUTS
if time(w)>=t_sagstart+0.006 & time(w)<=t_sagend+1.5
    linespeed_ctrl_out(w)=mean(linespeed_ctrl_out(10:4000));
    tension_ctrl_out(w)=mean(tension_ctrl_out(10:4000));
    control_substitution(w)=1;
end
%----------------------------------------------------------------%

%----------------------------------------------------------------%
%motor drives
%inputs: linespeed_ctrl_out(w),tension_ctrl_out(w)
%ouputs: torque_1(w),torque_5(w)
%each drive done separately

%***drive 1
%determine rotating field frequency
omegae1_integral=dt*((K_DR1/tau_DR1)*linespeed_ctrl_out(w-1))...
    -dt*((1/tau_DR1)*omegae1(w-1));
omegae1(w)=omegae1_integral+intsum_omegae1;
intsum_omegae1=omegae1(w);

%determine angles
drive1_thetas(w)=drive1_thetas_int+dt*omegae1(w);
drive1_thetas_int=drive1_thetas(w);

if abs(omegae1(w))<(6*2*pi)
    llvolts=VpHzegain1*exp(VpHzexp1*(omegae1(w)/(2*pi)));
    voltmag_1=llvolts/sqrt(3);
elseif abs(omegae1(w))>=(6*2*pi)
    llvolts=VpHzgain1*(omegae1(w)/(2*pi))+VpHzoffset1;
    voltmag_1=llvolts/sqrt(3);
end

%COAST COMMAND
if time(w)>=t_sagstart+0.025 & time(w)<=t_sagend+0.002
    coastcommand1(w)=1;
end

%DRIVE DROPOUT SEQUENCE AND DC BUS DECAY
%SECOND CONDITIONAL INCLUDED IN RECOVERY
if time(w)>=t_sagstart & time(w)<=t_sagend
    if dcbus1(w-1)>=drivetrip1
        dcbus1(w)=dcbus1(w-1)-dcdecayA*dt;
    end
    if dcbus1(w-1)<drivetrip1 | coastcommand1(w)==1
        dcbus1(w)=dcbus1(w-1)-dcdecayB*dt;
        driveout1(w)=0;
        voltmag_1=0;
        psi_qs1(w-1)=0;
        psi_qr1(w-1)=0;
        psi_ds1(w-1)=0;
        psi_dr1(w-1)=0;
    end
end

if trip1==0 & dcbus1(w)<drivetrip1
    trip1=1;
end
```

```
%if driveout1(w-1)==0 %maintains trip for no mitigation
if trip1==1;
    driveout1(w)=0;
    voltmag_1=0;
    psi_qs1(w-1)=0;
    psi_qr1(w-1)=0;
    psi_ds1(w-1)=0;
    psi_dr1(w-1)=0;
end

vas1=voltmag_1*sqrt(2)*cos(drive1_thetas(w));
vbs1=voltmag_1*sqrt(2)*cos(drive1_thetas(w)-2*pi/3);
vcs1=voltmag_1*sqrt(2)*cos(drive1_thetas(w)+2*pi/3);

vqss1=(2/3)*vas1-(1/3)*vbs1-(1/3)*vcs1;
vdss1=-(1/sqrt(3))*vbs1+(1/sqrt(3))*vcs1;

vqs1(w)=vqss1*cos(drive1_thetas(w))-vdss1*sin(drive1_thetas(w));
vds1(w)=vqss1*sin(drive1_thetas(w))+vdss1*cos(drive1_thetas(w));

%calculate dq fluxes
psi_qs1(w)=psi_qs1_int+dt*(vqs1(w-1)-Rs1*iqs1(w-1)...
    -omegae1(w-1)*psi_ds1(w-1));
psi_qs1_int=psi_qs1(w);

psi_ds1(w)=psi_ds1_int+dt*(vds1(w-1)-Rs1*ids1(w-1)...
    +omegae1(w-1)*psi_qs1(w-1));
psi_ds1_int=psi_ds1(w);

psi_qr1(w)=psi_qr1_int+dt*(vqr1-Rr1*iqr1(w-1)...
    -(omegae1(w-1)-2*thetadot_1(w-1))*psi_dr1(w-1));
psi_qr1_int=psi_qr1(w);

psi_dr1(w)=psi_dr1_int+dt*(vdr1-Rr1*idr1(w-1)...
    +(omegae1(w-1)-2*thetadot_1(w-1))*psi_qr1(w-1));
psi_dr1_int=psi_dr1(w);

%calculate dq currents
psi1=[psi_qs1(w);psi_ds1(w);psi_qr1(w);psi_dr1(w)];
I1=inv(Lmat1)*psi1;
iqs1(w)=I1(1);
ids1(w)=I1(2);
iqr1(w)=I1(3);
idr1(w)=I1(4);

%calculate developed torque
torque_1(w)=(3/2)*(2)*(Lm1)...
    *(iqs1(w)*idr1(w)-ids1(w)*iqr1(w));

%***drive 5
%determine rotating field frequency
omegae5_integral=dt*((K_DR5/tau_DR5)*tension_ctrl_out(w-1))...
    -dt*((1/tau_DR5)*omegae5(w-1));
```

```
omegae5(w)=omegae5_integral+intsum_omegae5;
intsum_omegae5=omegae5(w);

%develop pwm voltage signals
drive5_thetas(w)=drive5_thetas_int+dt*omegae5(w);
drive5_thetas_int=drive5_thetas(w);

if abs(omegae5(w))<(6*2*pi)
    llvolts=VpHzegain5*exp(VpHzexp5*(omegae5(w)/(2*pi)));
    voltmag_5=llvolts/sqrt(3);
elseif abs(omegae5(w))>=(6*2*pi)
   llvolts=VpHzgain5*(omegae5(w)/(2*pi))+VpHzoffset5;
    voltmag_5=llvolts/sqrt(3);
end

%COAST COMMAND
if time(w)>=t_sagstart+0.025 & time(w)<=t_sagend+0.002
    coastcommand5(w)=1;
end

%DRIVE DROPOUT SEQUENCE AND DC BUS DECAY
%SECOND CONDITIONAL INCLUDED IN RECOVERY
if time(w)>=t_sagstart & time(w)<=t_sagend
    if dcbus5(w-1)>=drivetrip5
        dcbus5(w)=dcbus5(w-1)-dcdecayA*dt;
    end
    if dcbus5(w-1)<drivetrip5 | coastcommand5(w)==1
        dcbus5(w)=dcbus5(w-1)-dcdecayB*dt;
        driveout5(w)=0;
        voltmag_5=0;
        psi_qs5(w-1)=0;
        psi_qr5(w-1)=0;
        psi_ds5(w-1)=0;
        psi_dr5(w-1)=0;
    end
end

%permanent trip after loss of drive
if trip5==0 & dcbus5(w)<drivetrip5
    trip5=1;
end

%if driveout5(w-1)==0; %maintains trip for no mitigation
if trip5==1;
    driveout5(w)=0;
    voltmag_5=0;
    psi_qs5(w-1)=0;
    psi_qr5(w-1)=0;
    psi_ds5(w-1)=0;
    psi_dr5(w-1)=0;
end

vas5=voltmag_5*sqrt(2)*cos(drive5_thetas(w));
vbs5=voltmag_5*sqrt(2)*cos(drive5_thetas(w)-2*pi/3);
```

```
vcs5=voltmag_5*sqrt(2)*cos(drive5_thetas(w)+2*pi/3);

vqss5=(2/3)*vas5-(1/3)*vbs5-(1/3)*vcs5;
vdss5=(-1/sqrt(3))*vbs5+(1/sqrt(3))*vcs5;

vqs5(w)=vqss5*cos(drive5_thetas(w))-vdss5*sin(drive5_thetas(w));
vds5(w)=vqss5*sin(drive5_thetas(w))+vdss5*cos(drive5_thetas(w));

%calculate dq fluxes
psi_qs5(w)=psi_qs5_int+dt*(vqs5(w-1)-Rs5*iqs5(w-1)...
    -omegae5(w-1)*psi_ds5(w-1));
psi_qs5_int=psi_qs5(w);

psi_ds5(w)=psi_ds5_int+dt*(vds5(w-1)-Rs5*ids5(w-1)...
    +omegae5(w-1)*psi_qs5(w-1));
psi_ds5_int=psi_ds5(w);

psi_qr5(w)=psi_qr5_int+dt*(vqr5-Rr5*iqr5(w-1)...
    -(omegae5(w-1)-2*thetadot_5(w-1))*psi_dr5(w-1));
psi_qr5_int=psi_qr5(w);

psi_dr5(w)=psi_dr5_int+dt*(vdr5-Rr5*idr5(w-1)...
    +(omegae5(w-1)-2*thetadot_5(w-1))*psi_qr5(w-1));
psi_dr5_int=psi_dr5(w);

%calculate dq currents
psi5=[psi_qs5(w);psi_ds5(w);psi_qr5(w);psi_dr5(w)];
I5=inv(Lmat5)*psi5;
iqs5(w)=I5(1);
ids5(w)=I5(2);
iqr5(w)=I5(3);
idr5(w)=I5(4);

%calculate developed torque
torque_5(w)=(3/2)*(2)*(Lm5)...
    *(iqs5(w)*idr5(w)-ids5(w)*iqr5(w));
%-----------------------------------------------------------------%

%-----------------------------------------------------------------%
%mechanical system

%assemble input matrix from motor developed torques
u1=torque_1(w);
u2=torque_5(w);

%calculate output matrix y=Cx+Du
%x is initialized for w=1, recalculated for next iteration later
y=C*x+D*[u1;u2];

%determine linespeed and tension variables from output matrix
linespeed(w)=y(1);
thetadot_1(w)=y(1)/r1;
thetadot_5(w)=y(2)/r5;
pten_actual(w)=y(3);
```

```
    sten_actual(w)=y(4);

    %slack
    if pten_actual(w) < 0
       pten_actual(w) = 0;
    end

    if sten_actual(w) < 0
        sten_actual(w) = 0;
    end

    %calculate x for next iteration
    x=x_intsum+dt*(A*x+B*[u1;u2]);
    x_intsum=x;

    %progress messages
    if rem(w,10000)==0
        w
        clock
    end
    %-----------------------------------------------------------------%

%loop end
end
%---------------------------------------------------------------------%
%END PROGRAM SAGSIM.M
```

Appendix E
Instrumentation Voltage Sag Response Data

This appendix contains the raw data obtained from sag testing the textile tension controller analog instrumentation.. All values are given in seconds. Voltage sag durations for ridethrough time tests were fixed at 450ms. Magnitudes were varied from 0% to 80% in 20% increments. When the sensor rode through the event without an observable disturbance, 'RT' is indicated. Maximum recovery times found during the test session were also recorded. Sag durations were varied up to 450ms to ensure the maximum recovery time was obtained.

Table E.1 Ridethrough times obtained from sag testing of tension sensor.

PHYS. INPUT	**V=0%**	**V=20%**	**V=40%**	**V=60%**	**V=80%**
5N	0.085	0.074	0.074	0.074	RT
14N	0.274	0.274	RT	RT	RT
23N	0.150	0.147	0.136	0.124	RT
32N	0.078	0.075	0.072	0.076	RT
41N	0.060	0.060	0.064	0.064	RT
50N	0.060	0.060	0.060	0.060	RT

Table E.2 Ridethrough times obtained from sag testing of payoff tachometer.

PHYS. INPUT	**V=0%**	**V=20%**	**V=40%**	**V=60%**	**V=80%**
500 mm/s	0.096	0.096	0.096	RT	RT
1000 mm/s	0.090	0.090	0.090	RT	RT
1500 mm/s	0.086	0.086	0.086	RT	RT
2000 mm/s	0.080	0.080	0.080	RT	RT
2500 mm/s	0.076	0.076	0.076	RT	RT

Table E.3 Ridethrough times obtained from sag testing of takeup tachometer.

PHYS. INPUT	**V=0%**	**V=20%**	**V=40%**	**V=60%**	**V=80%**
500 mm/s	0.094	0.094	0.094	RT	RT
1000 mm/s	0.086	0.086	0.086	RT	RT
1500 mm/s	0.080	0.080	0.080	RT	RT
2000 mm/s	0.080	0.080	0.080	RT	RT
2500 mm/s	0.076	0.076	0.076	RT	RT

Table E.4 Maximum recovery times obtained from instrument sag tests.

INSTRUMENT	**MAX RECOVERY TIME**
tension cell	0.088
payoff tachometer	0.246
takeup tachometer	0.246

Appendix F

AC Motor Drive Voltage Sag Response Data

This appendix contains AC motor drive ridethrough results for sags of varying magnitudes and phasing combinations, at a duration of 450ms. Sag magnitudes are expressed as percentages in phase order A-B-C. Combinations marked with a '*' indicate a dropout, where the drive halted its output during the sag.

Table F.1 AC motor drive sag response data.

(0,0,0)*	(20,0,0)*	(40,0,0)*
(0,0,20)*	(20,0,20)*	(40,0,20)*
(0,0,40)*	(20,0,40)*	(40,0,40)*
(0,0,60)*	(20,0,60)*	(40,0,60)*
(0,0,80)*	(20,0,80)*	(40,0,80)*
(0,0,100)*	(20,0,100)*	(40,0,100)*
(0,20,0)*	(20,20,0)*	(40,20,0)*
(0,20,20)*	(20,20,20)*	(40,20,20)*
(0,20,40)*	(20,20,40)*	(40,20,40)*
(0,20,60)*	(20,20,60)*	(40,20,60)*
(0,20,80)*	(20,20,80)*	(40,20,80)*
(0,20,100)*	(20,20,100)*	(40,20,100)*
(0,40,0)*	(20,40,0)*	(40,40,0)*
(0,40,20)*	(20,40,20)*	(40,40,20)*
(0,40,40)*	(20,40,40)*	(40,40,40)*
(0,40,60)*	(20,40,60)*	(40,40,60)*
(0,40,80)*	(20,40,80)*	(40,40,80)*
(0,40,100)*	(20,40,100)*	(40,40,100)*
(0,60,0)*	(20,60,0)*	(40,60,0)*
(0,60,20)*	(20,60,20)*	(40,60,20)*
(0,60,40)*	(20,60,40)*	(40,60,40)*
(0,60,60)*	(20,60,60)*	(40,60,60)*
(0,60,80)*	(20,60,80)*	(40,60,80)*
(0,60,100)	(20,60,100)	(40,60,100)
(0,80,0)*	(20,80,0)*	(40,80,0)*
(0,80,20)*	(20,80,20)*	(40,80,20)*
(0,80,40)*	(20,80,40)*	(40,80,40)*
(0,80,60)*	(20,80,60)*	(40,80,60)*
(0,80,80)	(20,80,80)	(40,80,80)
(0,80,100)	(20,80,100)	(40,80,100)
(0,100,0)*	(20,100,0)*	(40,100,0)*
(0,100,20)*	(20,100,20)*	(40,100,20)*
(0,100,40)*	(20,100,40)*	(40,100,40)*
(0,100,60)	(20,100,60)	(40,100,60)
(0,100,80)	(20,100,80)	(40,100,80)
(0,100,100)	(20,100,100)	(40,100,100)

Table F.2 AC motor drive sag response data (continued).

(60,0,0)*	(80,0,0)*	(100,0,0)*
(60,0,20)*	(80,0,20)*	(100,0,20)*
(60,0,40)*	(80,0,40)*	(100,0,40)*
(60,0,60)*	(80,0,60)*	(100,0,60)
(60,0,80)*	(80,0,80)	(100,0,80)
(60,0,100)	(80,0,100)	(100,0,100)
(60,20,0)*	(80,20,0)*	(100,20,0)*
(60,20,20)*	(80,20,20)*	(100,20,20)*
(60,20,40)*	(80,20,40)*	(100,20,40)*
(60,20,60)*	(80,20,60)*	(100,20,60)
(60,20,80)*	(80,20,80)	(100,20,80)
(60,20,100)	(80,20,100)	(100,20,100)
(60,40,0)*	(80,40,0)*	(100,40,0)*
(60,40,20)*	(80,40,20)*	(100,40,20)*
(60,40,40)*	(80,40,40)*	(100,40,40)*
(60,40,60)*	(80,40,60)*	(100,40,60)
(60,40,80)*	(80,40,80)	(100,40,80)
(60,40,100)	(80,40,100)	(100,40,100)
(60,60,0)*	(80,60,0)*	(100,60,0)
(60,60,20)*	(80,60,20)*	(100,60,20)
(60,60,40)*	(80,60,40)*	(100,60,40)
(60,60,60)*	(80,60,60)*	(100,60,60)
(60,60,80)*	(80,60,80)	(100,60,80)
(60,60,100)	(80,60,100)	(100,60,100)
(60,80,0)*	(80,80,0)	(100,80,0)
(60,80,20)*	(80,80,20)	(100,80,20)
(60,80,40)*	(80,80,40)	(100,80,40)
(60,80,60)*	(80,80,60)	(100,80,60)
(60,80,80)	(80,80,80)	(100,80,80)
(60,80,100)	(80,80,100)	(100,80,100)
(60,100,0)	(80,100,0)	(100,100,0)
(60,100,20)	(80,100,20)	(100,100,20)
(60,100,40)	(80,100,40)	(100,100,40)
(60,100,60)	(80,100,60)	(100,100,60)
(60,100,80)	(80,100,80)	(100,100,80)
(60,100,100)	(80,100,100)	(100,100,100)

Appendix G

LabView Code For Data Read and Profibus Functions

This appendix contains LabView code for the data file read and Profibus interface functions used in the Ridethrough PC.

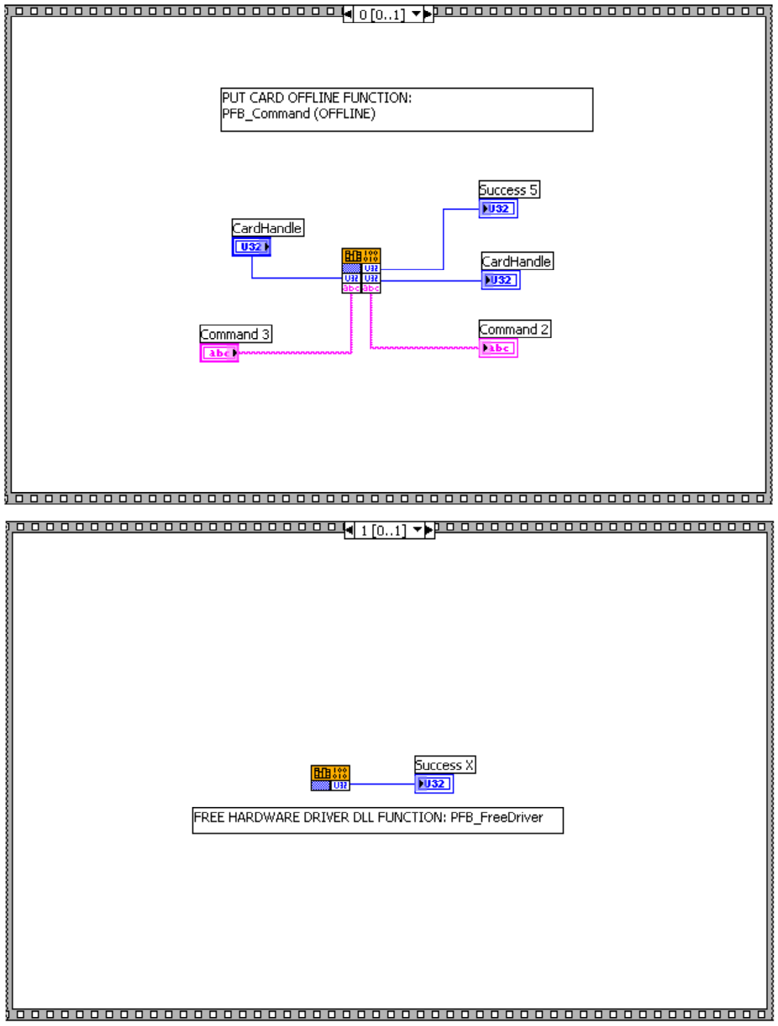
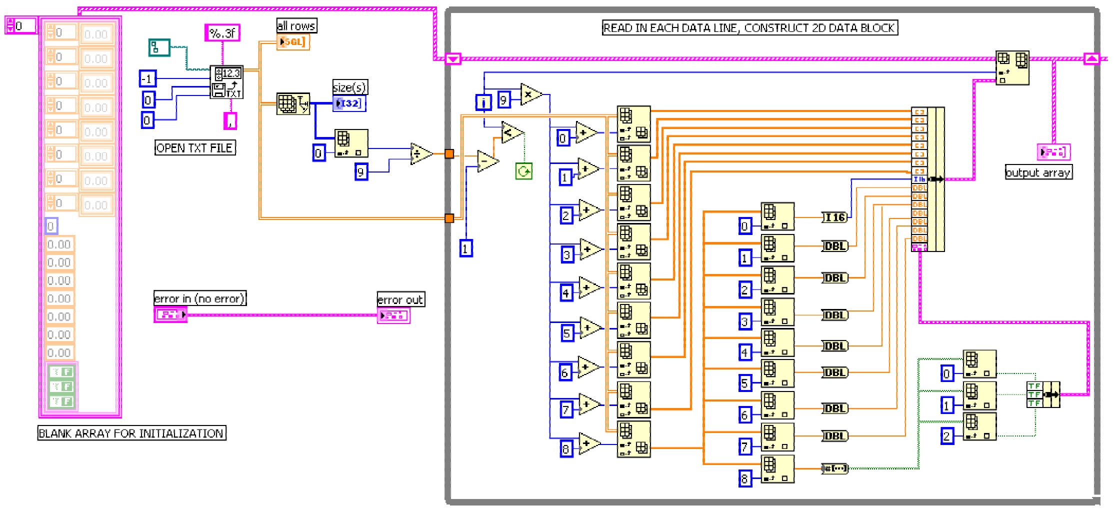
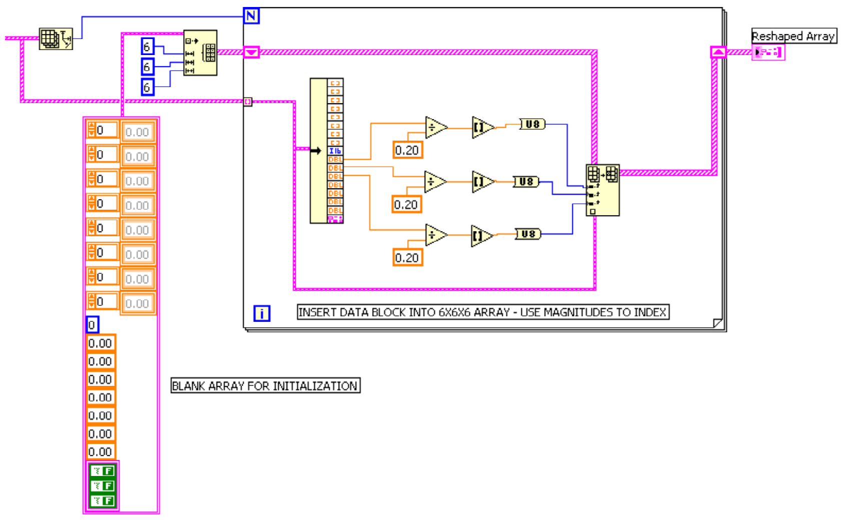


Figure G.1 Labview subroutine to close Profibus interface.



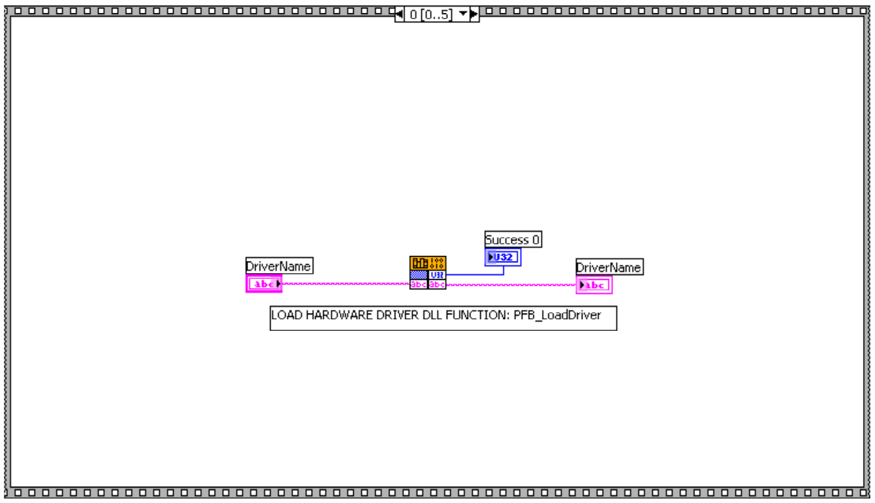
READ IN EACH DATA LINE, CONSTRUCT 2D DATA BLOCK
%.3f
all rows
size(s)
OPEN TXT FILE
error in (no error)
error out
output array
BLANK ARRAY FOR INITIALIZATION

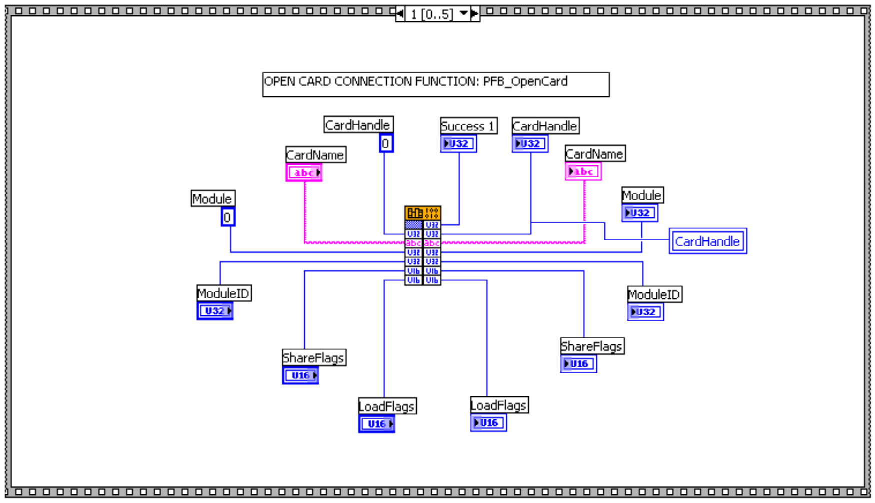
Figure G.2 Labview subroutine to read in data file for ridethrough PC (part 1 of 2).



Reshaped Array
INSERT DATA BLOCK INTO 6X6X6 ARRAY - USE MAGNITUDES TO INDEX
BLANK ARRAY FOR INITIALIZATION

Figure G.3 Labview subroutine to read in data file for ridethrough PC (part 2 of 2).





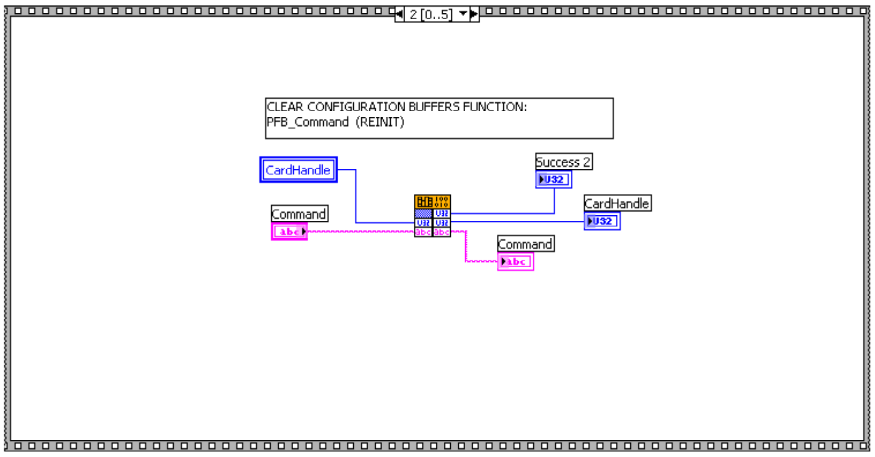


Figure G.4 Labview subroutine for Profibus open interface (part 1 of 2).

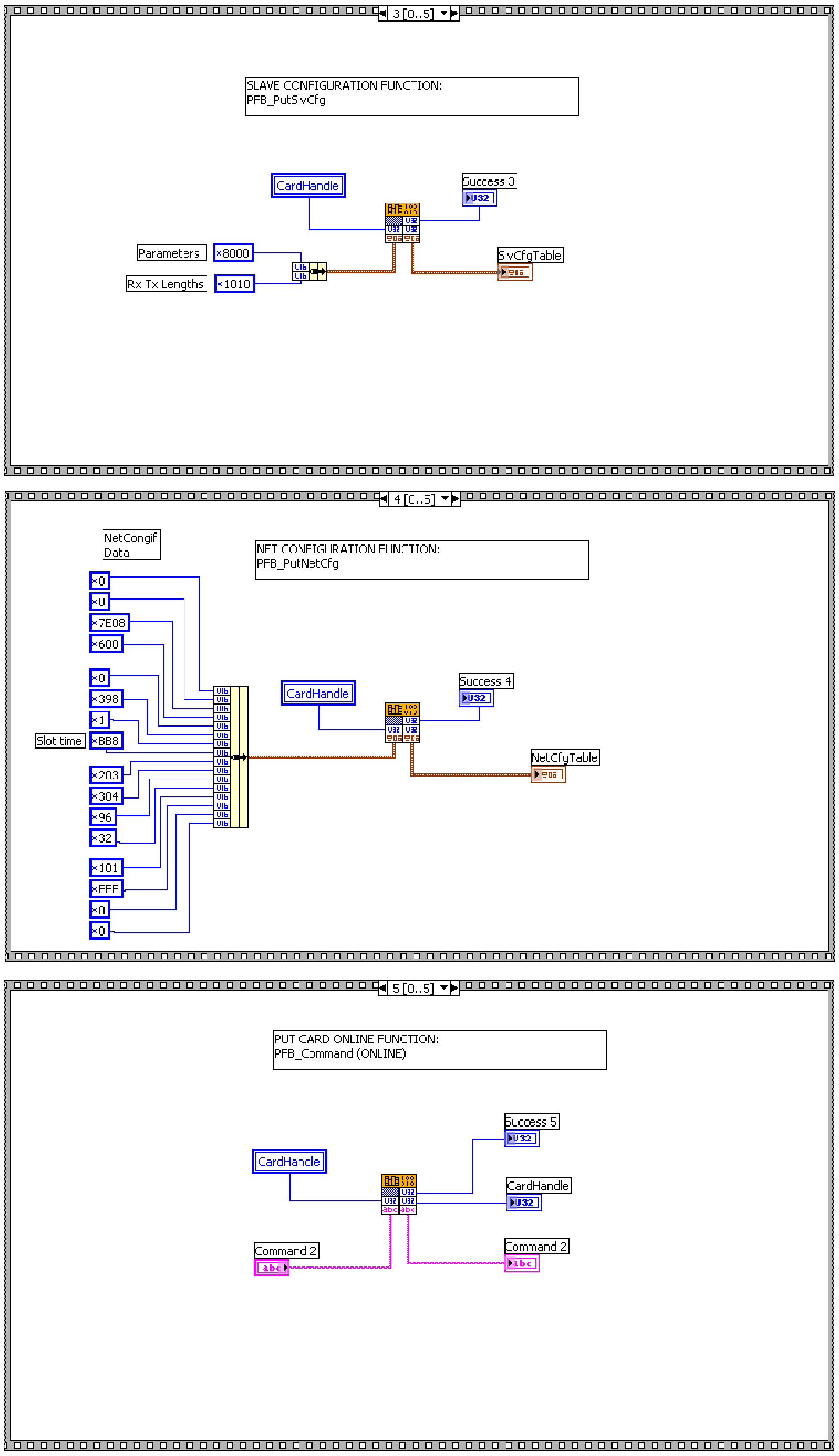


Figure G.5 Labview subroutine for Profibus open interface (part 2 of 2).

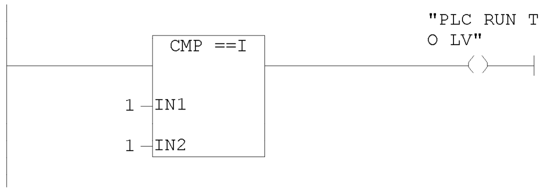
Appendix H

PLC Ladder Code

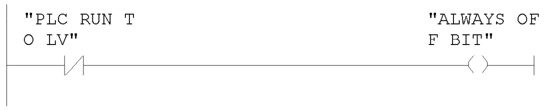
This appendix contains the ladder code used in the textile tension control process PLC and the added subroutine used to interface with the Ridethrough PC.

```
Block: OB1   MAIN WEBSTAND PROGRAM
SECTION ONE - GENERAL I/O AND DIRECTION DETERMINATION
```

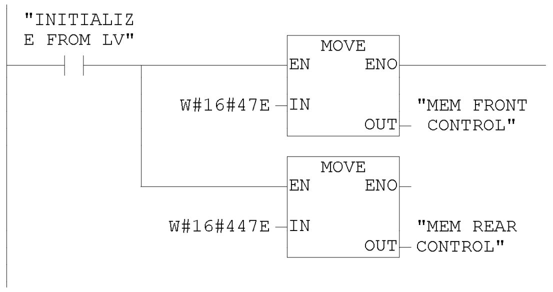
```
Network: 1      Create 'always high' bit: on if 1=1
```



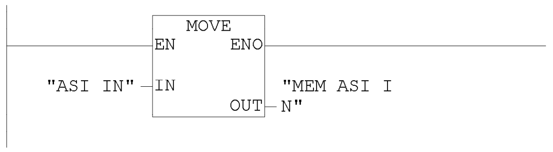
```
Network: 2      Create always off bit
```



```
Network: 3      Send initialize word to drives
```

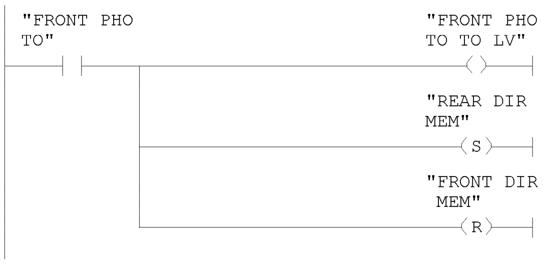


```
Network: 4      Move ASI input to appropriate location
```



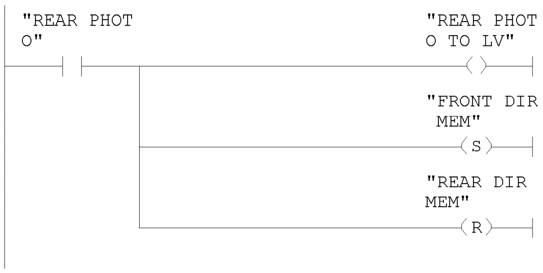
Network: 5   Front spool state affect on direction

When front triggers, front is full. Rear then becomes takeup
and front becomes payoff (PO=low, TU high)

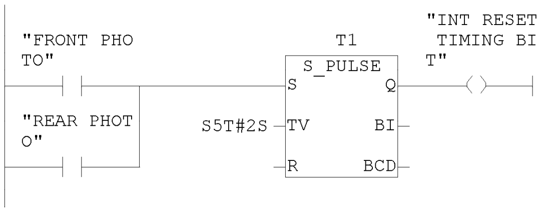


Network: 6   Rear spool state affect on direction

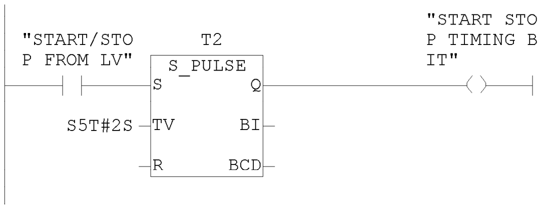
When rear triggers, rear is full. Rear then becomes payoff,
and front becomes takeup (PO=low, TU=high)



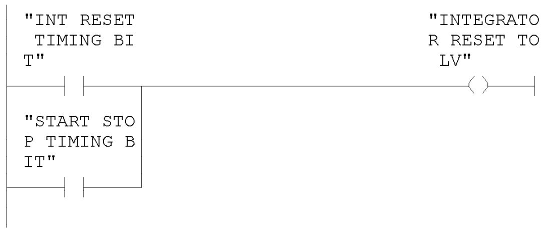
Network: 7   Set integral reset bit on reversal



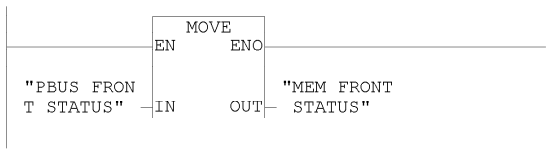
Network: 8   Set integral reset bit on reversal



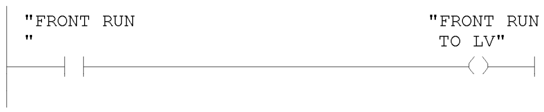
Network: 9   Set LV reset bit while timer is on



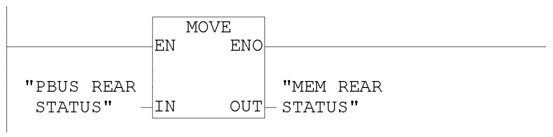
Network: 10   Move front status into memory location



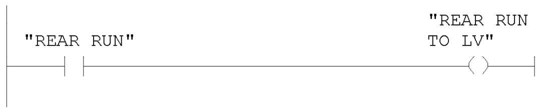
Network: 11   Determine front drive state



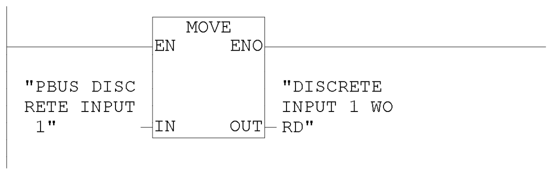
Network: 12   Move rear status into memory location



Network: 13   Determine rear drive state

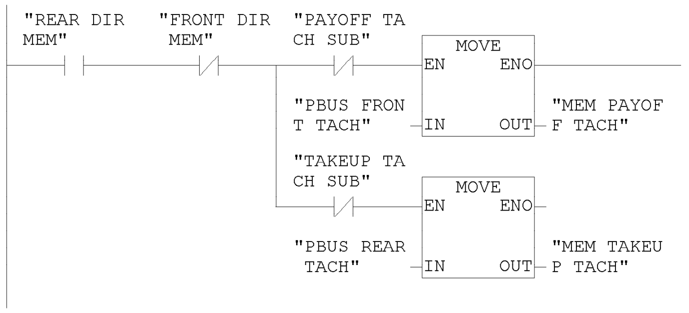


Network: 14   Move discrete input word



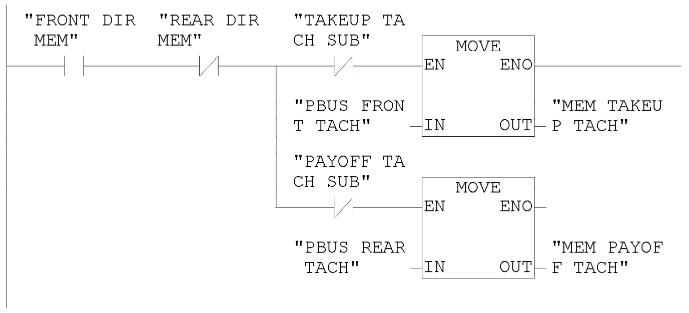
Network: 15   Move tach and radii into memory location - forward

If rear is takeup and front is payoff, move tachs and radii
to appropriate locations (running forward)

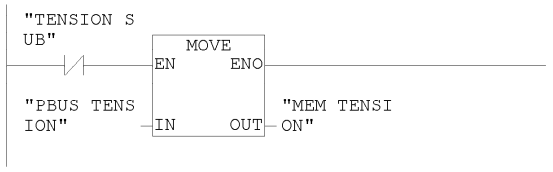


Network: 16   Move tach and radii into memory location - reverse

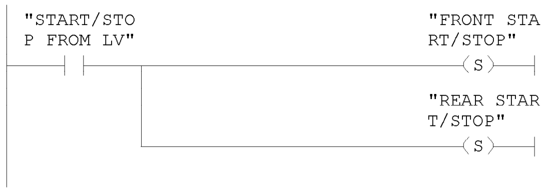
If front is takeup and rear is payoff, move tachs and radii
to appropriate locations (running reverse)



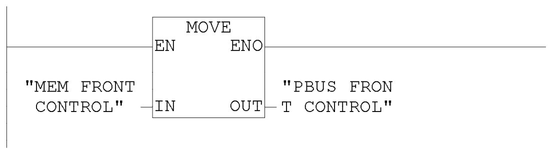
Network: 17   Move tension value into memory location



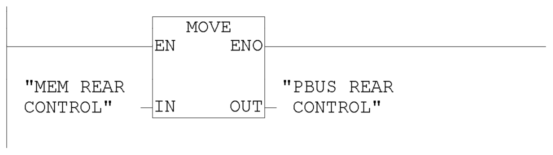
Network: 18   Start/stop drives



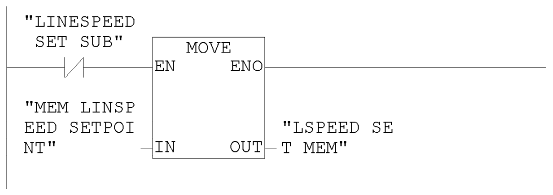
Network: 19   Move control word to front drive



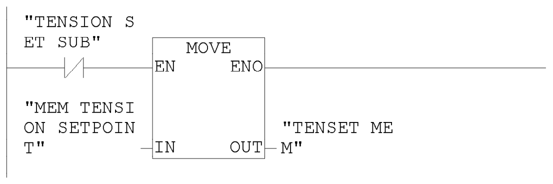
Network: 20   Move control word to rear drive



Network: 21   Move linespeed setpoint

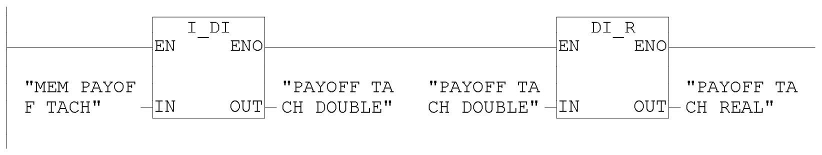


Network: 22   Move tension setpoint

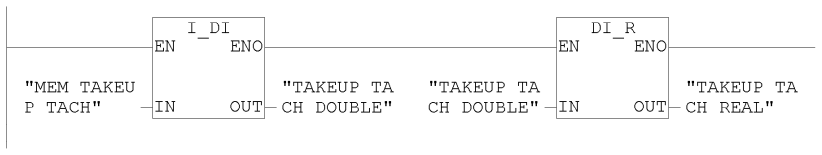


Network: 23   Convert payoff tach to real

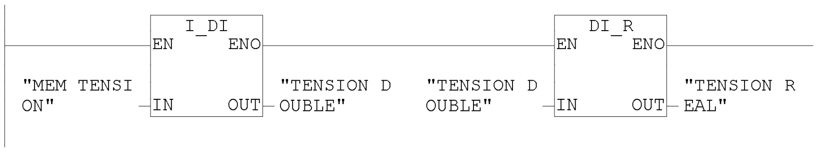
SECTION TWO - CONVERT CRITICAL VARIABLES TO REAL, SCALE TO APPROPRIATE UNITS



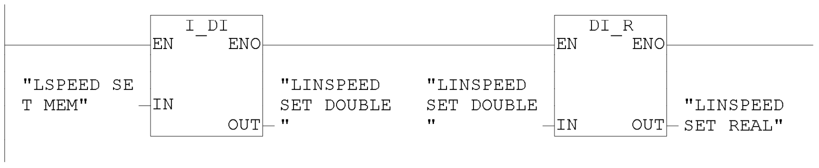
Network: 24   Convert takeup tach to real



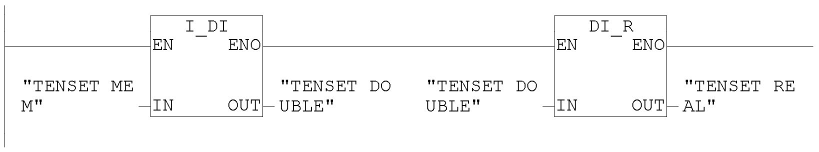
```
Network: 25     Convert tension to real
```



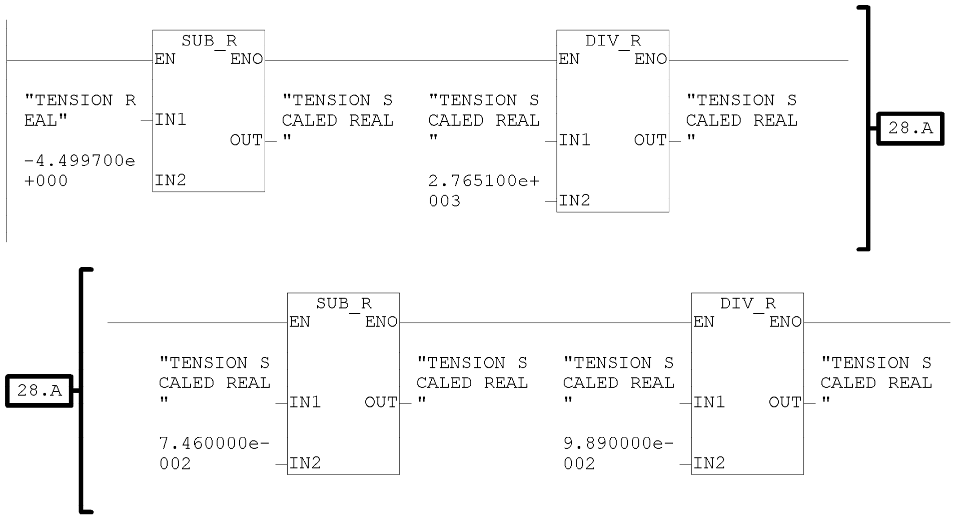
```
Network: 26     Convert linspeed setpoint to real
```



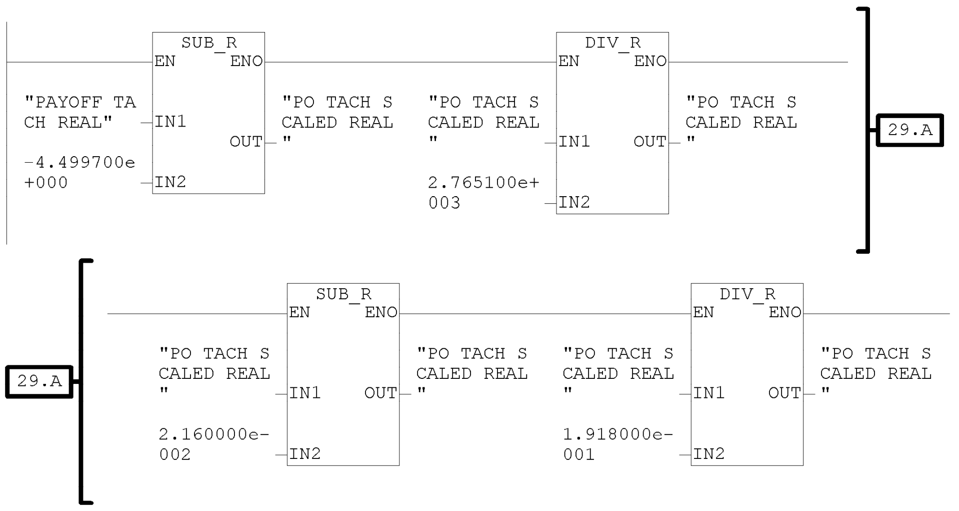
```
Network: 27     Convert tension setpoint to real
```



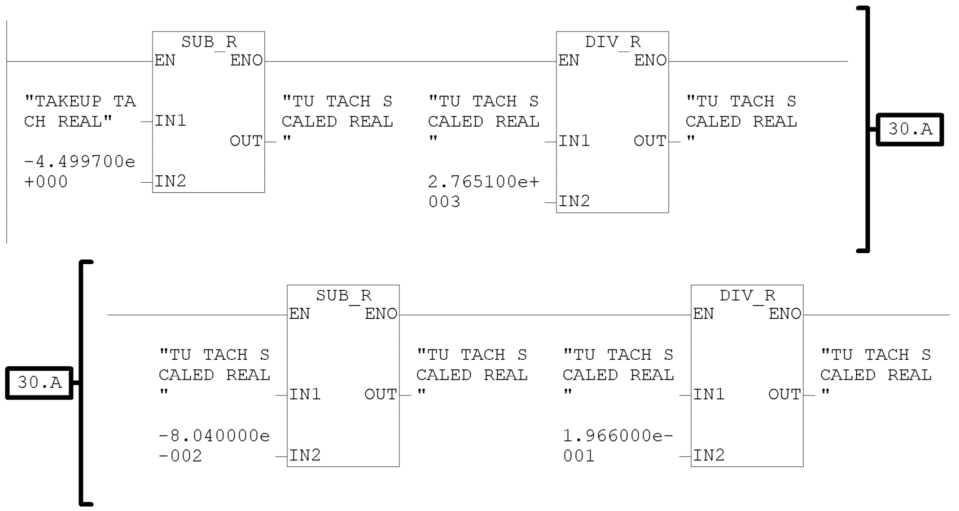
```
Network: 28     Scale tension input to newtons
SCALING STARTS HERE
```



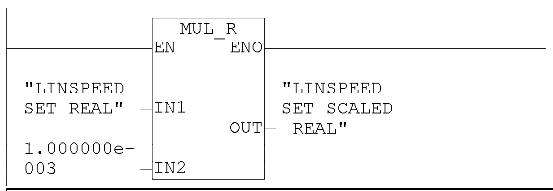
Network: 29   Scale PO tach input to radians per second (PO=FRONT)



Network: 30   Scale TU tach input to radians per second (TU=REAR)

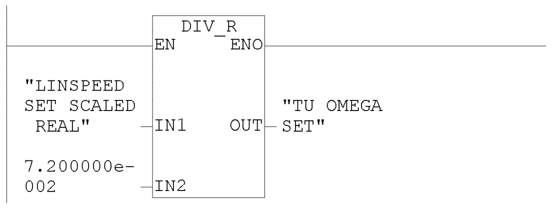


Network: 31   Convert linspeed set to meters

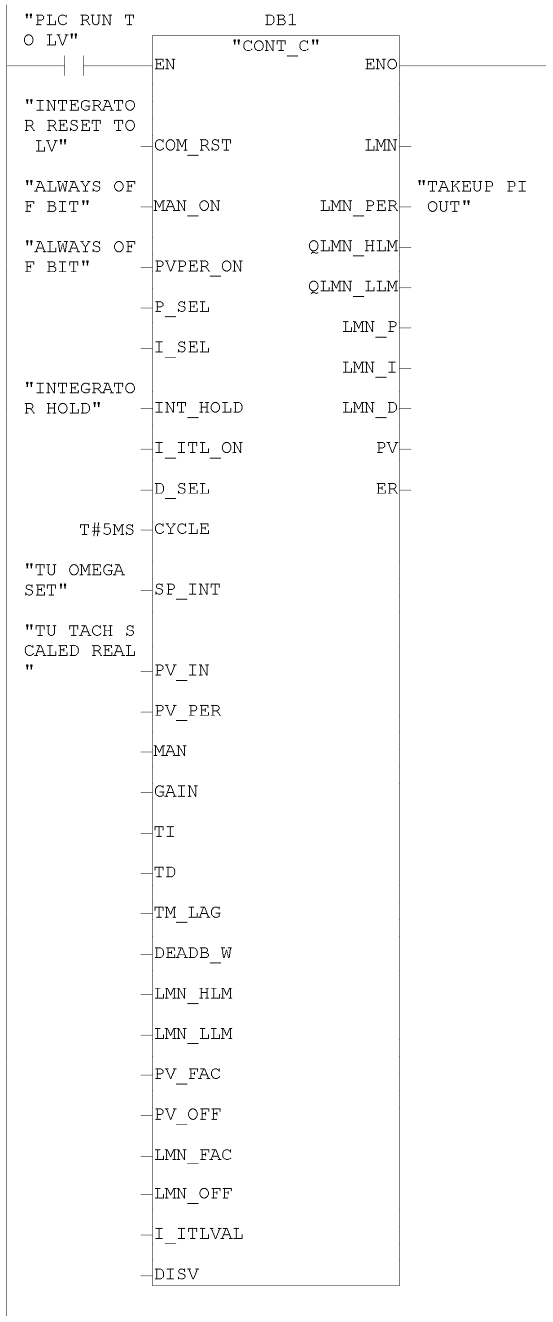


Network: 32   Create takeup angspeed setpoint value

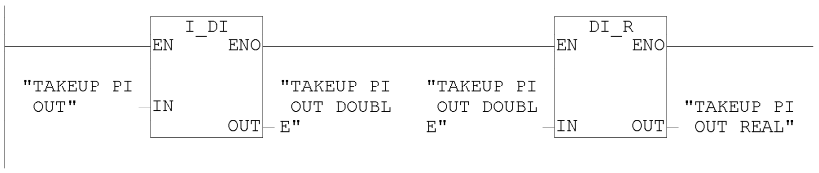
SECTION THREE - CONTROL CALCULATIONS
Takeup control starts here



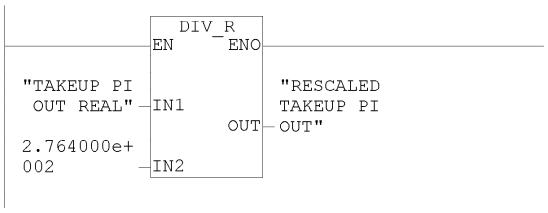
Network: 33   Takeup PID block



Network: 34   Convert takeup PI output to real

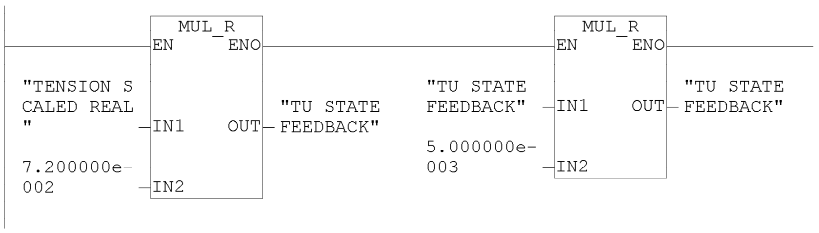


Network: 35   Divide real takeup PI out by intrinsic constant



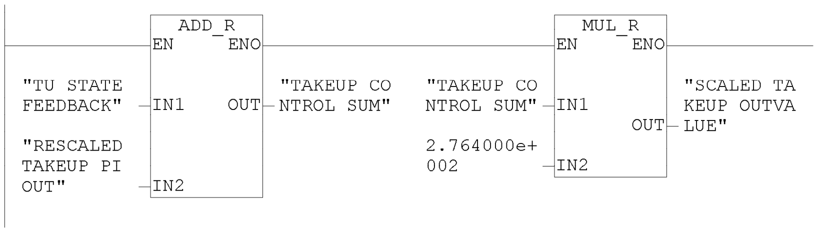
Network: 36   Takeup state variable manipulation

Multiply takeup radius by tension feedback, to obtain torque
Multiply torque value by state feedback gain value



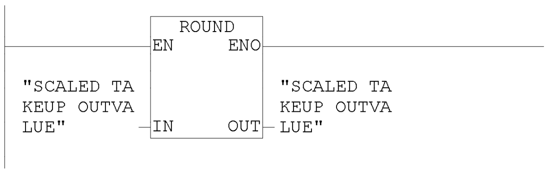
Network: 37   Create takeup output value

Add real scaled takeup PI out to scaled takeup state feedback
Multiply result by intrinsic constant



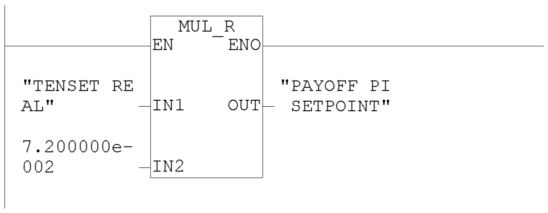
Network: 38   Convert takeup output value to double integer

Output rung will use the low order word of this value

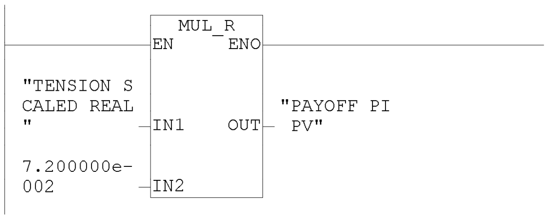


Network: 39   Multiply tension setpoint by payoff radius

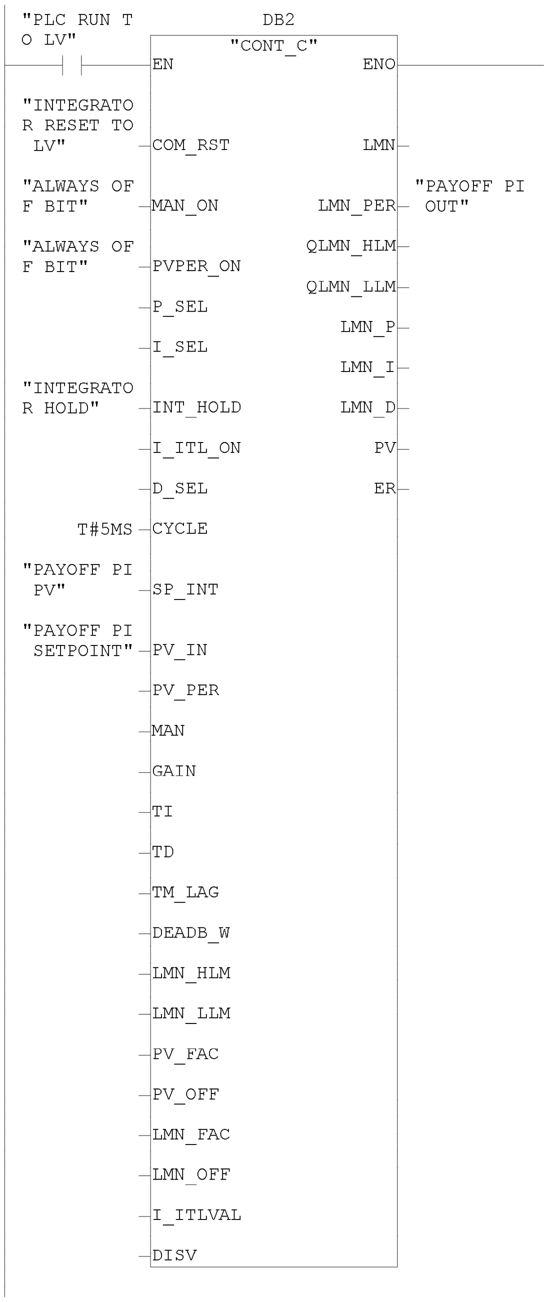
Payoff control starts here



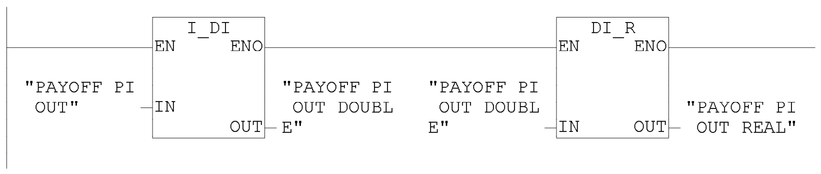
Network: 40   Multiply tension feedback by payoff radius



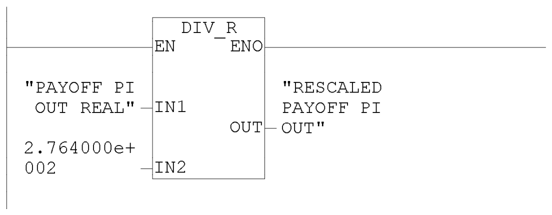
Network: 41   Payoff PID block



Network: 42   Convert payoff PI output to real

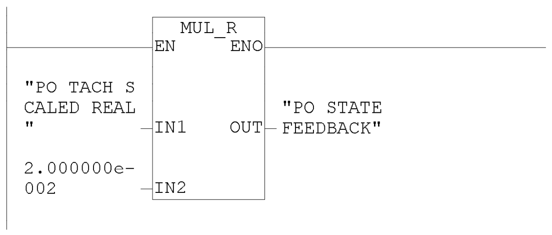


Network: 43   Divide real payoff PI out by instrinsic constant



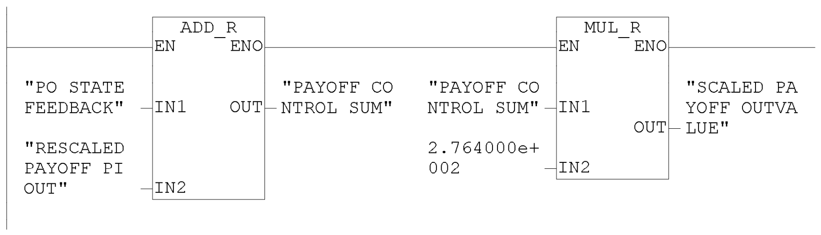
Network: 44   Payoff state variable manipulation

Multiply payoff speed by state feedback gain value



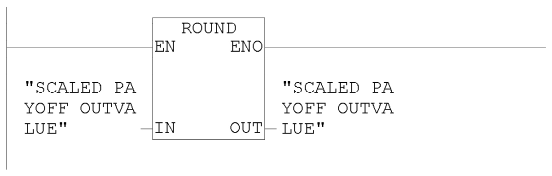
Network: 45   Create payoff output value

Add real scaled payoff PI out to scaled payoff state feedback
Multiply result by intrinsic constant

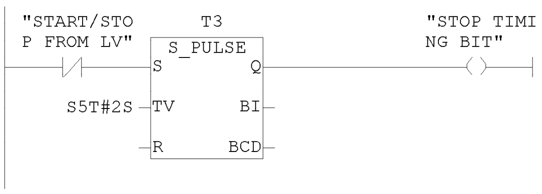


Network: 46   Convert payoff output value to double integer

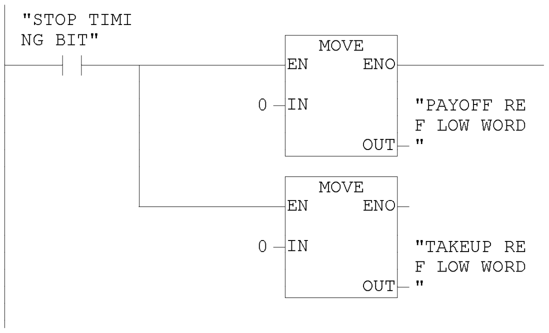
Output rung will use the low order word of this value



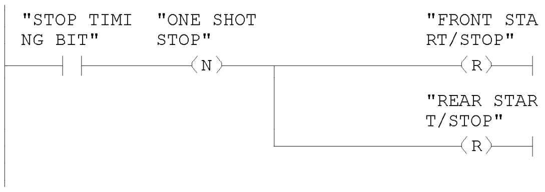
Network: 47 Drive stop command for normal use



Network: 48 Zero setpoint before normal stop command

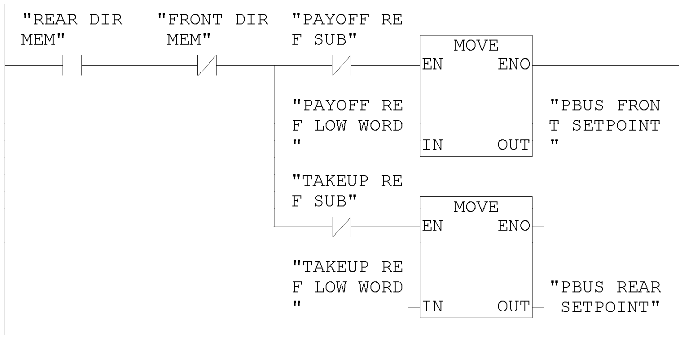


Network: 49 Stop drives with timing bit reset



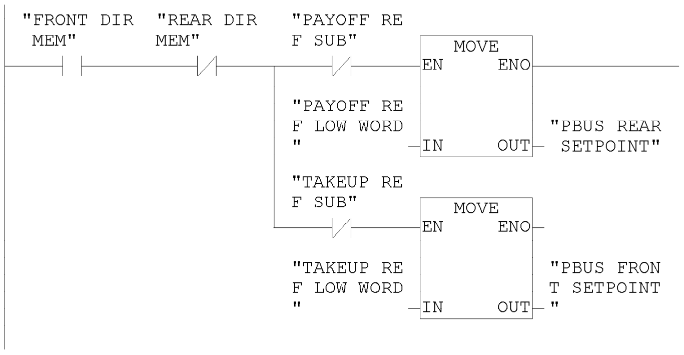
Network: 50 Move references to drives - forward operation

If rear is takeup and front is payoff, move drive references
to appropriate locations (running forward)

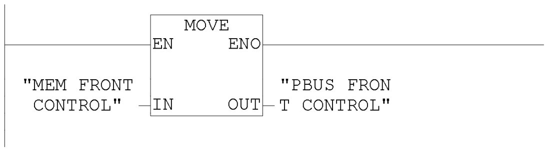


Network: 51   Move references to drives - reverse operation

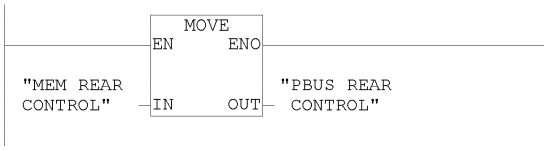
If rear is takeup and front is payoff, move drive references
to appropriate locations (running forward)



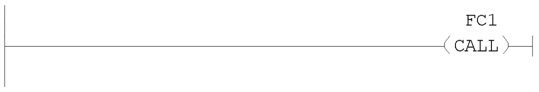
Network: 52   Move control word to front drive



Network: 53   Move control word to rear drive

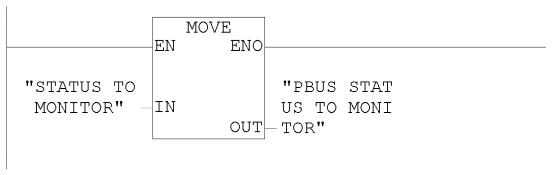


Network: 54   Call RTPC add-on code

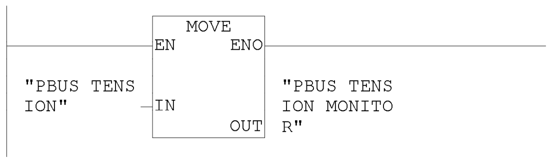


Block: FC1   ADDITIONAL PLC FUNCTION FOR RIDETHROUGH PC

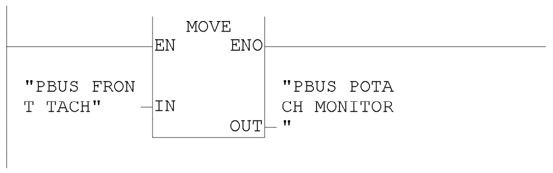
Network: 1   Move status word to RTPC



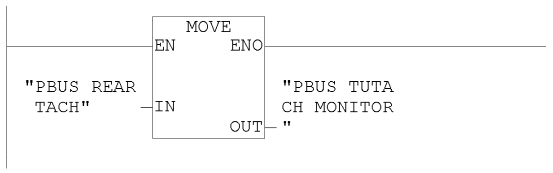
Network: 2   Move tension word to RTPC



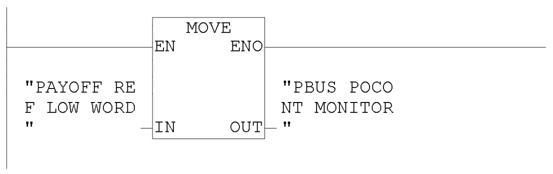
Network: 3   Move payoff tach word to RTPC



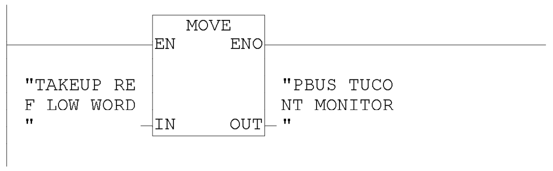
Network: 4   Move takeup tach word to RTPC



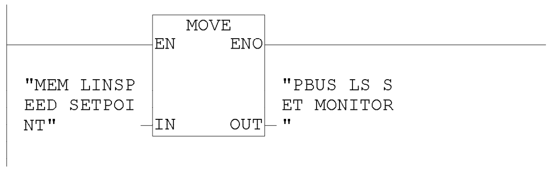
Network: 5   Move payoff reference word to RTPC



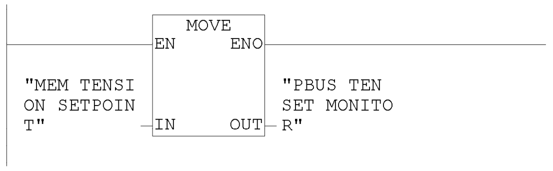
Network: 6   Move takeup reference word to RTPC



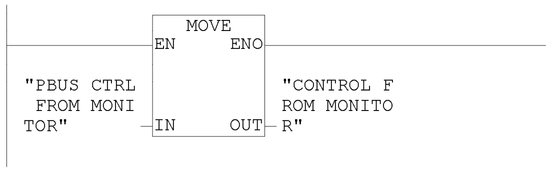
Network: 7   Move linespeed setpoint word to RTPC



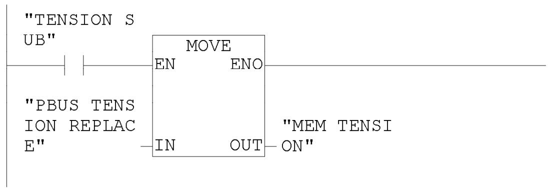
Network: 8   Move tension setpoint word to RTPC



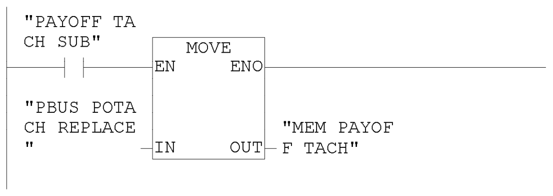
Network: 9   Move control word from RTPC



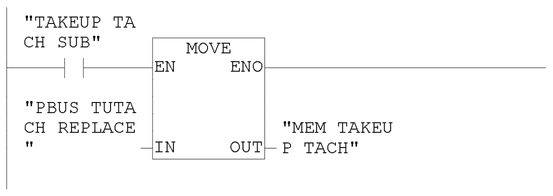
Network: 10   Move tension replace word from RTPC



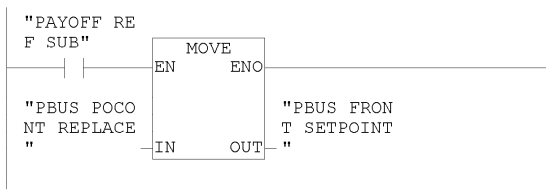
Network: 11 Move payoff tach replace word from RTPC



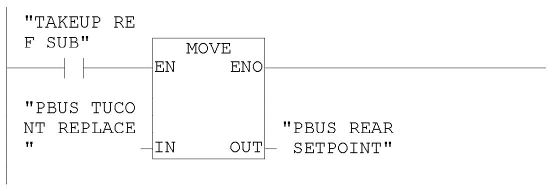
Network: 12 Move takeup tach replace word from RTPC



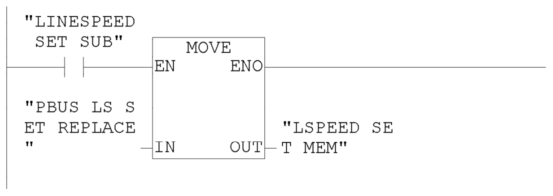
Network: 13 Move payoff reference replace word from RTPC



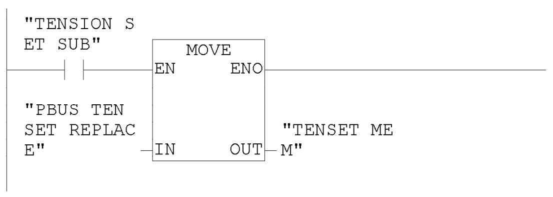
Network: 14 Move takeup reference replace word from RTPC



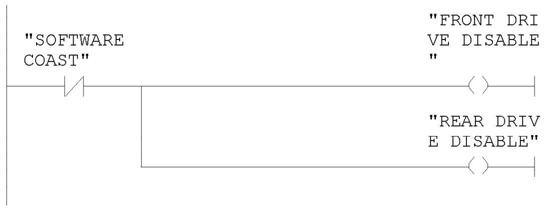
Network: 15 Move linespeed set replace word from RTPC



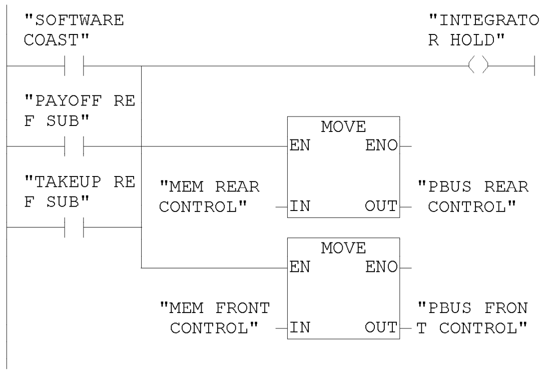
Network: 16 Move tension set replace word from RTPC



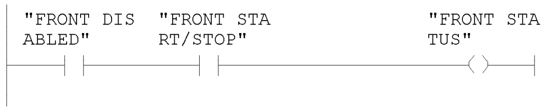
Network: 17 Disable drives on software coast command



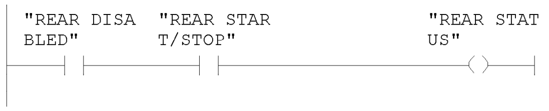
Network: 18 Hold integrator and output new control words



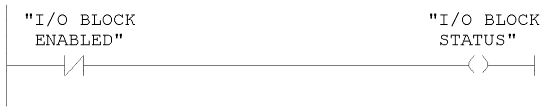
Network: 19   Front drive dropout status bit



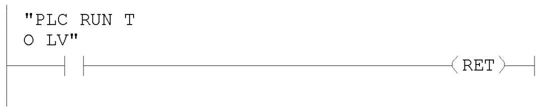
Network: 20   Rear drive dropout status bit



Network: 21   I/O block dropout status bit



Network: 22   Return to main program



Appendix I
Additional Ridethrough PC Software Flowcharts

This appendix contains additional flowcharts detailing the software structure for the Ridethrough PC program. They are provided as an addendum to the flowcharts given in Chapter 3, "Software Design".

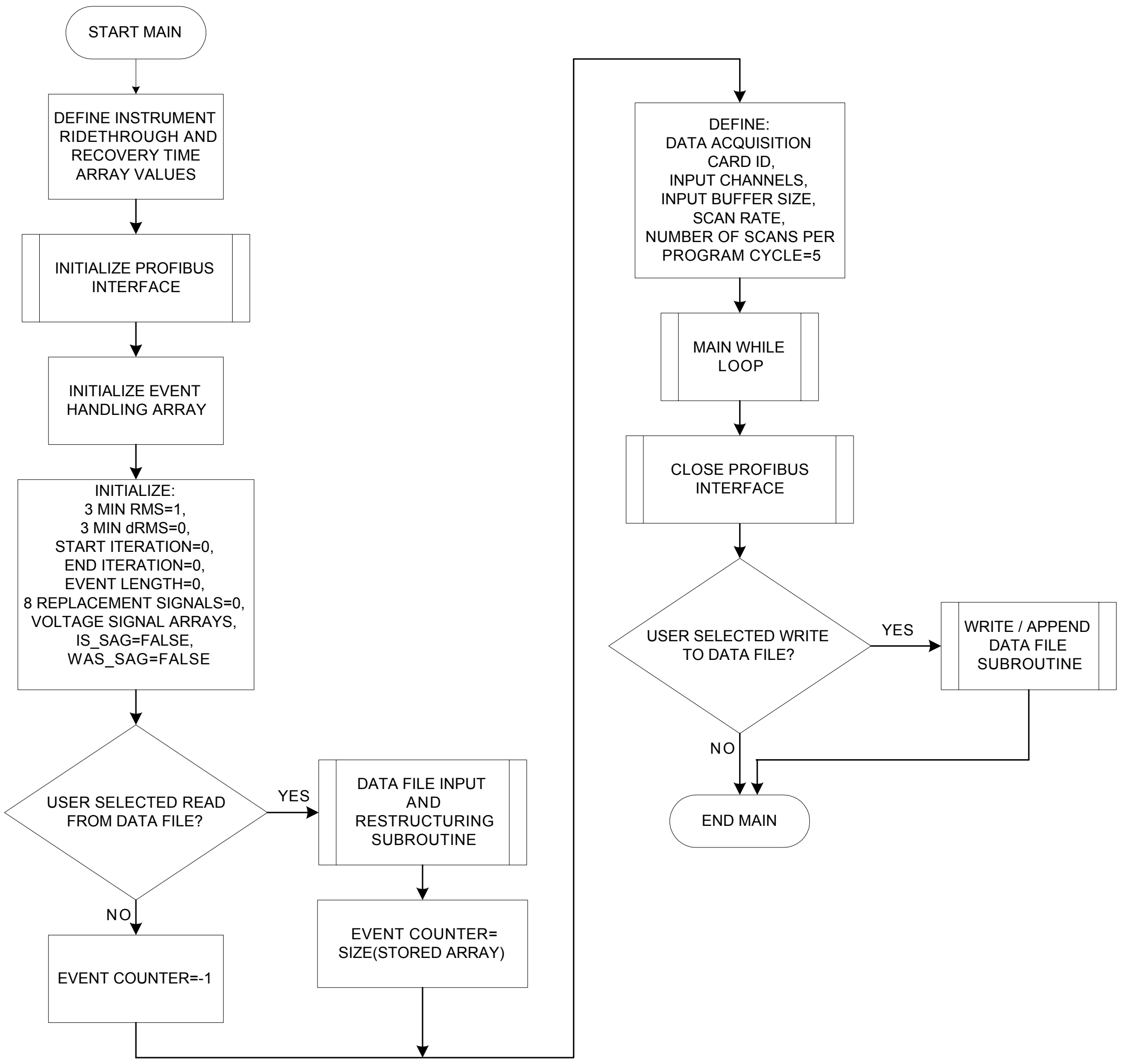

START MAIN
DEFINE INSTRUMENT RIDETHROUGH AND RECOVERY TIME ARRAY VALUES
INITIALIZE PROFIBUS INTERFACE
INITIALIZE EVENT HANDLING ARRAY
INITIALIZE: 3 MIN RMS=1, 3 MIN dRMS=0, START ITERATION=0, END ITERATION=0, EVENT LENGTH=0, 8 REPLACEMENT SIGNALS=0, VOLTAGE SIGNAL ARRAYS, IS_SAG=FALSE, WAS_SAG=FALSE
USER SELECTED READ FROM DATA FILE?
YES
DATA FILE INPUT AND RESTRUCTURING SUBROUTINE
EVENT COUNTER= SIZE(STORED ARRAY)
NO
EVENT COUNTER=-1
DEFINE: DATA ACQUISITION CARD ID, INPUT CHANNELS, INPUT BUFFER SIZE, SCAN RATE, NUMBER OF SCANS PER PROGRAM CYCLE=5
MAIN WHILE LOOP
CLOSE PROFIBUS INTERFACE
USER SELECTED WRITE TO DATA FILE?
YES
WRITE / APPEND DATA FILE SUBROUTINE
NO
END MAIN

Figure I.1 Flowchart for main program of ridethrough PC.

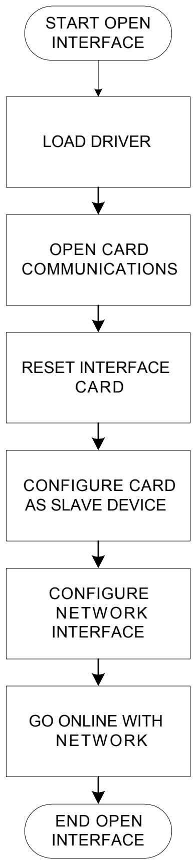


Figure I.2 Flowchart for Profibus open interface routine.

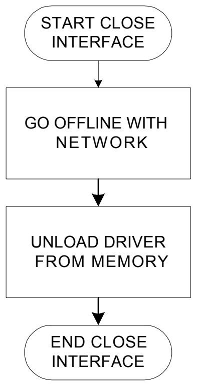


Figure I.3 Flowchart for Profibus close interface routine.

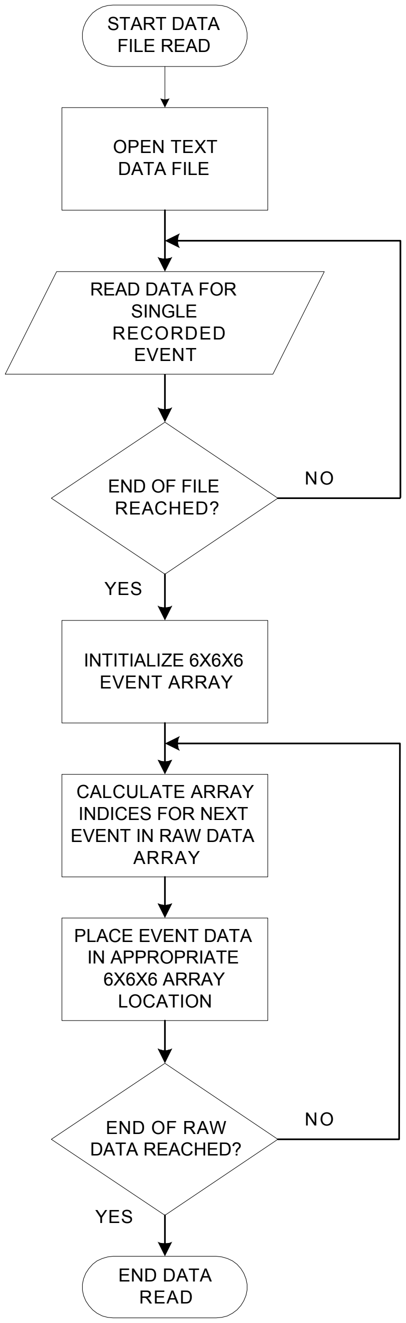

START DATA FILE READ
OPEN TEXT DATA FILE
READ DATA FOR SINGLE RECORDED EVENT
END OF FILE REACHED?
NO
YES
INTITIALIZE 6X6X6 EVENT ARRAY
CALCULATE ARRAY INDICES FOR NEXT EVENT IN RAW DATA ARRAY
PLACE EVENT DATA IN APPROPRIATE 6X6X6 ARRAY LOCATION
END OF RAW DATA REACHED?
NO
YES
END DATA READ

Figure I.4 Flowchart for data file read routine.

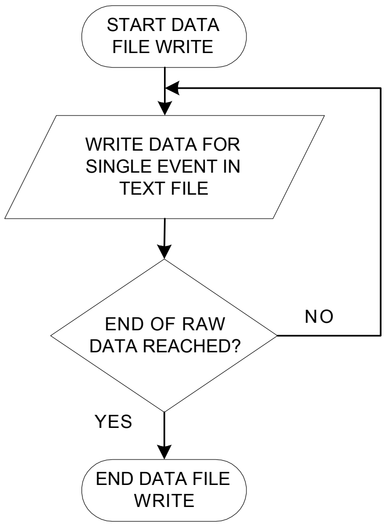


Figure I.5 Flowchart for data file write routine.

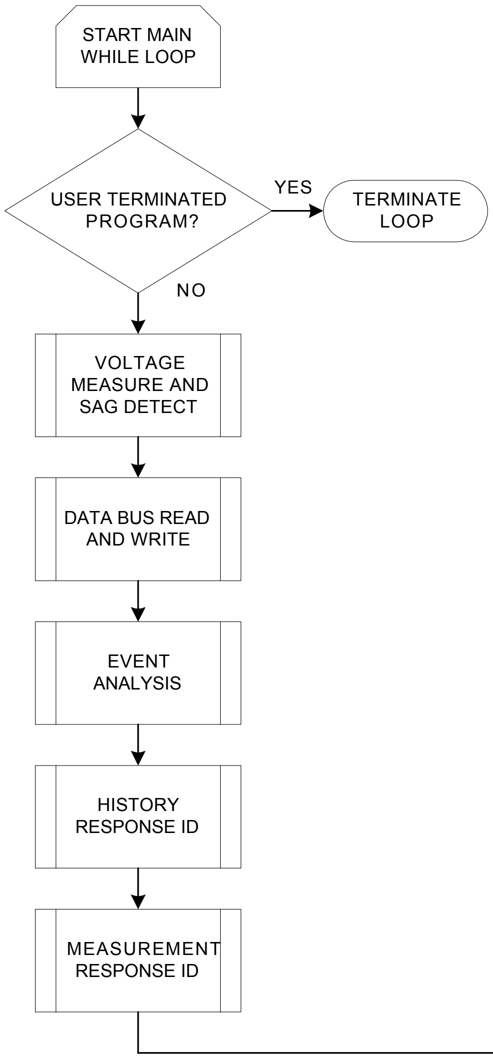


Figure I.6 Flowchart for main loop of ridethrough PC software.

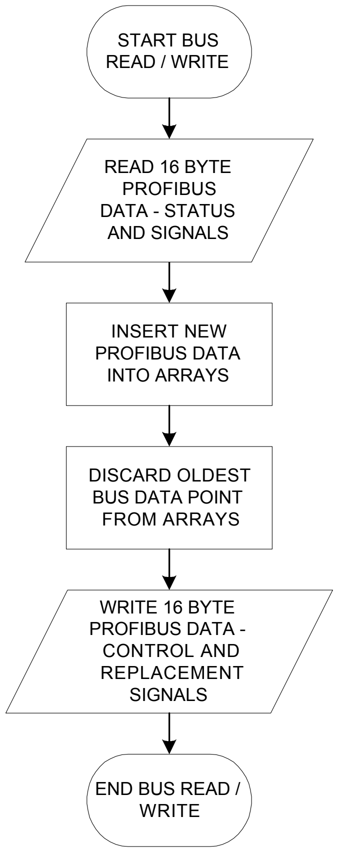

START BUS READ / WRITE
READ 16 BYTE PROFIBUS DATA - STATUS AND SIGNALS
INSERT NEW PROFIBUS DATA INTO ARRAYS
DISCARD OLDEST BUS DATA POINT FROM ARRAYS
WRITE 16 BYTE PROFIBUS DATA - CONTROL AND REPLACEMENT SIGNALS
END BUS READ / WRITE

Figure I.7 Flowchart for Profibus read and write routine.

Appendix J
Main LabView Code For Ridethrough PC

This appendix contains the main LabView code used in the Ridethrough PC. Structure windows that appear to be disconnected are secondary cases of the connected windows immediately adjacent to them.

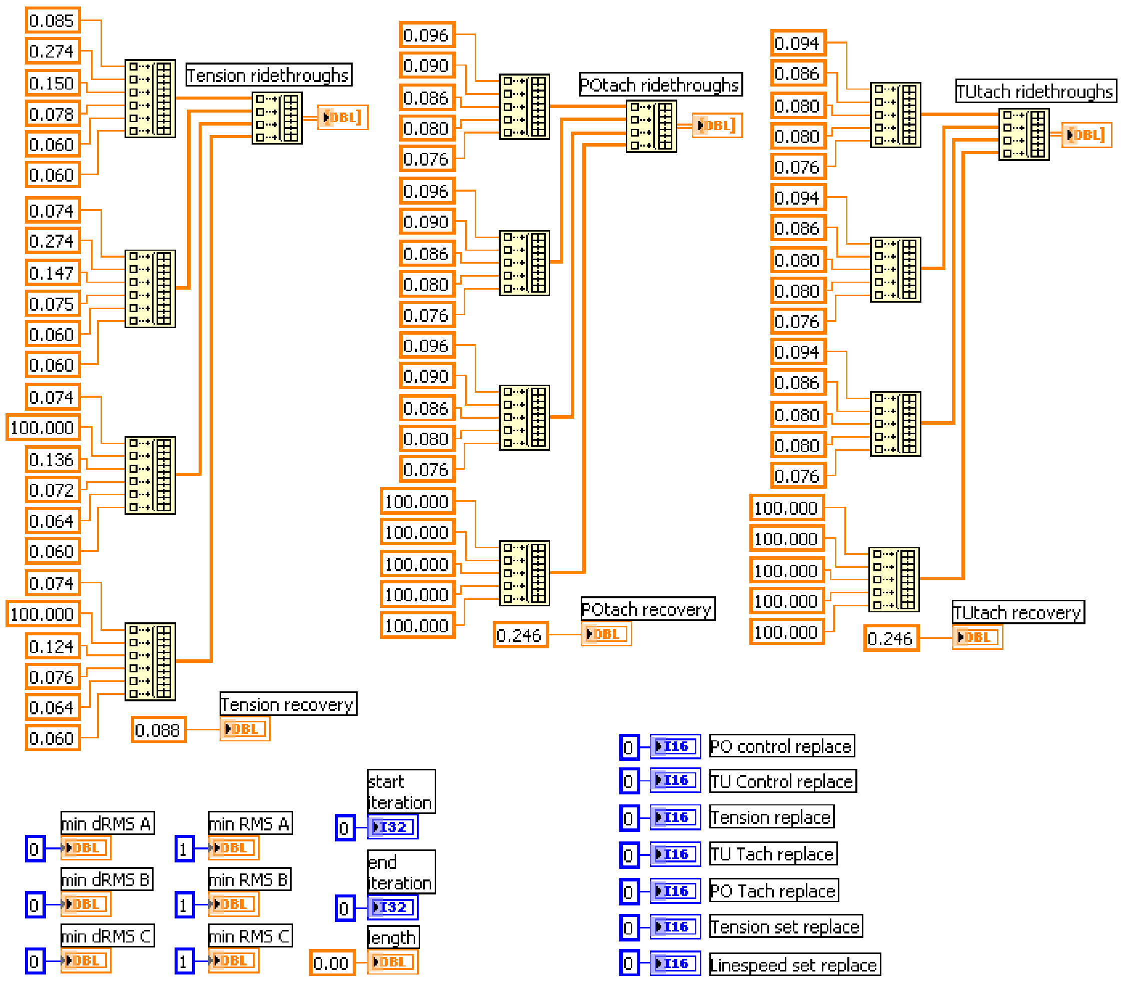


Figure J.1 Main Labview code - variable initialization before main loop.

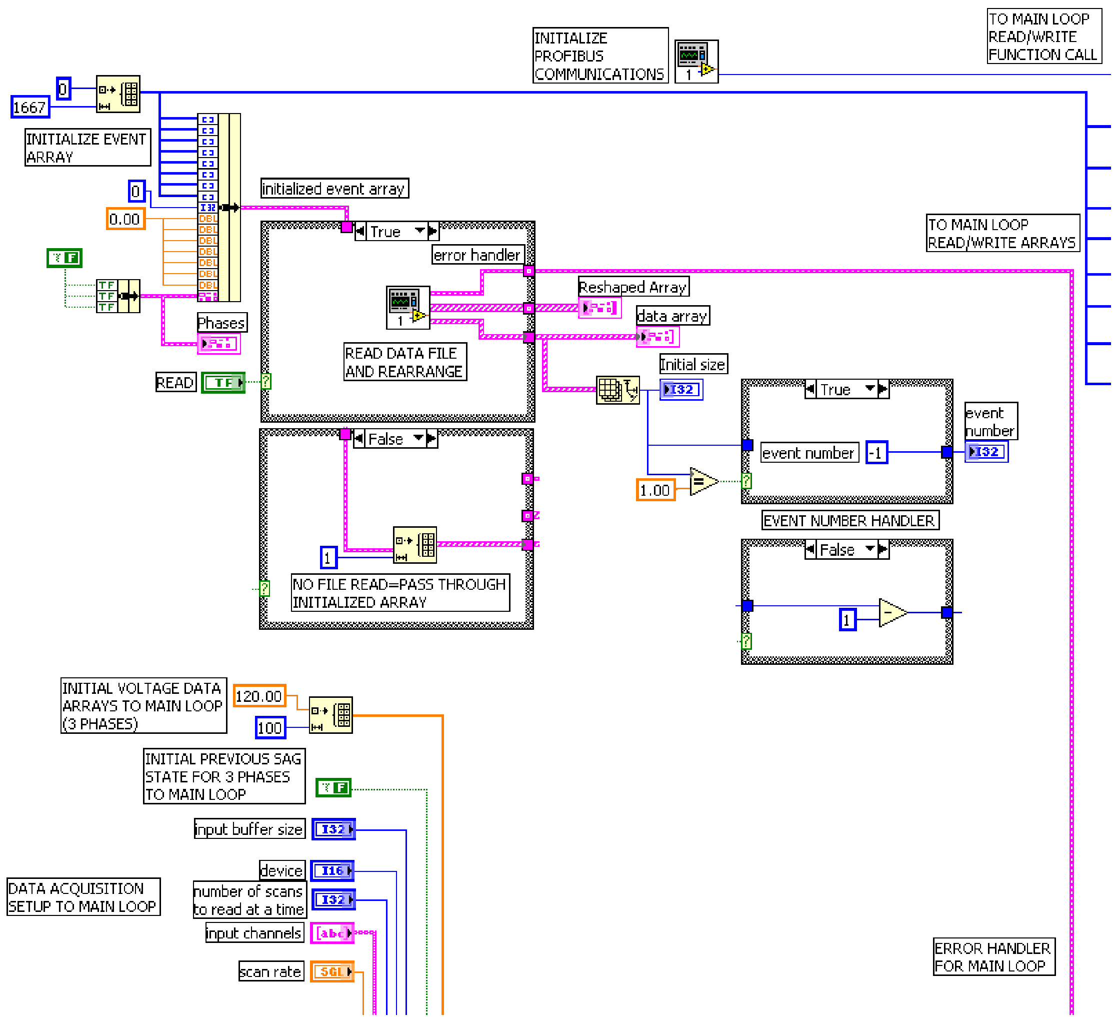

INITIALIZE PROFIBUS COMMUNICATIONS
TO MAIN LOOP READ/WRITE FUNCTION CALL
1667
0
INITIALIZE EVENT ARRAY
initialized event array
0
0.00
True
error handler
Reshaped Array
data array
READ DATA FILE AND REARRANGE
Phases
READ
Initial size
TO MAIN LOOP READ/WRITE ARRAYS
True
event number
-1
event number
1.00
EVENT NUMBER HANDLER
False
1
NO FILE READ=PASS THROUGH INITIALIZED ARRAY
False
1
INITIAL VOLTAGE DATA ARRAYS TO MAIN LOOP (3 PHASES)
120.00
100
INITIAL PREVIOUS SAG STATE FOR 3 PHASES TO MAIN LOOP
input buffer size
device
DATA ACQUISITION SETUP TO MAIN LOOP
number of scans to read at a time
input channels
scan rate
ERROR HANDLER FOR MAIN LOOP

Figure J.2 Main LabView code - data read, array initialization, and hardware setup before main loop.

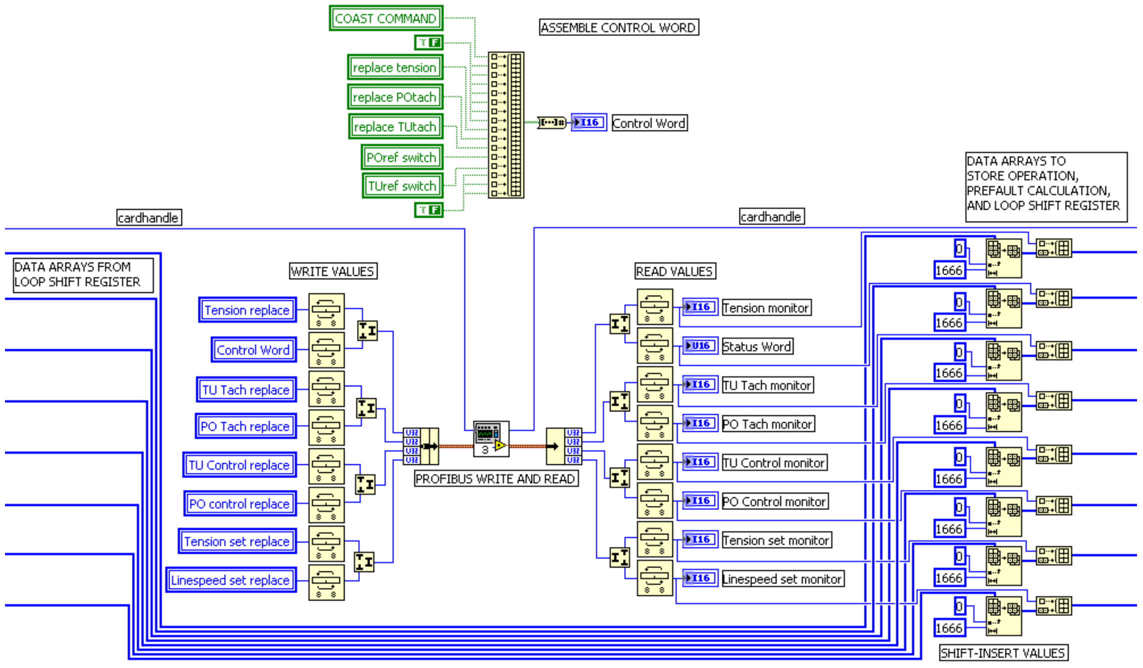


Figure J.3 Main LabView code - Profibus I/O and data monitoring.

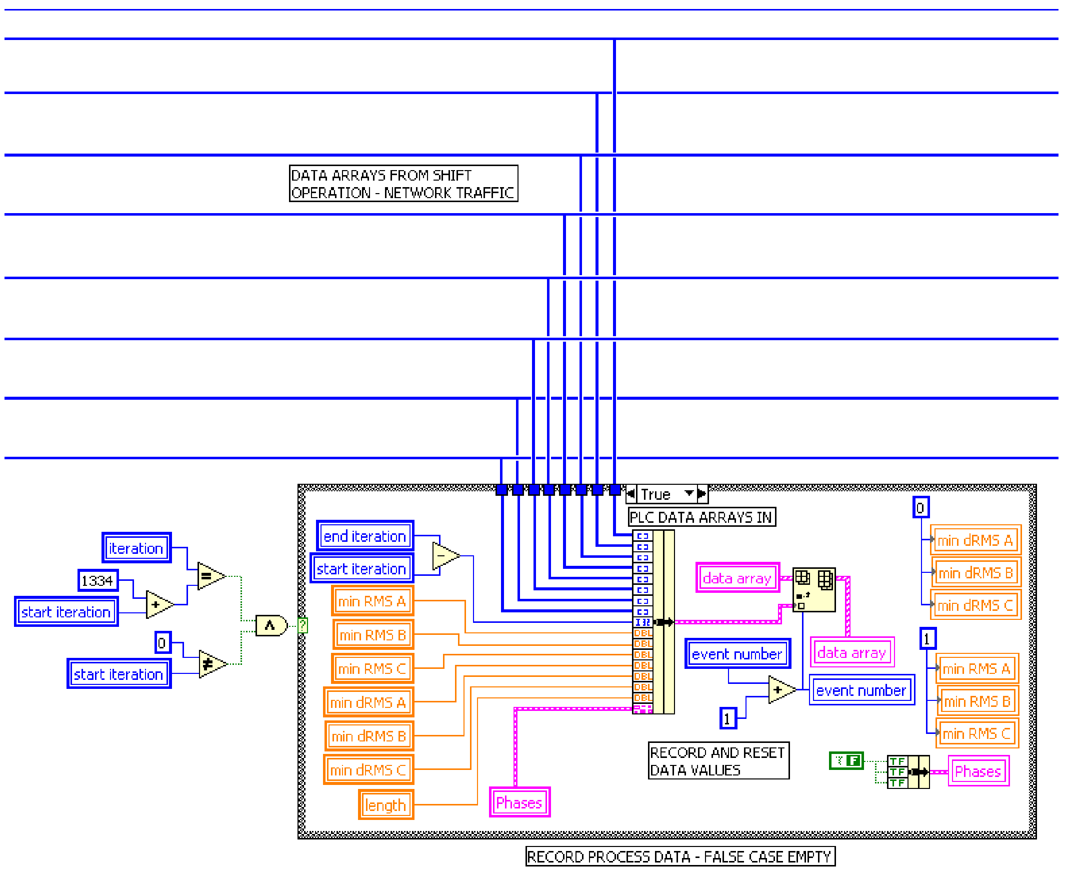


Figure J.4 Main LabView code - event data capture.

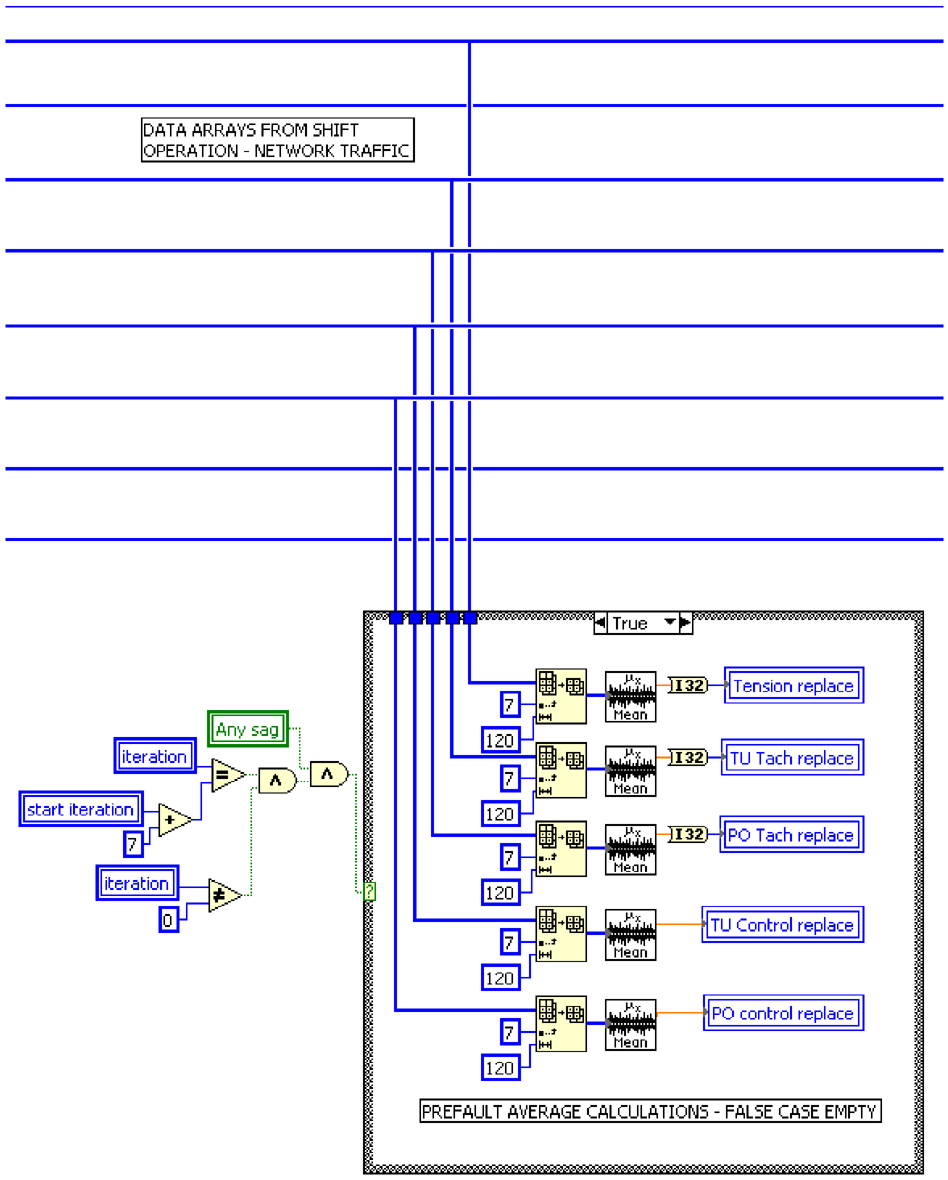


Figure J.5 Main LabView code - prefault average calculation.

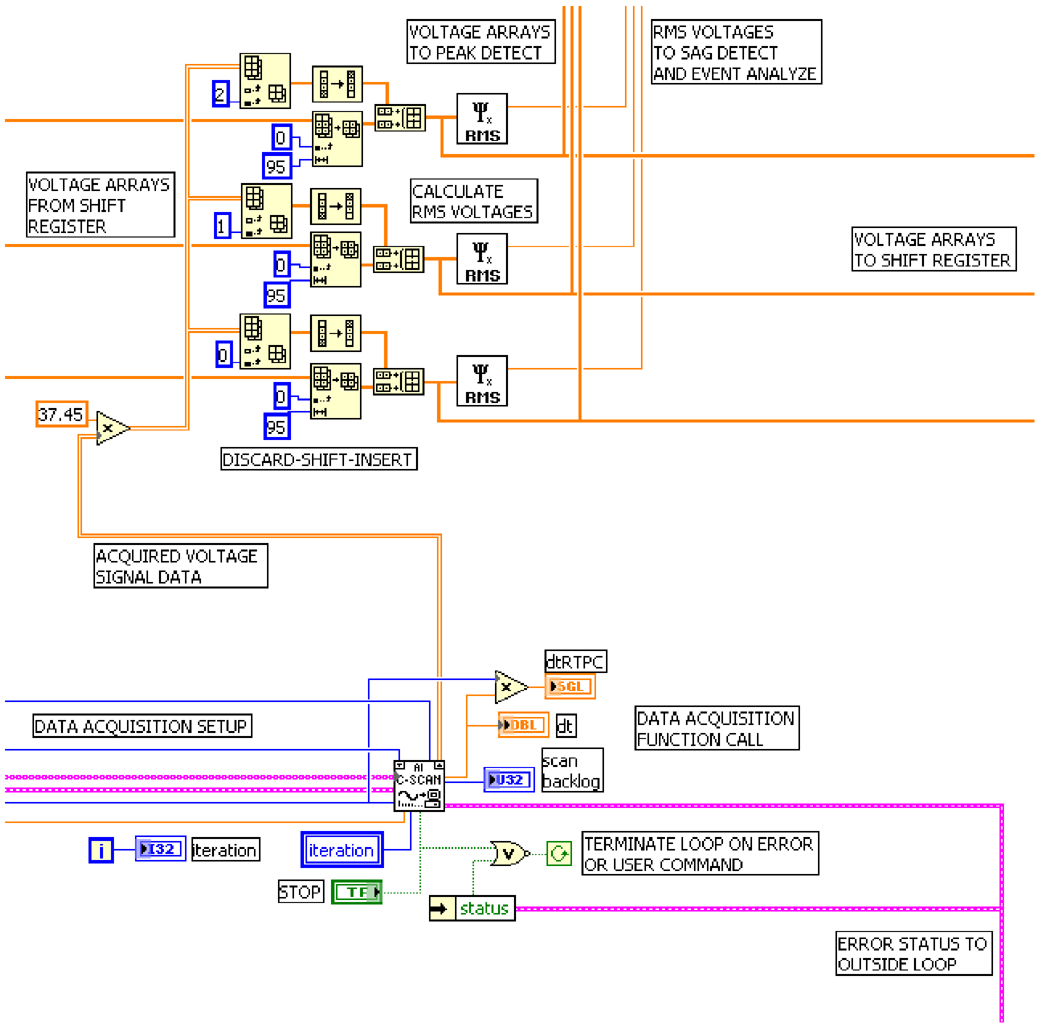

VOLTAGE ARRAYS TO PEAK DETECT
RMS VOLTAGES TO SAG DETECT AND EVENT ANALYZE
VOLTAGE ARRAYS FROM SHIFT REGISTER
CALCULATE RMS VOLTAGES
VOLTAGE ARRAYS TO SHIFT REGISTER
RMS
37.45
DISCARD-SHIFT-INSERT
ACQUIRED VOLTAGE SIGNAL DATA
dtRTPC
SGL
DATA ACQUISITION SETUP
DBL
dt
DATA ACQUISITION FUNCTION CALL
AI C-SCAN
U32
scan backlog
I32
iteration
iteration
TERMINATE LOOP ON ERROR OR USER COMMAND
STOP
TF
status
ERROR STATUS TO OUTSIDE LOOP

Figure J.6 Main LabView code - data acquisition and voltage data handling.

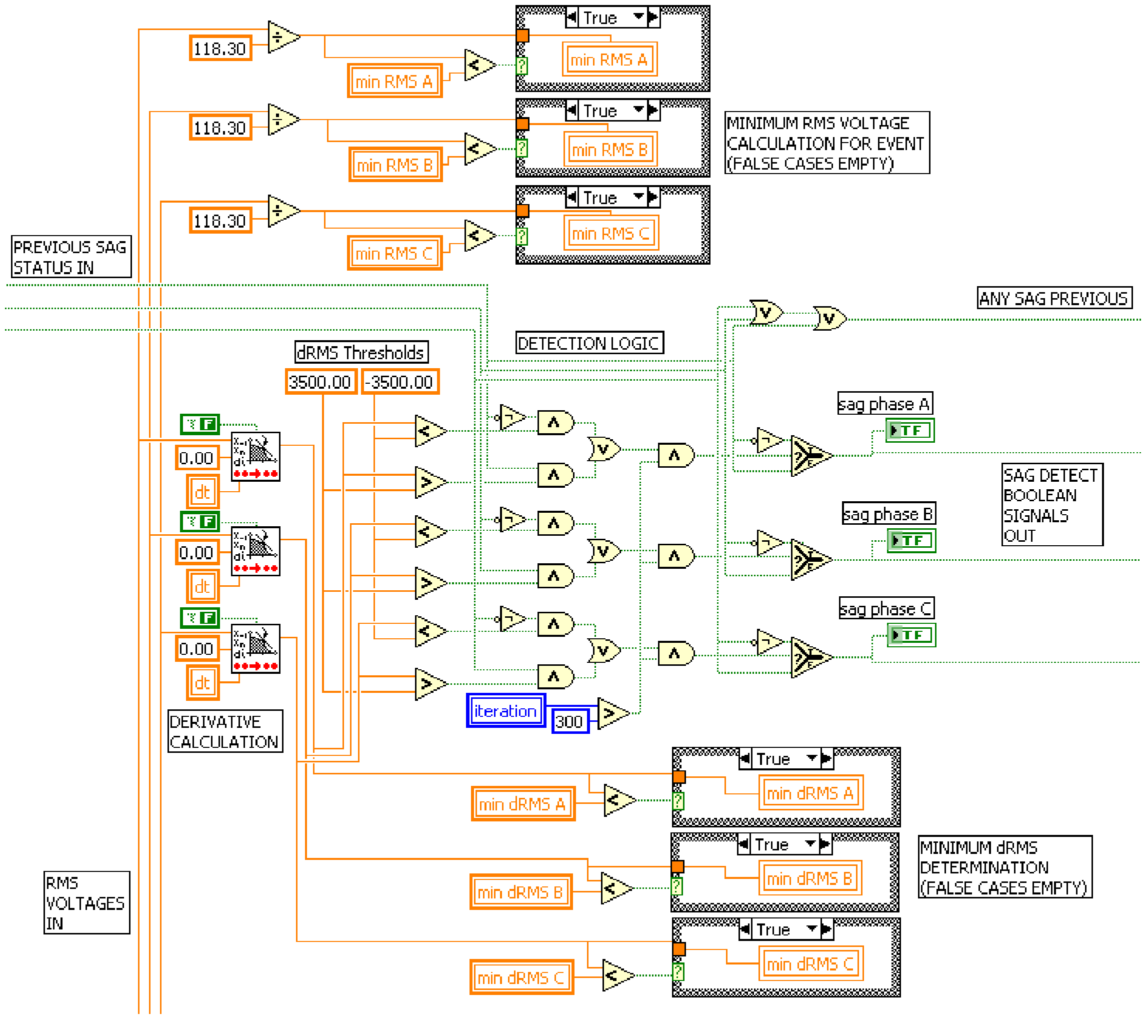
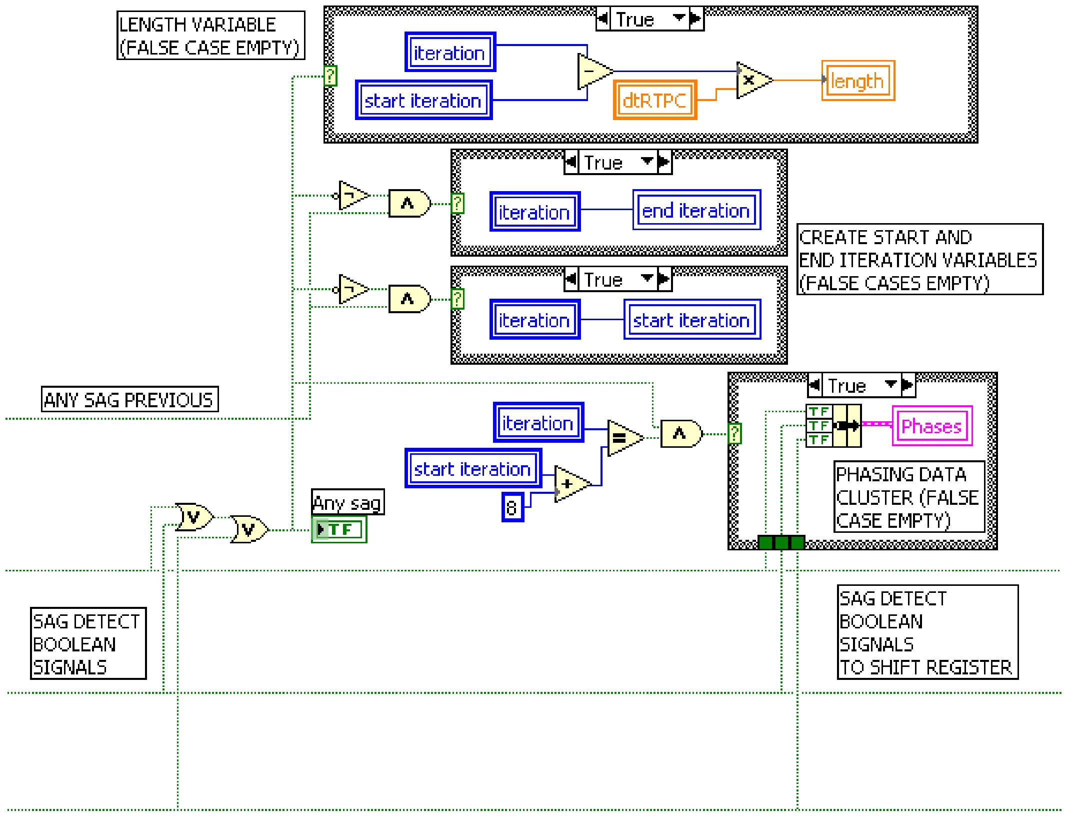


Figure J.7 Main LabView code - voltage sag detection, minimum RMS voltage, and minimum RMS voltage derivative determination.



LENGTH VARIABLE (FALSE CASE EMPTY)
True
iteration
start iteration
dtRTPC
length
True
iteration
end iteration
CREATE START AND END ITERATION VARIABLES (FALSE CASES EMPTY)
True
iteration
start iteration
ANY SAG PREVIOUS
True
iteration
start iteration
8
Phases
PHASING DATA CLUSTER (FALSE CASE EMPTY)
Any sag
SAG DETECT BOOLEAN SIGNALS
SAG DETECT BOOLEAN SIGNALS TO SHIFT REGISTER

Figure J.8 Main LabView code - event analysis instructions.

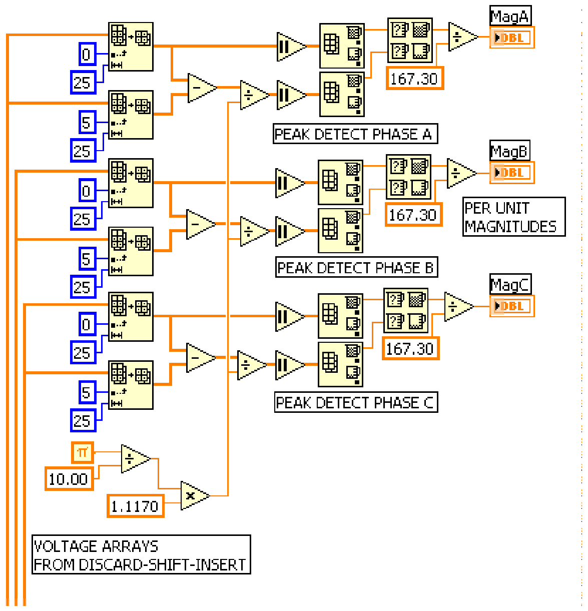


Figure J.9 Main LabView code - peak voltage detection instructions.

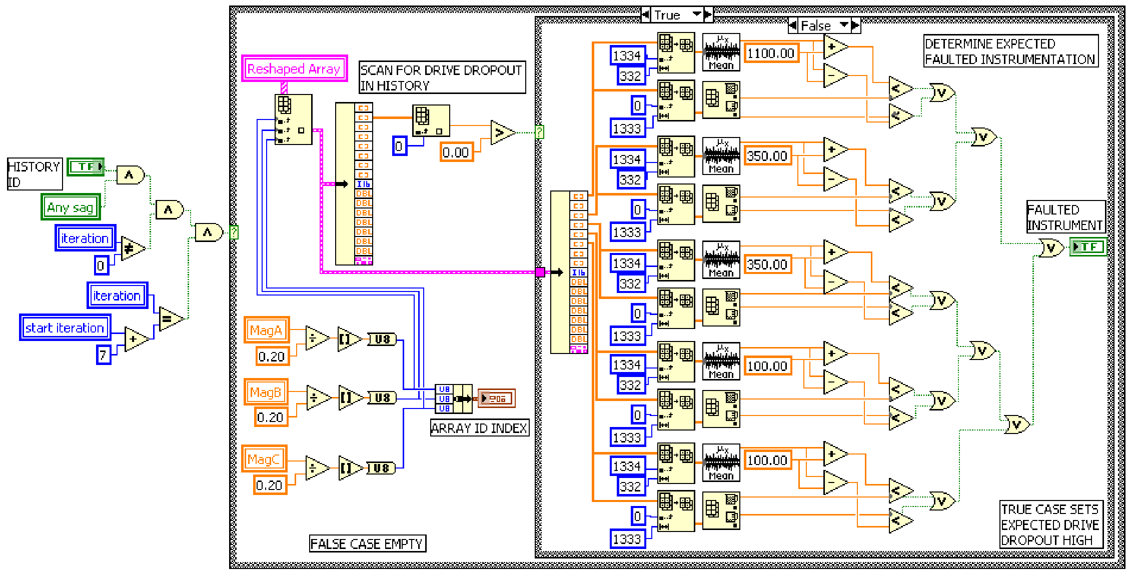


Figure J.10 Main LabView code - expected process response identification using recorded process history.

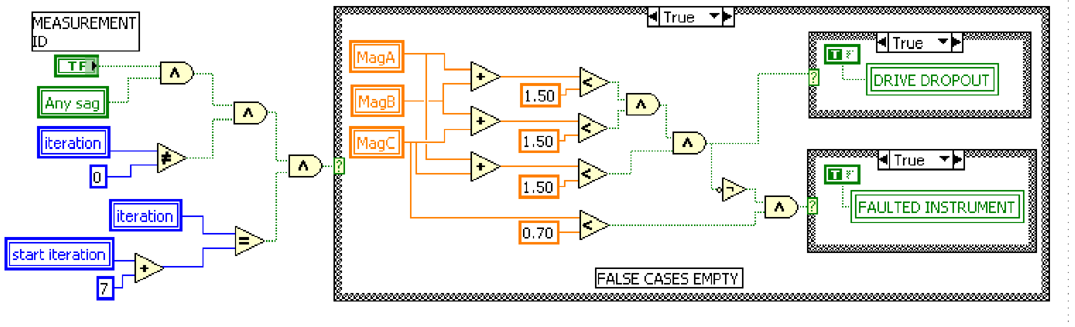


Figure J.11 Main LabView code - expected process response identification using formal region boundary definitions.

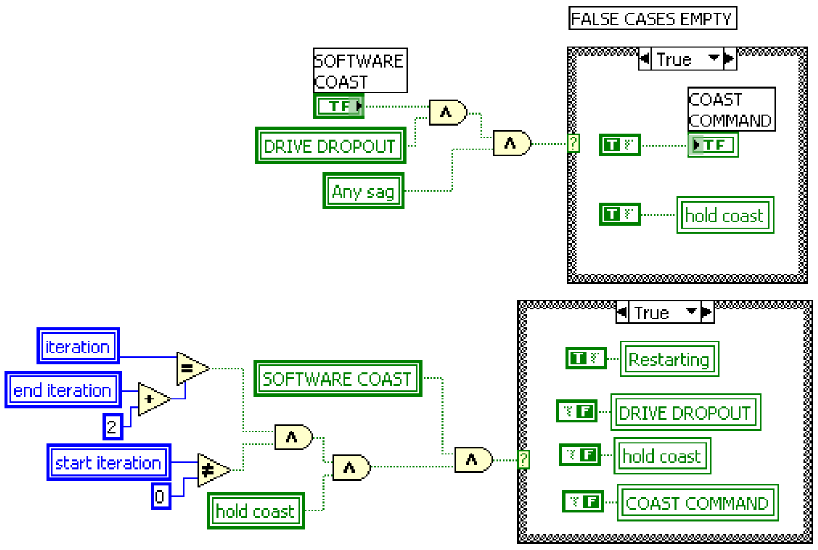


Figure J.12 Main LabView code - software coast agorithm instruction set.

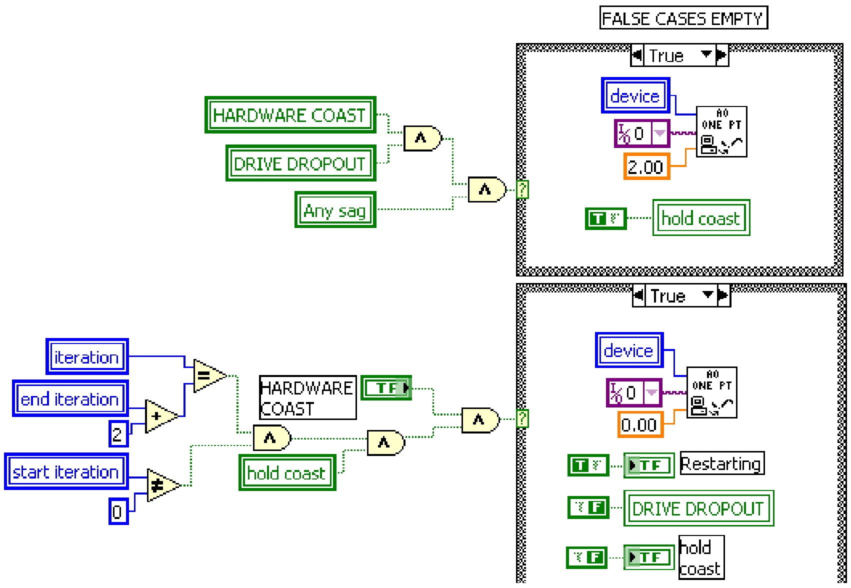


Figure J.13 Main LabView code - hardware coast algorithm instruction set.

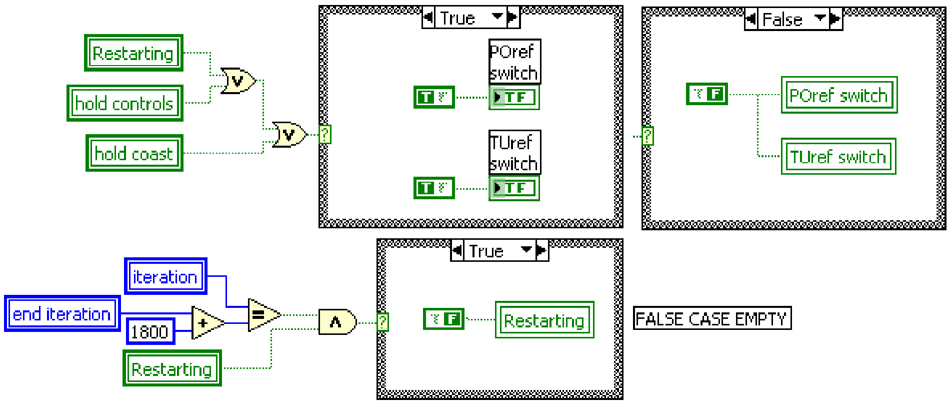


Figure J.14 Main LabView code - restarting instruction set.

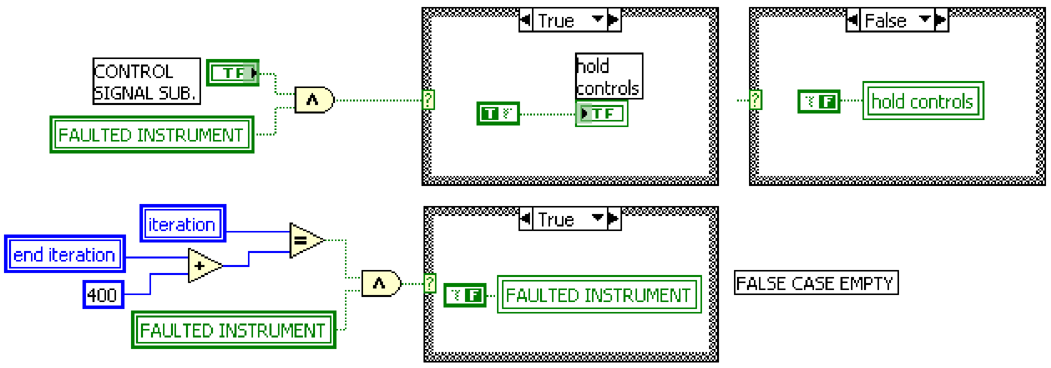


Figure J.15 Main LabView code - control signal substitution and status clearing instruction set.

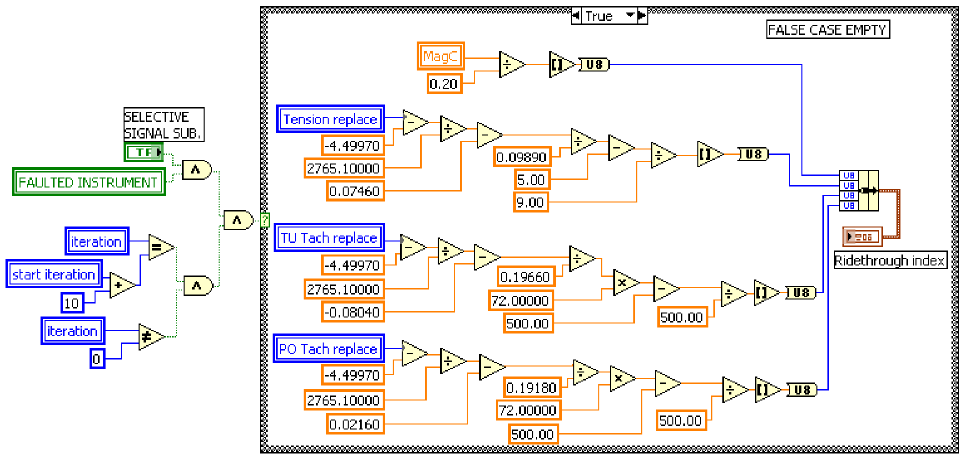


Figure J.16 Main LabView code - generation of ridethrough time index for selective signal substitution algorithm.

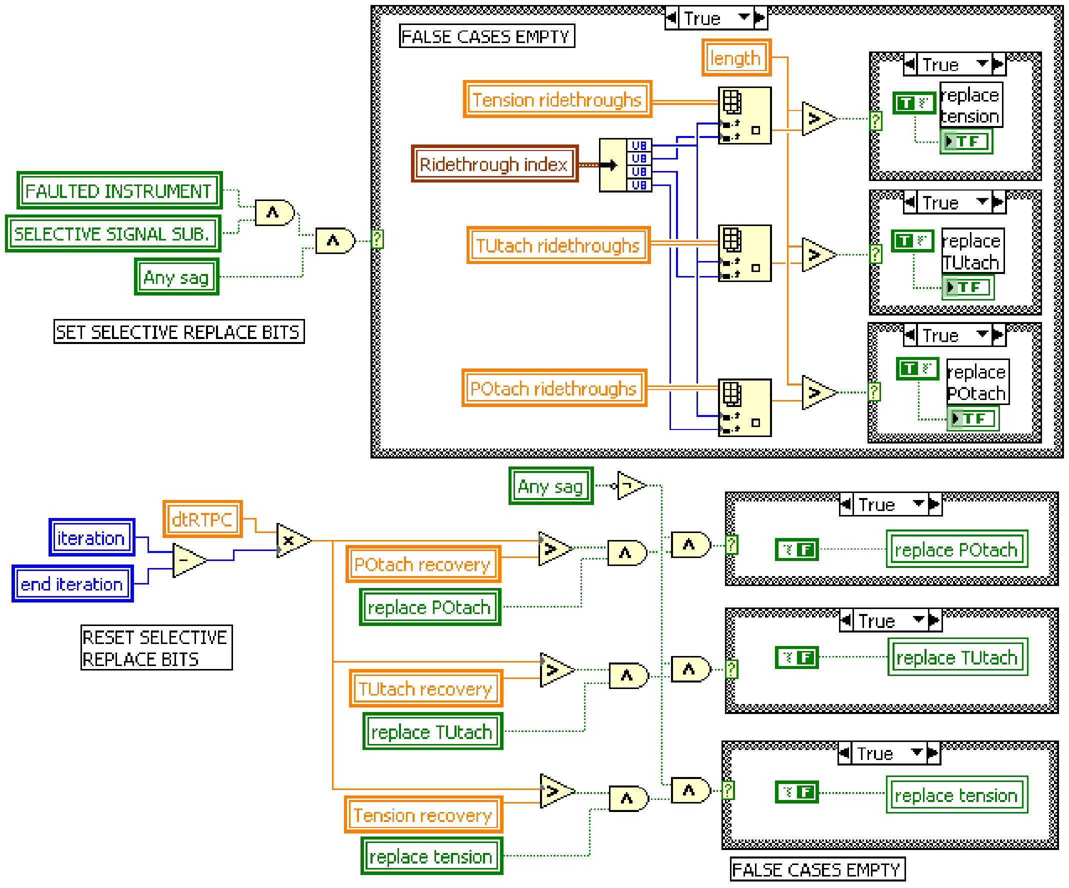


Figure J.17 Main LabView code - selective signal substitution algorithm replace bit set and reset instructions.

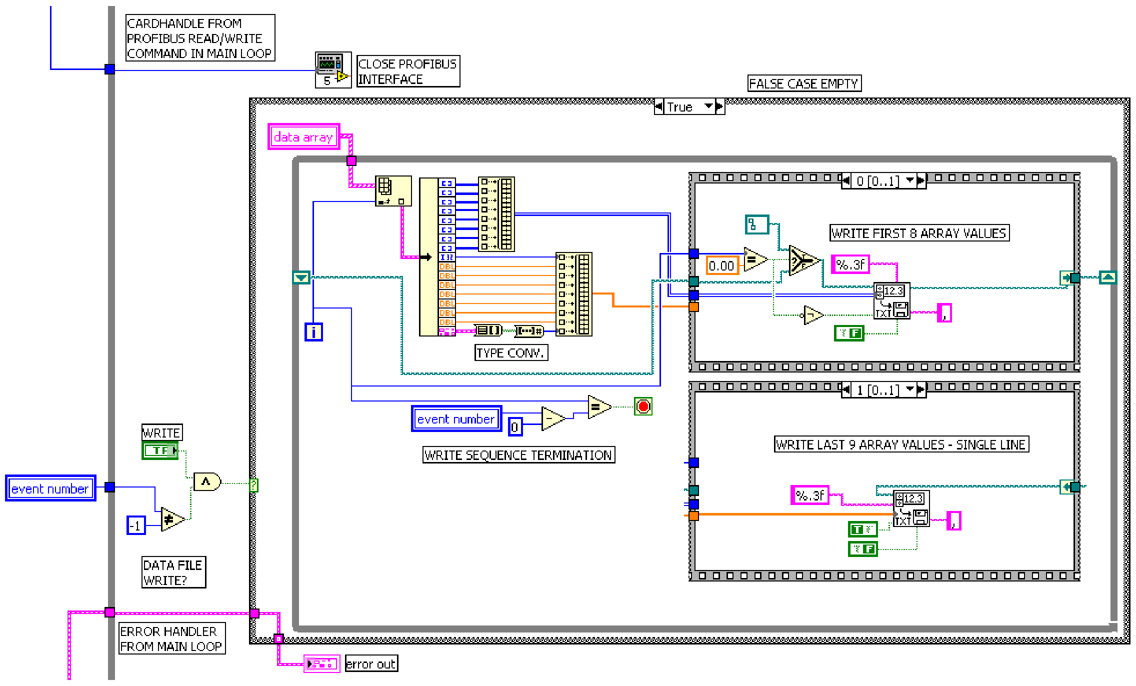

CARDHANDLE FROM PROFIBUS READ/WRITE COMMAND IN MAIN LOOP
CLOSE PROFIBUS INTERFACE
FALSE CASE EMPTY
True
data array
0 [0..1]
WRITE FIRST 8 ARRAY VALUES
0.00
%.3f
TYPE CONV.
event number
0
WRITE SEQUENCE TERMINATION
1 [0..1]
WRITE LAST 9 ARRAY VALUES - SINGLE LINE
%.3f
WRITE
event number
-1
DATA FILE WRITE?
ERROR HANDLER FROM MAIN LOOP
error out

Figure J.18 Main LabView code - data file write and Profibus interface close function call after termination of main loop.

BIBLIOGRAPHY

[1] "Solving the Fast Clock Problem", *Power Quality Testing Network Solutions*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.3, September 1994.

[2] "Electronic Digital Clock Performance During Steady-State and Dynamic Power Disturbances", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.17, March 1994.

[3] "Undervoltage Ride-Through Performance of Off-the-Shelf Personal Computers", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.7, December 1992.

[4] "Low-Voltage Ride-Through Performance of a Personal Computer Power Supply", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.11, March 1993.

[5] "IEEE Recommended Practice for Monitoring Electric Power Quality", *IEEE Standard 1159-1995*, New York, The Institute of Electrical and Electronics Engineers, Inc., 1995.

[6] "Power Quality Considerations for Adjustable Speed Drives", EPRI, 1991.

[7] "IEEE Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems", *IEEE Standard 519-1992*, New York, The Institute of Electrical and Electronics Engineers, Inc., 1992.

[8] "Article 517 - Health Care Facilities", *The National Electrical Code, 2008 Edition*, Quincy, MA, National Fire Prevention Association, Inc., 2008, pp.425-442.

[9] John P. Wagner. "The Cost of Power Quality in the ITE Industry", *Proceedings of the Second International Conference on Power Quality, PQA, '92: End Use Applications and Perspectives*, Atlanta, GA, EPRI, 1992, pp.A-3:1-A-3:5.

[10] Ambra Sannino, Michelle Ghans Miller, Math H.J. Bollen. "Overview of Voltage Sag Mitigation", *Proceedings of the IEEE 2000 Power Engineering Society Winter Meeting*, Vol. 4., January 2000, pp.2872-2878.

[11] Michael J. Sullivan, Terry Vardell, Mark Johnson. "Power Interruption Costs to Industrial and Commercial Consumers of Electricity" *IEEE Transactions on Industry Applications*, Vol.33, No.6, November/December 1997, pp.1448-1458.

[12] Marek Samotyj. "Solutions to Voltage Sag Problems", *EPRI Journal*, Vol.20, No.4, July/August 1995, pp.42-45.

[13] Mark F. McGranaghan, David R. Mueller, Marek J. Samotyj. "Voltage Sags in Industrial Systems" *IEEE Transactions on Industry Applications*, Vol.29, No.2, March/April 1993, pp.397-402.

[14] Hamish Laird, Simon Round, Richard Duke, Alister Gardiner. "Power Quality Observations at a Light Industrial Site", *Proceedings of the IEEE Eighth International Conference on Harmonics and Quality of Power*, Athens, Greece, October 1998, pp.88-93.

[15] Van Wagner, Thomas Grebe, Robert Kretschmann, Lawrence Morgan, Al Price. "Power System Compatibility with Industrial Process Equipment", *IEEE Industry Applications Magazine*, Vol.2, No.1, January/February 1996.

[16] J.V. Milanovic, R. Gnativ, K.W.M. Chow. "The Influence of Loading Conditions and Network Topology on Voltage Sags", *Proceedings of the IEEE Ninth International Conference on Harmonics and Quality of Power*, Orlando, FL, October 2000, pp.757-762.

[17] P.R. Chaney. *The Characterization of Voltage Sags on an Electric Power System*, Clemson, SC, Clemson University M.S. Thesis, 1998.

[18] Math H.J. Bollen. "The Influence of Motor Reacceleration on Voltage Sags", *IEEE Transactions on Industry Applications*, Vol.31, No.4, July/August 1995, pp.667-674.

[19] Larry Conrad, Kevin Little, Cliff Grigg. "Predicting and Preventing Problems Associated with Remote Fault-Clearing Voltage Dips", *IEEE Transactions on Industry Applications*, Vol.27, No.1, January/February 1991, pp.167-172.

[20] *Waveform Characterization of Voltage Sags: Definition and Algorithm Development*, Palo Alto, CA, EPRI, 1999.

[21] "Power Quality Considerations for the Textile Industry", EPRI & Duke Power Company, 1995.

[22] Linda S. Tillis, Charles W. Williams, Jr.. "Voltage Sag Investigation and Mitigation in an Electronics Manufacturing Plant", *IEEE Southcon '94 Conference Record*, Orlando, FL, March 1994, pp.378-381.

[23] C.A. Warren, T.A. Short, J.J. Burke, H. Morosini, C.W. Burns, J. Storms. "Power Quality at Champion Paper - The Myth and the Reality", *IEEE Transactions on Power Delivery*, Vol.14, No.2, April 1999, pp.636-641.

[24] Jeff Lamoree, Dave Mueller, Paul Vinett, William Jones, Marek Samotyj. "Voltage Sag Analysis Case Studies" *IEEE Transactions on Industry Applications*, Vol.30, No.4, July/August 1994, pp.1083-1089.

[25] "Low Voltage Ride-Through Performance of 5-HP Adjustable-Speed Drives", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.9, January 1993.

[26] "Low Voltage Ride-Through Performance of AC Contactor Motor Starters", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.10, February 1993.

[27] E.R. Collins, M.A. Bridgwood. "The Impact of Power Disturbances on AC-Coil Contactors," *Proceedings of the 1997 IEEE Textile, Fiber, and Film Industry Technical Conference*, Greenville, SC, 1997, pp.2-7

[28] *Programmable Logic Controllers used in Industrial Power Systems, PLC I Final Report*, Knoxville, TN, EPRI Power Electronics Applications Center, August 1996.

[29] "Ride-Through Performance of Programmable Logic Controllers", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.39, November 1996.

[30] "IEEE Recommended Practice for Evaluating Electric Power System Compatibility with Electronic Process Equipment", *IEEE Standard 1346-1998*, New York, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[31] Larry E. Conrad, Math H.J. Bollen. "Voltage Sag Coordination for Reliable Plant Operation" *IEEE Transactions on Industry Applications*, Vol.33, No.6, November/December 1997, pp.1459-1464.

[32] ITI(CBEMA) Curve, Published by Information Technology Industry Council (ITI), 1250 Eye Street NW, Suite 200, Washington, DC 20005, http://www.itic.org.

[33] S. Middlekauff. *Using a Sag-Generator to Identify Power Quality Problems in Industry*, Clemson, SC, Clemson University M.S. Thesis, 1996.

[34] E.R. Collins, Jr., R.L. Morgan. "A Three-Phase Sag Generator for Testing Industrial Equipment", *IEEE Transactions on Power Delivery*, Vol.11, No.1, January 1996, pp.526-532.

[35] R. Caldon, M. Fauri, L. Fellin. "Voltage Sag Effects on Continuous Industrial Processes: Desensitizing Study for Textile Manufacture", *Proceedings of the Second International Conference on Power Quality, PQA, '92: End Use Applications and Perspectives*, Atlanta, GA, EPRI, 1992, pp.D-13:1-D-13:7.

[36] M.A. Bridgwood, J. Spangler, P. Miroftsalis, E.R. Collins. "Characterizing the Behavior of a Multi-Loop Model Industrial Process Subjected to Complex Power Supply Voltage Sags", *Proceedings of the Eleventh International Power Quality '98 Symposium*, Santa Clara, CA, November 1998, pp.414-425.

[37] M.A. Bridgwood, O.C. Parks, E.R. Collins. "Using Macromodels to Evaluate a Process Control System's Response to Voltage Sags", *Proceedings of the Eleventh International Conference on Harmonics and Quality of Power*, Lake Placid, NY, September 2004, pp.717-720.

[38] Carlos Alvarez, Javier Alamar, Alex Domijan Jr., Alejandro Montenego, Zhidong Song. "An Investigation Toward New Technologies and Issues in Power Quality", *Proceedings of the IEEE Ninth International Conference on Harmonics and Quality of Power*, Orlando, FL, October 2000, pp.444-455.

[39] Kay Tuttle. "How to Position Power Quality Within a Utility", *Proceedings of the IEEE Ninth International Conference on Harmonics and Quality of Power*, Orlando, FL, October 2000, pp.438-443.

[40] Thomas M. Gruzs. "Power Quality - A Shared Responsibility", *Proceedings of the First International Power Quality Conference - Power Quality '89 Symposium*, Long Beach, CA, October 1989, pp.32-41.

[41] Douglas S. Dorr, M. Brent Hughes, Thomas M. Gruzs, Robert E. Jurewicz, John L. McClaine. "Interpreting Recent Power Quality Surveys to Define the Electrical Environment" *IEEE Transactions on Industry Applications*, Vol.33, No.6, November/December 1997, pp.1480-1487.

[42] Melisa Johns, Larry Morgan. "Voltage Sag Mitigation Through Ridethrough Coordination", *Proceedings of 1994 IEEE/IAS Annual Textile, Fiber and Film Industry Technical Conference*, Greenville, SC, May 1994, pp.1-5.

[43] Robert C. Degeneff, Russ Barss, Danial Carnovale, Steven Raedy. "Reducing the Effect of Sags and Momentary Interruptions: A Total Owning Cost Prospective", *Proceedings of the IEEE Ninth International Conference on Harmonics and Quality of Power*, Orlando, FL, October 2000, pp.397-409.

[44] Math H.J. Bollen, Thavatchai Tayjasanant, Gulali Yalcinkaya. "Assessment of the Number of Voltage Sags Experienced by a Large Industrial Customer" *IEEE Transactions on Industry Applications*, Vol.33, No.6, November/December 1997, pp.1465-1471.

[45] Milind M. Bhanoo. "Static Transfer Switch: Advances in High Speed Solid-State Transfer Switches for Critical Power Quality and Reliability Applications", *Proceedings of the 1998 IEEE Annual Textile, Fiber, and Film Industry Technical Conference*, May 1998, pp.5/1-5/8.

[46] Roger C. Dugan, Mark F. McGranaghan, H. Wayne Beaty. *Electrical Power Systems Quality, Second Edition*, New York, McGraw-Hill, 1996.

[47] H.J. Kim. *A New Approach to Correct Power System Voltage Disturbances Using a Six Pulse Inverter*, Clemson, SC, Clemson University M.S. Thesis, 2000.

[48] P. Wang, N. Jenkins, M.H.J. Bollen. "Experimental Investigation of Voltage Sag Mitigation By an Advanced Static Var Compensator" *IEEE Transactions on Power Delivery*, Vol.13, No.4, October 1998, pp.1461-1467.

[49] "Ferro-Resonant Transformer Output Performance Under Varying Supply Conditions", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.13, May 1993.

[50] "Ride-Through Performance of a Web Process Enhanced by a Constant-Voltage Transformer", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.40, November 1996.

[51] S.N. Tunaboylu. *Electromechanical Performance of Induction Motor/Drive Systems Subjected to Voltage Sags*, Clemson, SC, Clemson University Ph.D. Dissertation, 2002.

[52] "Low Voltage Ride-Through Performance of a Modified Personal Computer Power Supply", *Power Quality Testing Network Briefs*, Knoxville, TN, The EPRI Power Electronics Applications Center, No.12, April 1993.

[53] M.A. Bridgwood, P. Miroftsalis, E.R. Collins, J. Spangler. "Sag Injection/Detection Systems for PLC-Based Industrial Process Control Applications", *Proceedings of the Eleventh International Power Quality '98 Symposium*, Santa Clara, CA, November 1998, pp.511-519.

[54] N.T. Youell. *Development of a Real-Time, Three-Phase, Cost-Effective Voltage Disturbance Detector*, Clemson, SC, Clemson University M.S. Thesis, 2001.

[55] Oscar C. Montero-Hernandez, Prasad N. Enjeti. 'A Fast Detection Algorithm Suitable for Mitigation of Numerous Power Quality Disturbances', *Conference Record of the 36th IEEE Industry Applications Society (IAS) Annual Meeting*, Vol.4, October 2001, pp.2661-2666.

[56] J. Ackermann. 'Robustness Against Sensor Failures', *Automatica*, Vol.20, No.2, March 1984, pp.211-215.

[57] Koicha Suyama. 'Fault Detection of Redundant Sensors used in Reliable Sampled-Data Control Systems' *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, December 1998, Vol.1, pp.1161-1164.

[58] G. Hearns, M.J. Grimble. 'Fault Tolerant Strip Tension Control', *Proceedings of the American Control Conference*, Philadelphia, PA, June 1998, pp.2992-2996.

[59] W. Wolfermann. 'Sensorless Tension Control of Webs', *Proceedings of the Fourth International Conference on Web Handling*, June 1997, Web Handling Research Center, Oklahoma State University, Stillwater, OK, pp.318-340.

[60] Seung-Ho Song, Seung-Ki Sul. 'A New Tension Controller for Continuous Strip Processing Line', *IEEE Transactions on Industry Applications*, Vol.36, No.2, March/April 2000, pp.633-639.

[61] Norbert A. Ebler, Ragnar Arnason, Gerd Michaelis, Noel D'Sa. 'Tension Control: Dancer Rolls or Load Cells', *IEEE Transactions on Industry Applications*, Vol.29, No.4, July/August 1993, pp.727-739.

[62] J. Timothy Shea. 'U.S. Device/Sensor Bus Race: Three Contestants Outpace the Field', *Instrumentation and Control Systems*, Vol.71, No.6, June 1998, pp.61-65.

[63] B.D. Van Blerk, G. Diana. 'Development of a Generic Tension Control System', *AFRICON 1996, IEEE 4th AFRICON*, September 1996, Vol.2, pp.864-868.

[64] Priyadarshee D. Mathur, William C. Messner. 'Controller Development for a Prototype High-Speed Low-Tension Tape Transport', *IEEE Transactions on Control Systems Technology*, Vol.6, No.4, July 1998, pp.534-542.

[65] Bimal K. Bose. *Modern Power Electronics and AC Drives*, Upper Saddle River, NJ, Prentice-Hall, 2001.

[66] Alex McEachern. "Designing Electronic Devices to Survive Power Quality Events", *IEEE Industry Applications Magazine*, Vol. 6, No. 6, November/December 2000, p.66-9.

[67] *Duke Energy Process Sensors Test Study*, Prepared by EPRI PEAC Corporation, 2000.

[68] J.T. Spangler. *Forming the Mathematical Model of a Multiple Loop Feedback Controls System for Improving the Ride-Through Capabilities of a Complex Textile Process*, Clemson, SC, Clemson University M.S. Thesis, 1999.

[69] O.C. Parks. *The Development of a Serial Data Bus Driven Textile Process for Use in Power Quality Studies*, Clemson, SC, Clemson University M.S. Thesis, 2000.

[70] Ferdinand P. Beer, E. Russell Johnston, Jr.. *Vector Mechanics for Engineers: Dynamics*, New York, McGraw-Hill, 1984.

[71] William L. Brogan. *Modern Control Theory, Third Edition*, Upper Saddle River, NJ, Prentice-Hall, 1991.

[72] "Test Designation ASTM D5035-95", *Annual Book of ASTM Standards, Vol 7.01*, West Conoshocken, PA, American Society for Testing and Materials, 1999.

[73] Stephen J. Chapman. *Electric Machinery Fundamentals, Second Edition*, New York, McGraw-Hill, 1991.

[74] David G. Hart, William Peterson, David Uy, Jochen Schneider, Damir Novosel, Ray Wright. 'Tapping Protective Relays for Power Quality Information' *IEEE Computer Applications in Power*, Vol.13, Issue 1, January 2000, pp.45-9

[75] Mesut E. Baran, Jinsang Kim, David G. Hart, David Lubkeman, Glenn C. Lampley, William F. Newell. 'Voltage Variation Analysis for Site-Level PQ Assessment', *IEEE Transactions on Power Delivery*, Vol.19, No.4, October 2004, pp.1956-1961.

[76] Roy Moxley. 'Synchrophasors in the Real World', Pullman, WA, Schweitzer Engineering Laboratories Inc., http://www.selinc.com, 2005.

[77] Mark Adamiak, William Premerlani, Bogdan Kasztenny. 'Synchrophasors: Definition, Measurement, and Application', GE Corporation (GE Multilin/GE Corporate Research), http://www.gedigitalenergy.com, 2009.