National Instruments

Machine Design Guide

NATIONAL INSTRUMENTS™

Contents

Executive Summary 4

Chapter 1: Conceptual Machine Design and Mechanical Design 10

Chapter 2: Electrical Design 22

Chapter 3: Embedded Software Design 32

Chapter 4: Control Design 42

Revision 2.0, February 2011

Authors

Christian Fritz, NI Senior Product Manager, Advanced Machine Control and Robotics

Brian MacCleery, NI Senior Product Manager, Industrial and Embedded

Javier Gutierrez, NI Product Manager, Control

Todd Walter, NI Senior Product Manager, PAC and Machine Design

Executive Summary

Executive Summary

Introduction

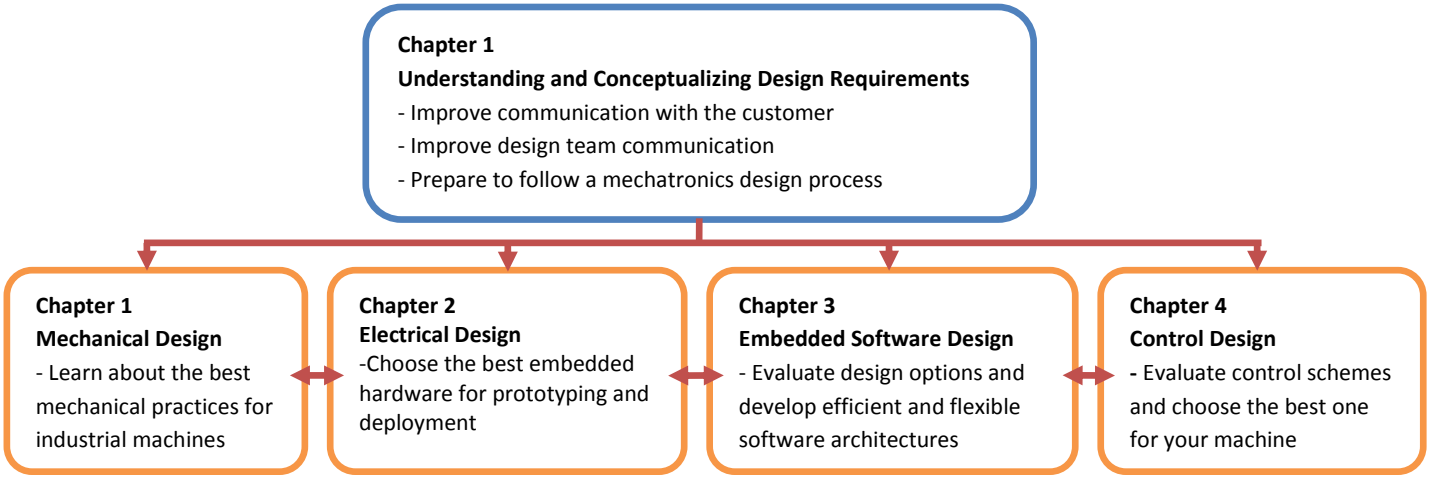
Faster, better, and cheaper is the name of the game for machine and device designers. Demands for higher performance and increasingly efficient machines designed in a shorter amount of time with smaller design teams are challenging machine designers to improve their design processes. The Aberdeen Group identified these top machine design challenges after interviewing engineering managers and designers from 160 machine design companies in January 2008 (see Table 1). The study also discovered that to meet these challenges, successful companies follow the mechatronics design approach – they increase integration between mechanical, electrical, control, and embedded programming design processes. This machine design guide examines the best practices and tools that successful machine designers use to make different design trade-offs, meet design challenges, and increase profit.

Pressures	Response
Shorter product development schedules	69%
Increased customer demand for better performing products	44%
Reduced development budgets	25%
Accelerated product customization	20%
Increased requirements to incorporate electronics and software into product	16%

Source: Aberdeen Group, January 2008

Table 1. The Aberdeen Group's research of 160 machine design companies identified the top challenges machine designers face.

With this guide, learn step-by-step best practices for the machine design process, starting from understanding customer requirements and conceptualizing design ideas in chapter 1 to following the mechatronics-integrated design approach, lowering design risk, and increasing machine productivity in chapters 2 through 4. Throughout the guide, in-depth application examples illustrate the approaches successful designers implement. Finally, use in-depth tutorials referenced in the chapters to take the next step and incorporate the recommended best practices. Following is an overview of the machine design process covered in this design guide:



Machine Design Challenges

Intense competition is putting pressure on machine and device builders to deliver systems with higher throughput, reduced operating cost, and increased safety. Machine builders have switched from rigid, single-purpose machines relying purely on mechanical gears and cams to flexible multipurpose machines by adopting modern control systems and servomotors. Although these improvements have made machines flexible, they have also introduced a significant amount of complexity to the machines and subsequently to the machine design process.

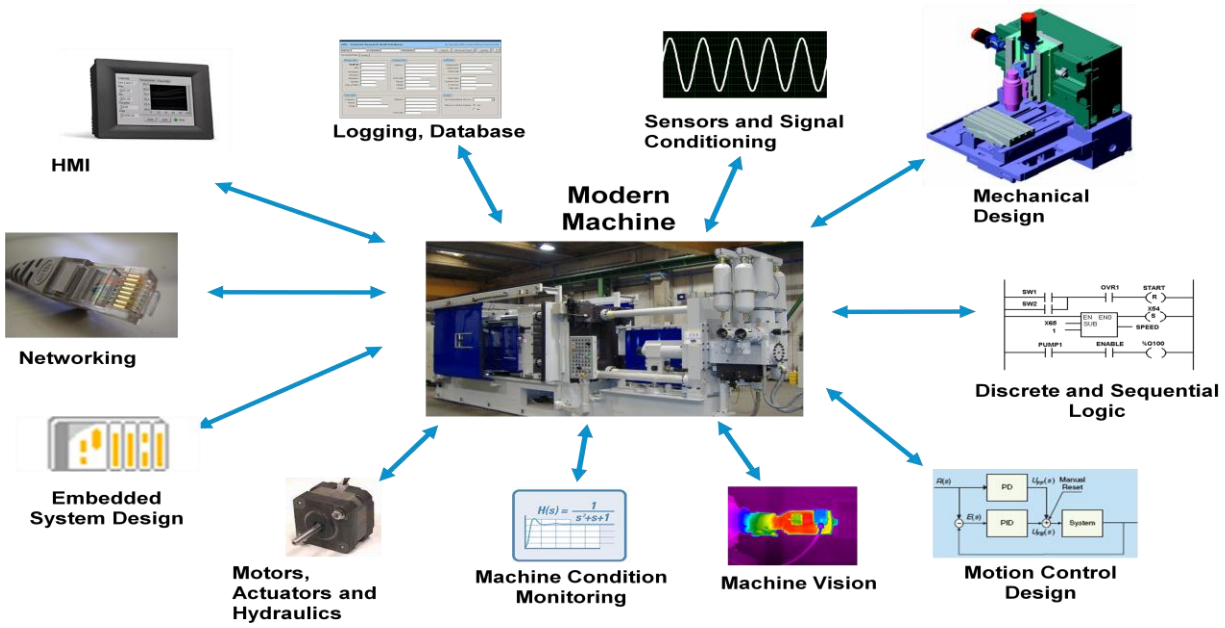


Figure 1. To maintain a leadership position, machine builders are designing increasingly complex machines.

Case Study 1: SLI Lighting Bulb Manufacturing Machine (The Challenge)

SLI Lighting is one of the world's leading innovators and manufacturers of artificial lighting sources. SLI designed an improved metal halide bulb that, unlike traditional bulbs, does not use the standard glass-on-electrode design. Instead, it uses electrodes embedded into a single piece of glass. Traditional machines designed to press melted glass around electrode-foil wires were not capable of producing these new lamps. Therefore, SLI needed to develop a more advanced machine that incorporated multiple axes of precise motion control to position the glass and electrodes, synchronize that position with vision inspection that locates the center point for the bulb, and control a laser to melt the glass with pulse-width modulation (PWM). Additionally, the machine had to be flexible to produce a wide range of other lighting products and allow easy and quick setup to change from one type of light bulb production to another. SLI machine designers need to develop new, flexible, high-throughput machines like this in a limited development time.

Case Study 2: Designing a Medical Device for Automated Cell Secretion Analysis (Challenge)

For more than 15 years, Biorep Technologies Inc. has designed and manufactured original equipment for the medical industry. By providing innovative machines and devices for medical diagnostics and surgery, Biorep delivers unique and practical solutions for hospitals and doctors around the world.
To improve the throughput and repeatability in cell secretion analysis, which is routinely conducted with pancreatic islets in type 1 diabetes research, Biorep developed a fully automated system incorporating human machine interface (HMI), motor control, industrial communication, and analysis functionality. To improve the flexibility and maintainability of the company's system, engineers need design tools that help them reuse algorithms throughout the design process and develop all components of the control system with the same programming tools.

Mechatronics Offers Solutions

More machine builders today are in positions similar to those of SLI and Biorep. Meeting these multidisciplinary engineering challenges requires improvements in all machine design areas. Mechatronics, gaining in popularity as a way to describe this evolution, represents an industry-wide effort to improve the design process by integrating the best development practices and technologies to streamline machine design, prototyping, and deployment. A mechatronics-based approach can lower the risks associated with machine design early in the design cycle and help machine designers meet the key challenges they face today.

In the past, following the mechatronics design approach has been difficult. But developments in the market, such as the integration between best-in-class software packages used for different design phases, the evolution of flexible and modular programmable automation controllers (PACs), intuitive graphical system design software, and advanced control design techniques, have helped machine designers successfully implement the mechatronics design approach. The following chapters in this machine design guide explore each of these advancements.

Chapter 1: Conceptual Machine Design and Mechanical Design

Understanding customer requirements and incorporating them into a machine design are two of the most critical parts of the machine design process. It is important at this stage to evaluate different machine mechanical assemblies and calculate the price-versus-performance trade-offs. Traditionally, machine designers have used past experiences or back-of-the-envelope calculations for the planning process. This chapter explores evolving simulation tools that help machine designers quickly simulate different mechanical assemblies in software and determine the key performance metrics for each assembly including the maximum machine throughput. This allows machine designers to significantly reduce machine design risk early in the design process. Read this chapter for an in-depth look at tools that can help you plan better for your next machine.

Chapter 2: Electrical Design

With the increasing complexity of machines, choosing a software and hardware platform that integrates control logic, I/O, analog measurements, motion, vision, human machine interfaces (HMIs), and machine condition monitoring can reduce machine design time and increase machine productivity. This chapter covers industrial control hardware advancements that combine the ruggedness of industrial-grade

systems and the flexibility and performance of PC and embedded technology. The chapter also discusses the increased I/O requirements of modern machines and devices and explores ways to integrate advanced measurement capabilities into different machine control hardware platforms.

Chapter 3: Embedded Software Design

Whether you are bidding on a machine design contract or planning a machine for use within your organization, it is important to evaluate options that can help you design a more productive and robust machine. This chapter looks at the different features – from machine condition monitoring for detecting factoring that reduces machine productivity to vision inspection for defect detection in manufactured parts – you should consider when designing machines. The chapter also explores the best practices for designing scalable software architecture for your machine.

Chapter 4: Control Design

Determining the optimal control system for a machine has a big impact on the productivity of the machine. This chapter describes best practices for identifying the control algorithm that is ideally suited for your machine.

Case Study 1: SLI Lighting Bulb Manufacturing Machine (Solution)

To meet the challenge of designing a flexible bulb manufacturing machine, SLI Lighting followed the mechatronics machine design process and incorporated a control system based on programmable automation controllers and the NI LabVIEW graphical programming environment. Because of this, the company had a single software and hardware platform for integrating precision motion control of servo and stepper motors, vision inspection of the glass, and PWM control of the CO2 laser. The resulting machine is high in productivity and scalable for manufacturing different bulb types. “With the LabVIEW platform, we can easily adapt to different process requirements and adapt the software for future enhancements.” – Danny Hendrikx, Production Machine Designer, SLI R&D

Figure 2. By using NI programmable automation controllers and the LabVIEW graphical programming environment, SLI Lighting developed a flexible bulb manufacturing machine.

Case Study 2: Designing a Medical Device for Automated Cell Secretion Analysis (Solution)

After considering the evolution of National Instruments products, Biorep decided to harness the power of LabVIEW graphical system design software to address the challenge of a unified development tool for all of the company's control needs.

Using National Instruments tools, Biorep Technologies Inc. and the Diabetes Research Institute at the University of Miami developed an automatic perifusion system that simplifies the process and allows for the stimulation of different cell types with total environmental control and the collection of its secretions according to programmable protocols. The system progressed through the design, prototype, and deployment stages in only three months.

Figure 3. By using mechatronics design techniques, NI programmable automation controllers, and LabVIEW, Biorep developed an innovative perifusion system in three months.

Conclusion

With the increasing demand for higher-productivity machines and devices, shorter design times, and lower machine costs, machine designers need to reconsider their design approaches. This machine design guide examines design techniques that successful machine builders are using to improve the productivity and lower the cost and risk of their machines.

Chapter 1

Conceptual Machine Design

Chapter 1: Conceptual Machine Design and Mechanical Design

Transitioning From the Idea to Software Design

New ideas are exciting. But where do engineers go once they think of an innovative idea? What is the first step on the way to a successful machine or device implementation?

With an innovative idea for a new device, engineers may be tempted to just start on a physical prototype right away. Resisting this temptation saves time and wasted effort in the long run. The time invested in paper design pays big dividends later and helps engineers avoid many common pitfalls in the design process. Paper design does not mean writing out detailed designs for prototype son paper with a pen or pencil. Paper design is creating a plan before implementing any software coding or hardware design. Some of the benefits of paper design are discussed below.

Getting Ideas Out of Your Head and on Paper

Solidify the details surrounding that great idea you have been kicking around. The more detail you can put on paper, the easier it is to translate to an actual functional prototype.

Fail Now, Not Later

Explore all the possible permutations of your idea. This is the time to let your imagination run wild because the biggest expense at this stage is time. Find the best implementation now while there is no sunk cost associated with a failed design. Fail early, fail often, and fail forward in the paper design phase to ensure your design is up to snuff.

Obtain Peer and Customer Feedback

Build consensus with partners and peers and obtain buy-in from early customers and investors. A good paper design helps others fully grasp your idea. The more effective you are at explaining your design ideas, the higher the quality of your peer feedback and the more likely you are to get funding and support. Even this early, you should be iterating on your design until it is defensible and stands up to critique.

Leverage CAD Software for Visualization

Use available computing resources to make a professional, detailed physical design. A design defined in CAD software with dimensions, notes, and materials specified is much more powerful than a back-of-the-napkin sketch. It implies a detailed, well-thought-out design and a serious intent. This helps you gain consensus and get buy-in because others can more effectively visualize your design. This is strictly for visualization purposes. Resist the temptation to move forward with the detailed design of your functional prototype at this stage. A more detailed design comes later.

Use a Graphical Representation of Your System

Use standard techniques like a flowchart or a state diagram to define the functionality of your device. This flowchart or state diagram forms the basis you need to make software architecture decisions and defines the framework of your algorithm design.

Define Your Requirements

How do you go from a great idea and a back-of-the-napkin sketch to a detailed paper design? The first step is to clearly define your goals by making a list of user requirements. These requirements should be as specific as possible. Research is crucial at this early stage to be sure you can meet your outlined requirements. Is your design feasible? Will it realistically be able to meet your requirements? Make sure that you distinguish between needs and wants for your design. As an innovator, you may be tempted to add advanced but not completely necessary features to your prototype. Know your objectives and stick to them.

Abstract Components from Requirements

With abstraction, you can describe an application without defining how to write the application. Abstraction generalizes the application to a high conceptual level. There are generally two types of abstraction: procedural and data. Procedural abstraction separates what a procedure accomplishes from how the procedure is implemented. An example of a procedural abstraction is opening a file. You can open a file in many different ways, but you do not need to be concerned with that when abstracting.

Data abstraction separates the data you want to store from the physical means of storing the data. An example of data abstraction is an array of data. An array can represent data with a more logical view without requiring the user to be concerned with the details of its implementation. To assist with abstraction, remove key verbs and nouns from your system requirements document. From these verbs and nouns, you can determine what your program needs to accomplish and the objects that will be a part of your user interface. The verbs and nouns also help you determine the hardware components necessary to complete your prototype.

Flowcharts

Once you have gleaned a set of abstracted components from your device requirements, you can use a flowchart to move from abstracted components to a software program. Flowcharts help you develop a good understanding of the application flow by dividing the application into more manageable pieces. NI LabVIEW is a graphical programming environment designed for measurement and automation applications, which makes it an ideal tool for quickly converting your paper design to code. Because a LabVIEW block diagram is similar to a flowchart, moving from the flowchart to software code is a quick process.

State Diagrams

A state diagram is a specific type of flowchart that indicates the states of a program and the transitions between states. Each state satisfies a condition, performs an action, or waits for an event. The transitions between the states are conditions, actions, or events that cause the program to move to the next state. State diagrams are useful for prototyping because almost all embedded systems use a state architecture. That is, they are designed with the understanding that the prototype is always in a given state, even if that state is idle.

To begin creating a state diagram, you need to have a full understanding of your application and anything within your system that is part of your user interface. This can be a value that is displayed or a user input such as a numeric or Boolean control. A well-designed flowchart assists in this process. Next, write down each state and how your prototype transitions from one state to another. Most software

development environments provide tools to quickly transfer a state chart into executable software. One of the later chapters discusses the advantages of graphical programming and how you can use state charts in LabVIEW.

After transitioning your idea into a paper design and getting buy-in from key stakeholders and customers, pull the design teams together and start prototyping.

Mechatronics-Oriented Techniques to Improve Communication and Reduce Risk

Introduction

To stay ahead of the competition, leading machine design teams are incorporating sophisticated features such as synchronized motion, vision, I/O, real-time signal processing and analysis, remote monitoring and support, data logging, and more. At the same time, industrial automation equipment purchasers are placing more emphasis on the importance of optimizing the design not just for increased performance but also for environmental factors such as reduced energy consumption and scrap material. To meet these demands, machine designers are replacing the gear- and cam-based motion control systems that characterized the traditional machine design process with newer software-controlled electric motor drive systems. This transition from a mechanical design process to an electromechanical design process is having a profound impact on how machine teams work together. The modern machine is a collection of interconnected subsystems with many design dependencies that cross the boundaries between the mechanical, electrical, software, and embedded system domains.

Despite increasing complexity, most design teams report more pressure than ever to shorten development times, reduce risk, and lower costs. This combination of forces is causing many teams to reevaluate their development practices and consider a more mechatronics-oriented approach to make the development process more parallel. Mechatronics-oriented design tools improve machine and device development by simulating the interaction between mechanical and electrical subsystems throughout the design process. Historically, teams of engineers from different disciplines worked in silos and in sequential development. Design decisions were made independently, resulting in longer development times and higher costs. Now, to streamline development in a mechatronics approach, the teams work in parallel and collaborate on design, prototyping, and deployment. According to a recent study of mechatronics design teams conducted by the Aberdeen Group and sponsored by NI, best-in-class companies take steps to improve their communication and collaboration across engineering disciplines, define and track the status of design requirements, and identify system-level problems earlier in the development. For example, best-in-class companies are 5.3 times more likely than the industry average to use simulation and test tools.

The following high-level summary of recommended development practices from the Aberdeen mechatronics study reflects the practices of the best-performing companies among the more than 140 companies interviewed for the study. [1]

- Include all engineering disciplines in formal design reviews throughout the development
- Formally document cross-disciplinary design issues and notify engineers immediately if changes to a subsystem may affect their design

- Set design performance metrics for all engineering disciplines and formally gather, manage, and track design requirements throughout development
- Use simulation and test techniques to evaluate system performance, detect flaws, and determine trade-offs

In this chapter, explore the several ways you can use graphical system design technologies from National Instruments to improve communication among design team members and with customers. Learn how new mechatronics-oriented tools such as LabVIEW and SolidWorks can help you bring team members together in the early development stages to refine the design requirements, evaluate the pros and cons of different conceptual design options, and reduce risk by performing design analysis and optimization across electromechanical boundaries.

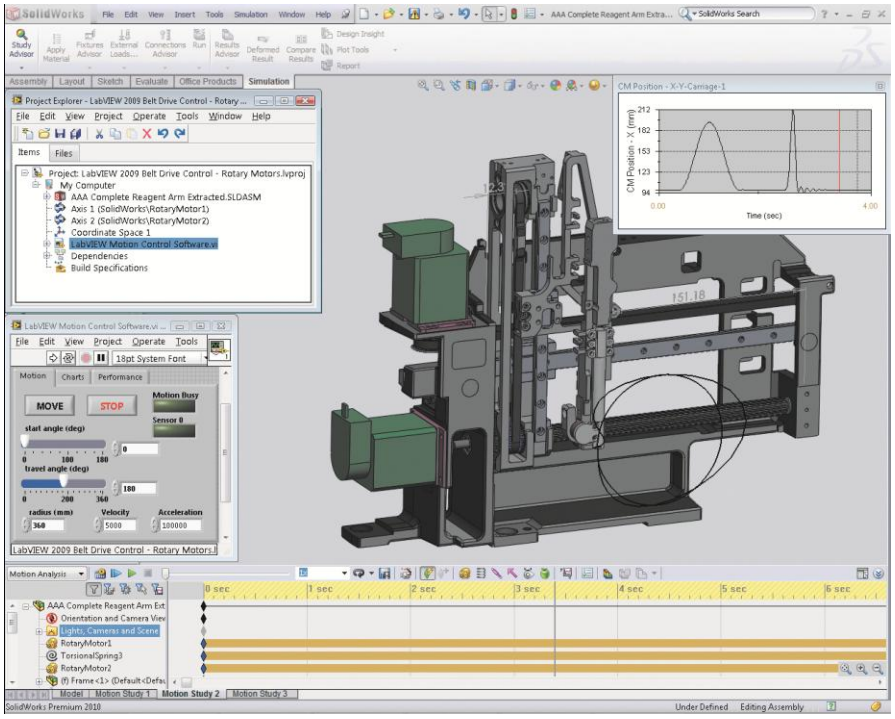


Figure 1. The integration of mechanical design tools and control design tools enable engineers and scientists to experience the dynamic behavior of their machine before building a first prototype.

Identifying Key Design Decisions and Dependencies

To ensure the final design meets your customer requirements, it is critical to work with the customer early in the development and track the requirements throughout the development cycle. According to the Aberdeen Group, successful mechatronics design teams conduct frequent design reviews with the entire cross-functional team and actively try to identify design decisions that affect more than one group. Figure 2 shows some of the key design decisions made by each engineering discipline in a typical mechatronics machine design process and illustrates some of the key dependencies between groups.

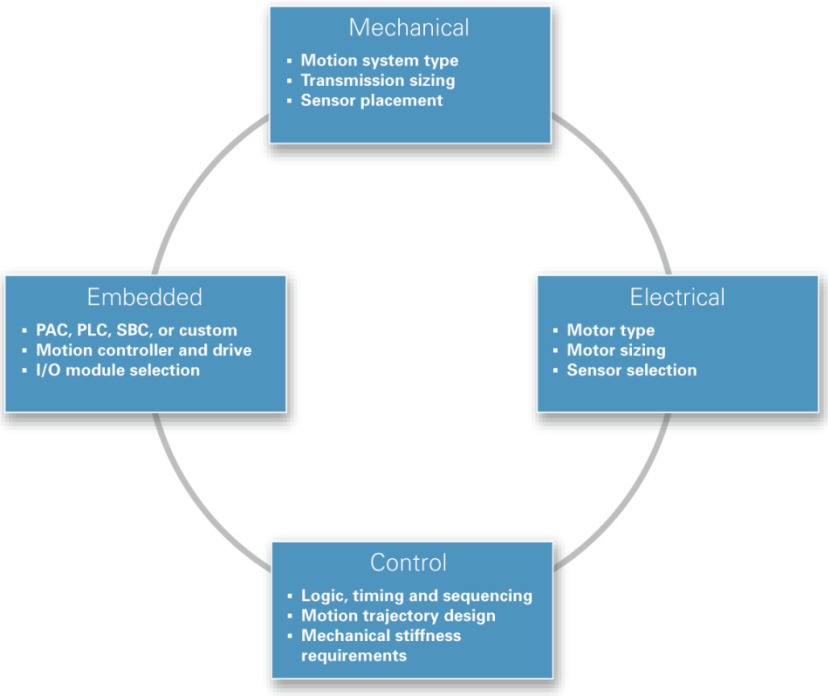


Figure 2. Machine design requires effective communication between engineering disciplines.

One of the first decisions your team needs to make is which type of motion control conceptual design to use because this decision affects nearly every other aspect of the machine. As shown in Figure 3, common motion system types include linear Cartesian gantries (such as a three-axis pick-and-place configuration of linear actuators), rotary indexing tables, SCARA robots, and conveyers.

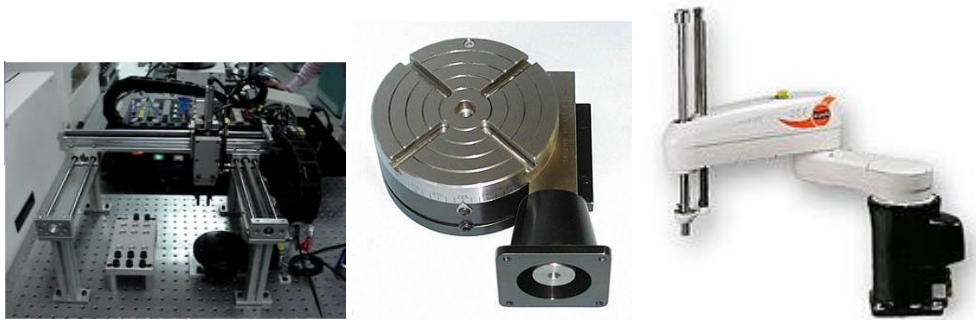


Figure 3. Typical motion system types, from left to right, include Cartesian gantry, rotary table, and SCARA robot.

The proper selection of mechanical transmission components and motors is critical to achieving precise, high-throughput motion. Motor sizing depends primarily on the torque and velocity requirements needed to execute the motion trajectories and the inertia of the mechanical load with respect to the motor shaft. Motion trajectory design is critical because acceleration information is embedded in the motion profiles, and the required torque depends on the acceleration requirements for the mechanical configuration with mass, friction, moment arms, and so on. In other words, it's not enough to simply specify that the part moves from point "A" to point "B" in "x" amount of time. The details of the motion

trajectory make a difference because they determine how the part accelerates and decelerates along the path. Motion trajectories are typically specified by the target position, move type (straight line, arc, or contour), acceleration profile (trapezoidal or S-curve), and move constraints (velocity, acceleration, and jerk limits). If the mechanical engineer decides to change the screw pitch of a lead-screw actuator, the change may require the motor to be resized because it changes the load inertia reflected to the motor shaft and the torque versus velocity requirements for the motor. Likewise, if the control engineer changes the control logic to improve the cycle time of the machine by reducing dead time, the resulting increase in the operating duty cycle for the motors may require resizing. In other words, virtually every decision related to the design of the motion control system involves cross-disciplinary issues, so good communication and collaboration are essential. Because design changes by any group may have a ripple effect, conducting a more parallel development process augmented by new simulation technologies can yield a more optimized design and help identify flaws earlier in the development process. Throughout development, the entire team should be notified if any changes are made to the motion control system. In the traditional machine design model, the design was done sequentially and passed from one group to the next – typically mechanical first. Oftentimes, the control team didn't become heavily involved until the first physical prototype was built and became ready for programming. By that time, it's very expensive to change the design, so the sequential approach represents a very risky development process.

In the following sections, explore two powerful simulation techniques you can use early in the machine design cycle to help improve the definition of your design requirements, compare conceptual design options, examine engineering trade-offs, evaluate the dependencies between your development groups, and refine your design requirements.

Technique 1: Use Motion Simulation Tools to Create a Virtual Prototype of Your Machine

To help bridge the gap between the engineering domains involved in the machine design process, tool vendors such as National Instruments and SolidWorks are reaching across the aisles to provide a more integrated mechatronics development experience. Modern 3D CAD tools now incorporate motion simulation capabilities that designers can use to examine the force and torque properties of their machinery based on the assembly mate, mass, and friction properties contained in the 3D model. By integrating these simulation capabilities as a native part of the mechanical CAD environment, developers no longer need to have the specialized mathematical modeling expertise traditionally required to simulate mechanical assemblies. However, for team members to create a realistic virtual prototype simulation of the machine operation, they must connect the simulation environment to industrial-grade motion control software. This helps designers simulate machine operation using the same type of control software environment they will use to program the actual motion control system when they build the first physical prototype of the machine.

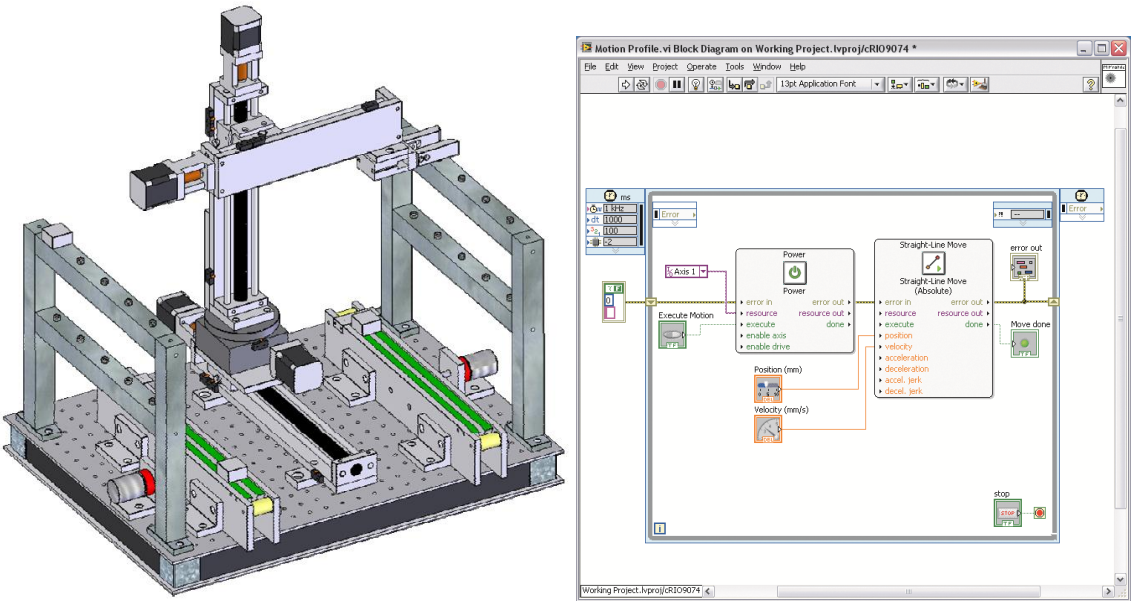


Figure 4. Taiwanese developers used virtual machine prototyping simulations in the design of this automatic storage machine.

This ability to perform virtual prototype simulations of the machine operation early in the development cycle has numerous benefits. First and foremost, the team can use these new tools to visualize and simulate the machine operation and control logic from within the standard design tool environment. This can promote better communication among everyone involved in the design process, including the developers, engineering managers, and customers, as well as help the team refine the design requirements to meet customer needs. It can also help to identify the most risky or least well-defined aspects of the solution that may warrant more detailed investigation. Furthermore, bringing all of the engineers together in the very early stages of the design to discuss, debate, and refine the implementation can help the team identify the key dependencies between the engineering groups. Finally, the team can use these tools to iterate on the design concept early in the design process while the design is still in the CAD modeling stage of development, which provides a much faster and lower-cost way to optimize the design compared to building and testing physical prototypes. For example, team members can evaluate the pros/cons and engineering trade-offs of various motion implementations such as whether they prefer a robotic motion control system to a Cartesian gantry or a precision lead-screw assembly to a lower-cost chain drive mechanism. Virtual machine prototyping tools also have specific benefits for the individual engineering disciplines involved in the design. For mechanical engineers, the simulation results data can provide force and torque loading information for stress and strain analysis. The simulation data is based on the actual motion profile trajectories, with their trapezoidal or S-curve acceleration profiles, and constraints such as velocity, acceleration, and jerk limits, resulting in more accurate force and torque load information. This helps the mechanical engineers design the machine so that the mechanical assembly is stiff enough to handle the dynamic loading caused by machine operation. For electrical engineers, virtual prototype simulations can help validate that a complementary set of component selection choices has been made for mechanical transmission components, motors, drives, and sensing devices such as limit switches, proximity probes, and position encoders. Control engineers not only can design the high-level statechart logic for the supervisory-level machine operation but also perform detailed design that includes the digital control logic, timing/sequencing, and motion profiles. They can perform motion trajectory design using high-

level function blocks that represent straight line, arc, contour moves, or even advanced techniques such as electronic gearing and camming.

The seamless integration of the LabVIEW NI SoftMotion Module and DS SolidWorks software delivers a design environment that is ideal for virtual prototyping. You can easily connect existing SolidWorks CAD models to LabVIEW, which automatically links the motor actuators and position sensors defined in the model. Using the high-level motion functions provided by LabVIEW NI SoftMotion, you can develop sophisticated motion control applications that include logic based on sensor feedback. Design teams, customers, and sales engineers then can use the virtual prototype to visualize realistic machine operations and analyze cycle time performance. Note that all of the software developed for simulation purposes can be used for design documentation. Furthermore, if you choose to implement your control system on an NI PAC platform such as NI CompactRIO, you can seamlessly transfer all of your simulation code to the embedded system for production use and deployment.

Technique 2: "Mock Up" Your Machine Operation Logic and HMI

This section examines a simulation technique that can help your team define the proper architecture for machine operation and engage your customer in refining the design requirements early in the development. By designing a statechart diagram that represents the high-level operation of the machine and connecting it to a simulated human machine interface (HMI) for the machine, your development team and customer can interact with a simulation of the high-level architectural design for the machine and provide feedback before you begin detailed software development. You can use statecharts to help simulate the overall operation of your machine and create an HMI mock-up that you can share with your customer.

From the customer viewpoint, the HMI for your machine can provide a great deal of information on how it will perform its task and how operators will interact with the control system during operation. From an engineering viewpoint, statechart diagrams, as shown in Figure ***5***, are an intuitive and succinct way to represent the high-level operation of the machine, from process operations to startup, shutdown, and fault-handling routines. Statechart diagrams can help your developer define a system-level specification for the design and formally partition the machine operation into well-defined subsystems. For example, how does the machine work with other machinery and equipment involved in the automation system, and how do materials flow in and out of the machine? Statecharts are sometimes referred to as "executable specifications" because the diagram format represents an intuitive, hierarchical way to specify the software architecture for the machine in a desktop environment. The same code can be targeted to the real-time processor or field-programmable gate array (FPGA) of a programmable automation control system for deployment. Like other LabVIEW graphical programming paradigms, statecharts are self-documenting because the code is written in an intuitive, high-level, readable format.

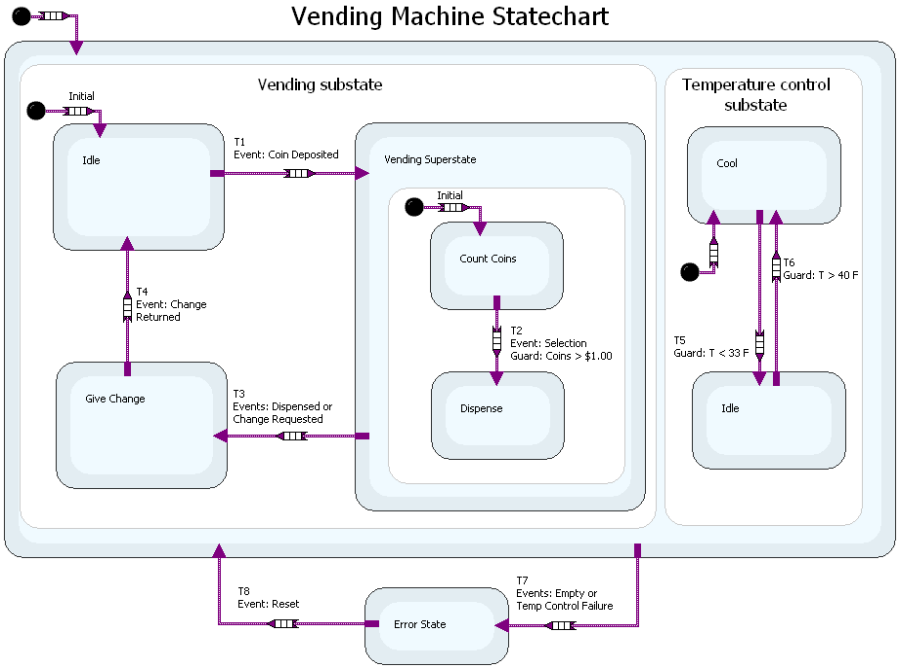


Figure 5. *Statechart diagrams, like this one for a vending machine application, help segment machine operations in well-defined subsystems.*

Statechart diagrams can facilitate effective design architectures and software development practices by segmenting the machine operation into well-defined subsystems and formally specifying how the machine transitions from one operating mode to another. Once you define the high-level software architecture, you can individually develop and test the code for the subsystems. This enables a "divide and conquer" approach to development, which simplifies the development process and fosters modular, portable, and reusable code. Figure ***5***, for example, shows the statechart diagram for a simple vending machine application. The vending substate manages the counting of money and the dispensing of product and change while the temperature control system regulates temperature. Based on sensor feedback, operator input, or the passage of time, the machine control system transitions from one state to another and updates the outputs. The statechart also provides an intuitive representation for multiple operations occurring concurrently and in parallel, such as the temperature control subsystem. Statecharts provide a framework for organizing software development. Because statecharts are executable, you can use them for simulation and reuse the code when deploying to the actual embedded control system, such as NI CompactRIO. The PackML statechart, defined by the OMAC organization, provides a standard statechart framework for machine control operation that includes startup, production, shutdown, and error-handling states, as described in Chapter 4. A statechart framework based on the PackML Version 3.0 model provides an excellent starting point for development and is available for download from NI Developer Zone.[2]

Figure ***6*** shows the operator interface simulation for a machine design based on the PackML statechart architecture. In addition to using the LabVIEW Statechart Module for the operational logic, this user interface also features industrial automation controls and indicators (such as the pipe and valve objects) available with the LabVIEW Datalogging and Supervisory Control (DSC) Module. The LabVIEW DSC Module includes an advanced HMI object and tools to simplify enterprise-wide machine monitoring and

tag-based communication with programmable logic controllers (PLCs) via industry-standard communication protocols. With LabVIEW DSC, you can automatically log machine operational data and statistics to a historical database, perform alarming, track production statistics such as throughput and uptime/downtime, and process yield performance data.

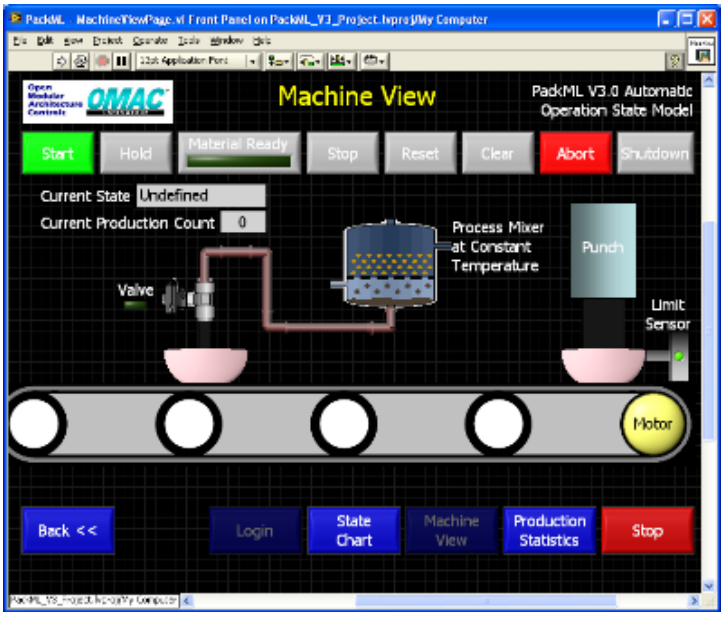


Figure 6. Machine control operator interface simulations offer an intuitive way for customers to provide feedback on the proposed implementation.

Conclusion

This chapter reviewed some tools and techniques you can use to facilitate the early conceptual design process for your machine using a mechatronics-oriented approach and philosophy. You learned about the concept of graphical system design using LabVIEW graphical programming tools and NI PAC machine control platforms to abstract low-level complexity and provide a streamlined path from early design to prototyping and deployment. You examined mechatronics-oriented development techniques practiced by best-in-class machine design companies and explored the key design decisions and cross-disciplinary dependencies involved in the design of electronically controlled motion systems. Then you learned about two development techniques that teams can use in the early conceptual design phase of development. The first technique, called virtual machine prototyping, involves using new motion simulation tools that are integrated into the 3D CAD environment and are driven by industrial-grade motion control software. The second technique is to use statechart diagrams to simulate the high-level operation of the machine, including the operator interface HMI for the machine. In both cases, you can reuse all simulation software during the detailed design phase and for documentation purposes. If you select NI PAC hardware as the embedded control and monitoring system for the machine, you can deploy the same motion control software and statechart logic to the embedded system for use in the production machine. Overall, these state-of-the-art tools and techniques can foster improved communication within your design team and with customers to help you reduce development time, cost, and risk.

[1] Aberdeen Group Research Benchmark Report, “System Design: New Product Development for Mechatronics,” January 2008, http://www.aberdeen.com/link/sponsor.asp?spid=30411059&cid=4576.
[2] National Instruments Developer Zone, “A State Chart Design Pattern for PackML Machine Control,” http://zone.ni.com/devzone/cda/epd/p/id/5391.

Chapter 2

Electrical Design

Chapter 2: Electrical Design

Choosing the Right Controller

Introduction

Engineers and machine builders are under increasing pressure to differentiate their machines, provide new capabilities to decrease operational costs for their customers, and design and produce these systems quickly. While a deliberate mechatronics-based design process with emphasis placed on up-front design and simulation helps machine builders create sophisticated machines in less time, a critical decision is the selection of the embedded controller. Selecting the appropriate controller with the right I/O capabilities and software support not only increases machine capabilities but also can reduce design time by helping engineers and machine builders reuse work done in the original design steps and by eliminating the need to rework control strategies when deploying your algorithms into the final machine or device.

Build Versus Buy

Deciding which technology to implement as the machine's primary control system is one of the first key engineering milestones. In addition to considering processor architectures, operating system capabilities, and other components, you must decide which portion of the system to design and which portion to buy off the shelf. By designing and building a custom controller, you can completely customize the end solution and optimize costs, but any design specification changes or oversights can cause lengthy and expensive delays. Alternatively, using an off-the-shelf platform increases the cost of goods sold (COGS), and you may pay for features that are not necessary for your design. Nevertheless, off-the-shelf systems typically offer a faster validation cycle and, therefore, shorter time to market.

More often than not, technical capabilities are not the determining factor when deciding between build and buy. Rather, it usually comes down to a simple financial analysis. If the return on investment of the engineering cost incurred in product development is justified by eventual profits, then you have made a good decision. To make an educated decision, you must accurately estimate the cost of building your own custom solution. This is never as simple as it seems. If you add up just the cost of the board components plus the hardware and software development time, you are grossly underestimating the total investment. You need to consider other "hidden" costs before you accurately assess the true cost of the job.

For example, manufacturing and inventory costs typically account for an additional 20 to 30 percent of the COGS of the system. In addition, on average about 30 percent of total software development time is spent on operating system, driver, and middleware development – though by choosing a packaged platform with integrated hardware and software, you can eliminate the need for this "board bring-up." Also, you need to account for other hidden costs including environmental regulations, validation, end-of-life components, and last-minute specification changes forcing design alterations and complete redesigns. Finally, the least tangible but potentially most impactful cost is the opportunity cost of spending engineering time on designing this aspect of the system rather than other revenue-generating projects.

Once you assess the engineering investment, you can calculate a simple break-even financial analysis. Imagine you assign two engineers to spend nine months developing a custom board from specs to shipping, for an investment of approximately $300,000 (USD). Next, you spend $25,000 each on pre-equipment, prototyping, prerelease units, and tooling and other nonrecurring engineering expenses, raising the total investment to $400,000. After you complete this effort, the custom design cost of goods is $400 less than an off-the-shelf platform.

$$\frac{Engineering\ Investment}{Cost\ Reduction} = Break\ Even\ Point$$

$$\frac{\$400{,}000}{\$400/unit} = 1000\ units$$

Equation 1. You can use a simple equation to calculate the break-even point of any custom design.

Using Equation 1, you see that the investment breaks even at the 1,000th unit and is not profitable until you ship the 1,001st unit. Again, this does not account for the other "hidden" costs discussed above. However, if you choose a packaged embedded system, you can shorten your time to market, and the early profits pay for cost optimization and feature improvements. In this way, you amortize your investment over the life of the product rather than investing all of the money in up-front development.

With the Graphical System Design calculator (www.ni.com/buildvsbuy/), National Instruments offers an online tool that helps machine and device builders evaluate and quantify the cost of both approaches.

Controller Options

Machine and device builders have a wide variety of different controller options. Ranging from traditional programmable logic controllers (PLCs) to modern programmable automation controllers (PACs) and industrial PCs to custom hardware, each approach provides distinct advantages and disadvantages.

Most of the platforms share the same basic electrical components: a microprocessor and associated memory to run the control code, I/O modules to convert sensor signals to digital data, a bus to pass data between the modules and the processor, and communications ports to program the controller and pass data between networked devices. Where these devices differ is in the actual components used and the software to program these devices.

PLCs are specialized industrial computers that were first introduced in the 1960s to replace relay banks for digital control. Programmed with vendor-specific software, they have a fixed execution model with a reliable and easy-to-use scanning architecture where the control engineer is concerned only with the design of the control code because the input cycles, output cycles, and housekeeping cycles are all performed automatically. This makes them particularly popular in applications where nonprogrammers – such as end customer maintenance staffs who update and modify the control code – need access to the controllers. This rigidity also allows PLC designers to optimize their designs and select lower-cost processors and memory architectures. However, the strict software architecture can also make it inflexible for custom applications such as communications, data logging, or custom control algorithms.

PCs are general-purpose computers initially designed to handle a variety of non-real-time applications. Unlike a PLC, the typical PC uses rotating media and is not designed for rugged industrial use. On the positive side, however, PCs are based on physical and electrical standards that allow easy expansion and system upgrades. Compared to a PLC, a typical PC uses a more powerful floating-point processor and includes more memory, making it suitable for more intensive data-processing applications. The software on a PC starts with a general-purpose operating system such as Microsoft Windows. Windows is capable of running many software programs, which provides great flexibility but makes the system less reliable and stable. PC applications are developed with a wide variety of common programming languages and tools, unlike the relatively restricted and specialized set of languages used on PLCs. The lack of a predefined software architecture and physical ruggedness in typical PCs makes PCs less usable for "standard" control applications. However, a PC is much more able to adapt and incorporate different types of control and processing functions for custom or complex applications. Also, the greater I/O and connectivity of a PC make it much easier to integrate disparate, even proprietary, devices into systems. Because of their powerful processing capacity, networking capability, and graphical interfaces, PCs play a key role in supervisory and advanced control, human machine interfacing, data logging, and enterprise communication.

Custom hardware, by definition, is specialized circuitry designed to meet a specific need. Because it is expressly designed for each application, it may use any hardware architecture. Typical custom hardware uses include providing add-on functionality to a larger system such as sensor-level signal processing, offloading high-speed control loops, or completely controlling high-volume applications such as embedded electronic control units (ECUs) used in automotive or custom circuitry in medical or consumer devices. Although in theory these devices could be designed to use almost any programming language, in general, the focus on high volume and low cost pushes most designers to use low-level programming tools and to write these applications in ANSI C or VHDL.

PACs combine the advantages of the high-performance PC with the flexibility of custom circuitry in a rugged, industrial-grade form factor. The availability of high-performance PC components suitable for industrial environments makes it possible to create controllers with PLC ruggedness and PC architectures for performance and openness. Examples are processors with -40 to +85 °C operation and mass storage without moving parts. Additionally, the inclusion of real-time operating systems (RTOSs) such as Phar Lap from Ardence (formerly VenturCom) or VxWorks from Wind River provide reliability and determinism, which are often not available in a general-purpose operating system such as Windows. These RTOSs offer the capability to control all aspects of the control system, from the I/O read and write rates to the priority of individual threads on the controller. Individual vendors add abstractions and I/O read/write structures to make it simpler for engineers to build reliable control applications. The result is flexible software suited for multidomain applications that mix discrete, process, and motion custom control, as well as the capability to perform other tasks such as signal processing, data logging, and communication.

Figure 1. Programmable automation controllers provide the reliability of the PLC and the functionality of the PC.

National Instruments manufactures PACs that run NI LabVIEW software. LabVIEW features an intuitive graphical programming style, similar to flowcharts, to provide the functionality of a full-featured programming language with an easy-to-use interface. With a background in measurements, National Instruments is extending PACs beyond simple I/O by incorporating higher-speed measurements and machine vision capabilities. Many industrial applications collect high-speed measurements for vibration or power quality applications. The collected data is used to monitor the condition of rotating machinery, determine maintenance schedules, identify motor wear, and adjust control algorithms. Normally collected using specialized data acquisition systems or stand-alone instrumentation, the data is incorporated into a control system using a communication bus. National Instruments PACs can take high-accuracy measurements at hundreds of thousands of samples per second and then directly process the data in real time on high-performance CPUs to perform sophisticated control, monitoring, and diagnostics. Engineers also can incorporate vision into their control systems. Vision is an area of automation that has gained a lot of momentum in the last decade. In a manufacturing environment, engineers can use visual inspection to identify many flaws or mistakes that are difficult to detect using traditional measurement techniques. Common applications include part inspection for manufacturing or assembly verification such as checking for correct component placement on a circuit board, optical character recognition (OCR) to examine date codes or to sort products, and optical measurements to find flaws in products or to sort based on quality criteria. National Instruments PACs incorporate vision, high-speed measurements, logic, and motion control, eliminating the need to integrate dissimilar hardware and software platforms.

Modern PACs Eliminate the Need for Custom Hardware

Although PACs represent the latest in programmable controllers, the future for PACs hinges on the incorporation of embedded technology to eliminate the need for custom hardware. One way to eliminate this need is to use software to define hardware. Field-programmable gate arrays (FPGAs) are electronic components commonly used by electronics manufacturers to create custom chips that engineers can use to place intelligence in new devices. These devices consist of configurable logic blocks that perform a variety of functions, programmable interconnects that act as switches to connect the function blocks together, and I/O blocks that pass data in and out of the chip. Engineers use software to define the functionality of the configurable logic blocks and the way they are connected to each other

and to the I/O. FPGAs are comparable to having a computer that literally rewires its internal circuitry to run a specific application.

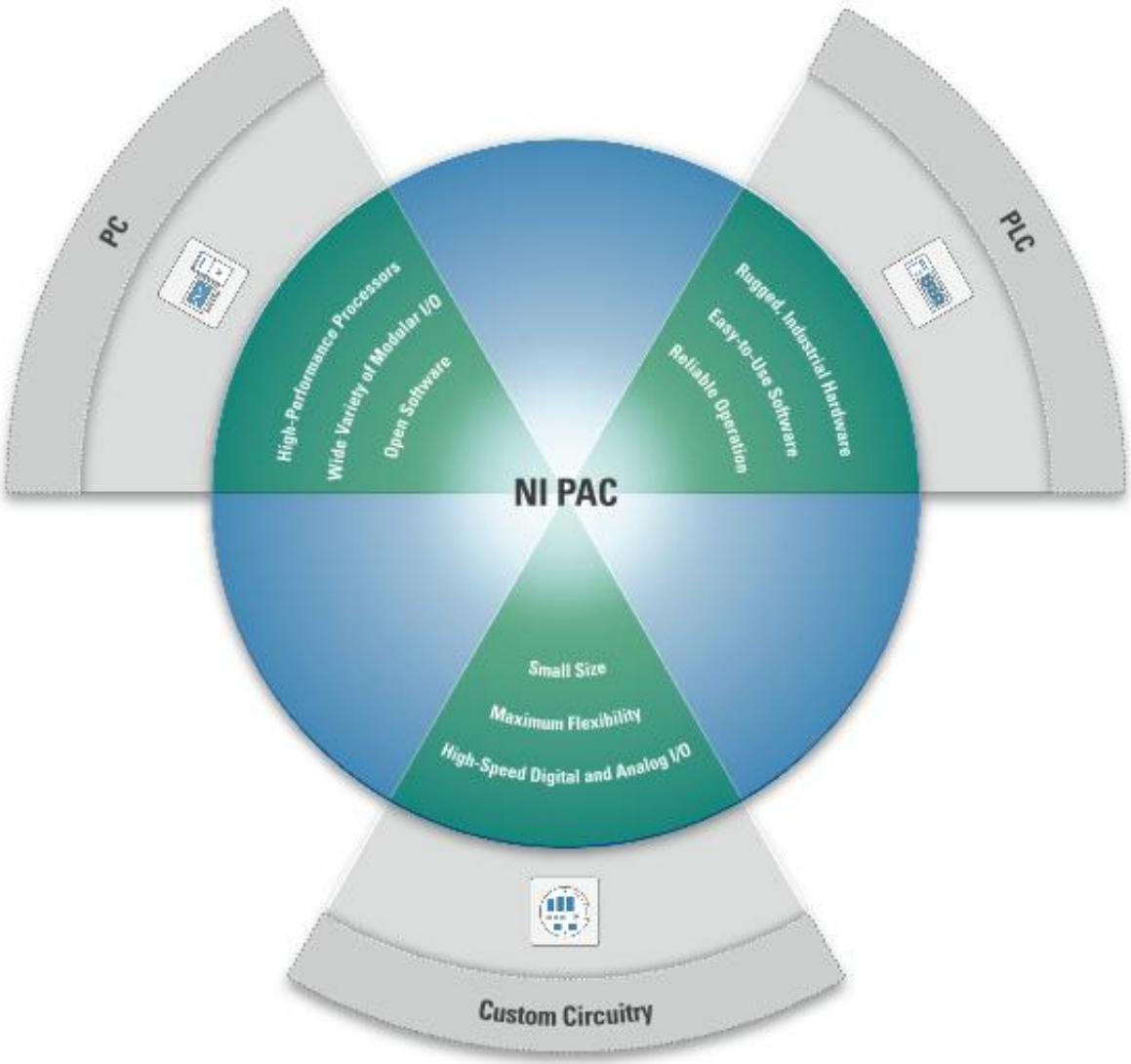


Figure 2. National Instruments PACs not only combine the features of PC and PLC platforms but also offer the performance and flexibility of custom hardware through programmable FPGAs.

FPGA technology has traditionally been available only to hardware designers who were proficient in low-level programming languages such as VHDL. However, control-engineers can use the LabVIEW FPGA Module to create custom control algorithms that are downloaded onto FPGA chips built into NI PACs. With this capability, engineers can incorporate sensor-level signal processing and time-critical control functions such as motion control and sensor health monitoring. Because the control code runs directly in silicon, engineers can quickly create applications that incorporate custom communication protocols or high-speed control loops including up to 1 MHz digital control loops and 200 kHz analog control loops.

National Instruments PACs

NI PACs offer the capability to add advanced I/O, advanced processing, and advanced communication to existing systems. This includes motion, vision, high-speed acquisition, and signals that require special signal conditioning such as electrical power measurements, strain, linear-voltage differential transformers (LVDTs), and vibration. With FPGAs, PACs perform sophisticated analysis and high-speed control. PACs can also interface to the enterprise by directly logging information for insertion into SQL databases or hosting Web pages. National Instruments offers several PAC platforms based on LabVIEW including PXI and CompactRIO.

Figure 3. LabVIEW and PXI

PXI features a modular and rugged Eurocard form factor based on CompactPCI to deliver a modular, compact, and rugged industrial system. A PXI system is controlled by an embedded controller with a high-performance multigigahertz processor. It offers easy access to cabling with connectors on the front of PXI modules. The PXI platform offers a broad range of measurement modules and connectivity to field devices using CAN, DeviceNet, RS232, RS485, Modbus, and FOUNDATION fieldbus.

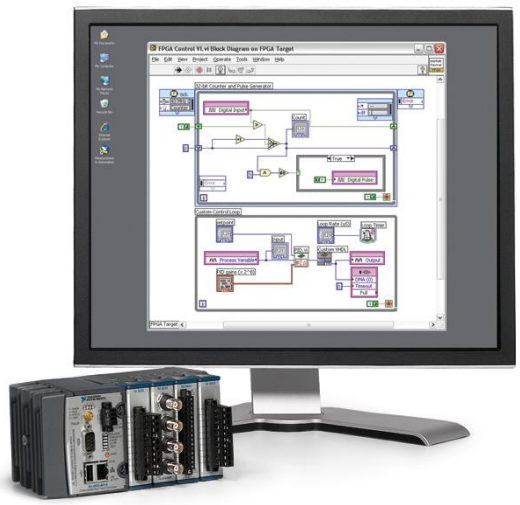


Figure 4. LabVIEW and CompactRIO

CompactRIO is an FPGA-based reconfigurable control and acquisition system designed for applications that require a high degree of customization and high-speed control. The architecture combines a real-time embedded processor for complex algorithms and custom calculations with a reconfigurable I/O (RIO) FPGA core. The CompactRIO platform accommodates up to eight analog or digital I/O modules manufactured by either National Instruments or other companies. It is ideal for complex and high-speed applications such as machine control and, with FPGA, is a good option for applications that normally require custom hardware development.

Software Reuse

One barrier that prevents machine designers from selecting the appropriate controller for a specific application is the learning curve associated with mastering new software and the ability to reuse algorithms and designs among projects. This is especially problematic when incorporating advanced functionality into machines such as machine condition monitoring, custom motion control, or enterprise integration. For efficiency, machine shops must standardize on a core set of software tools than can cover all of their application needs such as logic, process, motion, HMI, vibration, vision, IT integration, communications, and logging. To address this problem, National Instruments has invested more than 1,000 man-years of development in LabVIEW and its integration with NI PAC platforms. LabVIEW is designed to be open, allowing connectivity to databases, third-party PLCs, and serial devices and encouraging code reuse from programming and design languages such as ANSI C and The MathWorks, Inc. Simulink® environment. Every PAC National Instruments sells is programmed with LabVIEW, so designers can easily scale between platforms based on their application requirements without the need to learn new tools.[i]

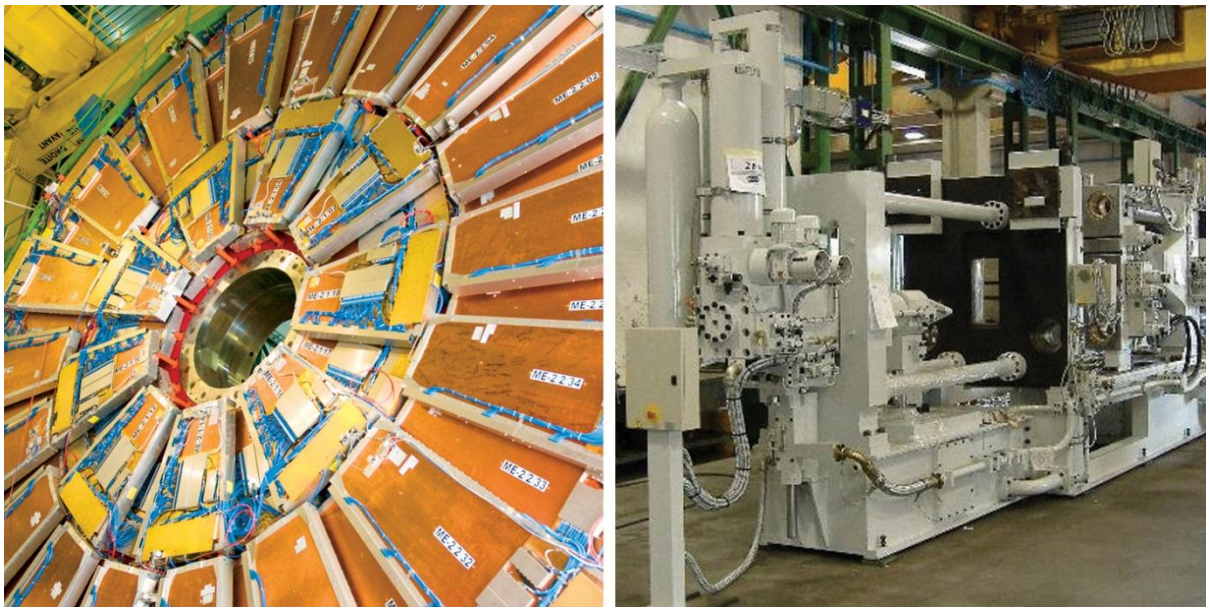


Figure 5. With a common programming language, designers have the freedom to choose the right platform for the application, whether CompactRIO for high-speed control of die-casting press machines at EUROelectronics or PXI for distributed control of 108 collimators for particle research at CERN.

System integrator Captronic Systems builds turnkey systems for control and measurement in a variety of industries. Because each system the company designs has unique needs, Captronic standardized on LabVIEW to provide the flexibility to tackle disparate applications without incurring the cost of learning new tools. These applications require different controller platforms, and Captronic is able to use the same software but select the appropriate platform for each application, whether balancing rotors for power generation turbines, performing the online inspection of 5,000 ton presses, or conducting low plug endurance tests. For instance, Captronic was contracted to build a control system for TIG welding. In this application, different objects are welded based on a recipe specific to the shape, size, and material composition of the object to be welded. Various safety interlocks are incorporated because welding current can be as high as 200 A. An operator interface is provided on a Windows PC and welding profiles can be imported from AutoCAD. This system performs motion control to position the weld object and to control the arc distance using the weld voltage level as feedback. Because of the computationally intensive algorithms and the precision motion control, Captronic chose to implement this system using a National Instruments PXI programmable automation controller programmed with LabVIEW. For another job, Captronic was hired to create the controller for the engine in an underwater crawler. This application required embedded control and logging but also needed a rugged platform that could withstand wide temperature variations and high shock and vibration. Captronic was able to continue to use LabVIEW for the development of the underwater crawler control but, in this case, ran its control code on a CompactRIO PAC. Additionally, because of the openness of LabVIEW, designers who require customization beyond that available from an NI PAC can still reuse their software investment. For example, a major Japanese automotive manufacturer used LabVIEW and CompactRIO to develop a control system in its R&D center. With CompactRIO, the manufacturer could rapidly iterate on designs and control strategies; however, the mechanical form factor did not meet the company's needs for the next stage of development. LabVIEW has the ability to target arbitrary 32-bit processors, so the manufacturer could have taken its development work and built custom hardware to meet its packaging needs. But because CompactRIO met the existing control and electrical requirements, it was ideal to

simply redesign the form factor. This customer opted to work with a custom electrical design company, Flextronics (now Prevas), who licensed the rights from National Instruments to develop custom layouts of the CompactRIO platform. This helped the manufacturer quickly transition its application directly from the CompactRIO PAC to a custom layout.

[i] Simulink® is a registered trademark of The MathWorks, Inc.

Chapter 3

Embedded Software Design

Chapter 3: Embedded Software Design

Introduction

The software architecture plays a pivotal role in the functionality of a machine. A carefully designed architecture can offer maximum flexibility to accommodate evolving design requirements in addition to room for scalability in future upgrades. This chapter covers machine design software requirements and best practices for code architecting based on hardware resources.

Software Requirements for Machine Design

Machine control systems typically incorporate a human machine interface (HMI) and a real-time control system. Real-time controllers offer reliable, predictable machine behavior, while HMIs provide the machine operator a graphical user interface (GUI) for monitoring the machine's state and setting its operating parameters. A typical machine control system contains the following base functionality:

- Analog and digital I/O
- A memory table for sharing I/O and variable (tag) values
- A sequencing engine that defines the machine behavior

In addition to these standard capabilities, machine and device builders often require support for more sophisticated functionality such as the following:

- High-speed data acquisition and analysis
- Motion control
- Vision/inspection
- Custom hardware-based signal processing
- Data logging
- Connectivity to corporate networks for database connectivity and so on

HMI applications are often implemented on a PC running Windows or more compact touch panel computers running an embedded OS such as Windows XP Embedded. HMI features typically include the following:

- Touch screen operation
- A paged display system with navigation controls
- Data entry objects (buttons, keypads, and so on)
- Alarm/event displays and logs

Control System Configurations

The simplest machine control system consists of a single controller running in a "headless" configuration (see Figure 1). This configuration is used in applications that do not need an HMI except for maintenance or diagnostic purposes.

Figure 1. A Headless Controller

The next level of system capability and complexity adds an HMI and/or more controller nodes (see Figure 2). This configuration is typical for machines controlled by a local operator.

Figure 2. A Local Machine Control System

Complex machine control applications may involve many controllers and HMIs (Figure 3). They often incorporate a high-end server that acts as a data-logging and forwarding engine. This system configuration supports physically large or complex machines. With it, you can interact with the machine from various locations or distribute specific monitoring and control responsibilities among a group of operators.

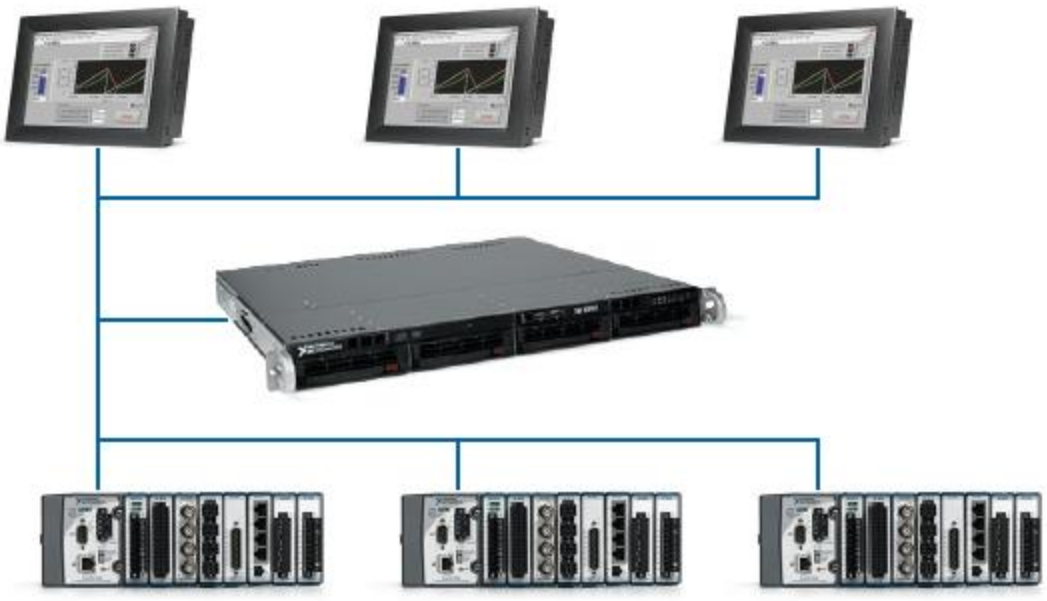


Figure 3. A Distributed Machine Control System

Control System Architectural Block Diagrams

A control system with a programmable automation controller (PAC) and an HMI contains all the software components you need to build most machine control applications. Understanding how to build a basic PAC and HMI helps you scale to any machine control system.

1. A controller has components to communicate with and interface to outside devices such as sensors and actuators, HMIs, and network devices
2. Store current data in a memory table (sometimes called a tag engine)
3. Run logic to control the machine or process
4. Perform housekeeping tasks such as start-up
5. Monitor and report system faults

An HMI has similar components except instead of performing control, it provides a user interface (UI). You can implement additional tasks such as alarm and event detection and logging on both the controller and the HMI.

Figure 4. High-Level View of the Local Machine Control Architecture

By analyzing controller operations, you can break down the system into smaller components, each responsible for a specific task in the overall application. Figure 5 shows the controller architecture and individual components of the machine control application. Some of these components are ready-to-run as part of the machine control reference architecture, while others must be developed as part of the design and implementation of a specific machine control application.

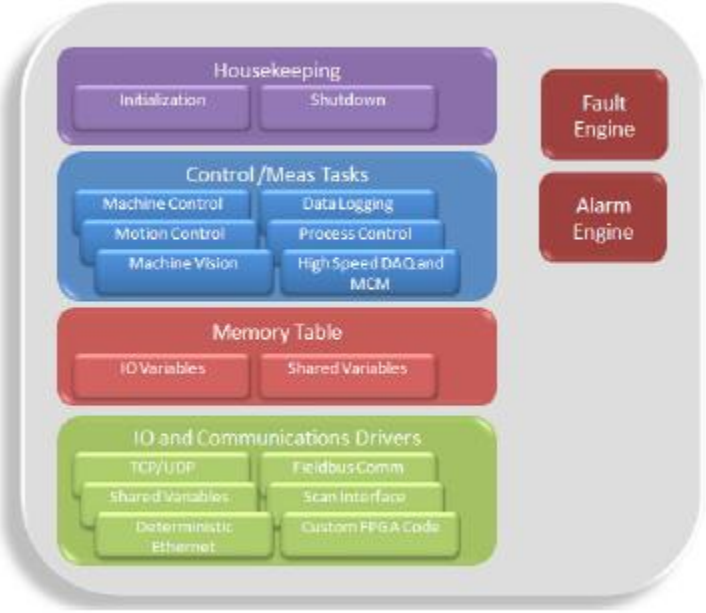


Figure 5. Controller Architecture and Components

To provide in-depth information about each of these common software components for machine control applications and their implementation with graphical design tools, National instruments created the *NI CompactRIO Developers Guide*, a comprehensive document that guides you through the

development and deployment of machine control applications. For more information, visit www.ni.com/compactriodevguide/.

I/O Access

Most control applications require data from I/O modules or devices. Using direct I/O access where subroutines directly send and receive inputs and outputs from the hardware is the ideal method for waveform acquisition and signal processing. This method offers high-speed data acquisition and provides larger sets of data to the embedded processor. This is ideal for applications such as vibration monitoring. Control applications normally use single-point reads and writes and can become very large with multiple states, all of which need access to the I/O. Accessing I/O introduces overhead on the system and can be relatively slow. Additionally managing multiple I/O accesses throughout all levels of a program makes it-difficult to change I/O and implement features such as simulation or forcing. To avoid these problems, the control routine uses a scanning I/O architecture. In a scanning I/O architecture, the physical hardware is accessed only once per loop iteration. Input and output values are stored in memory in a construct known as an I/O memory table, and the control code accesses the memory space instead of the direct hardware. This architecture provides numerous benefits:

- I/O abstraction so subVIs and functions can be reused (no hard coding of I/O)
- Low overhead
- Deterministic operation
- Support for simulation
- Support for “I/O forcing”
- Elimination of the risk of I/O changes during logic execution

I/O Scan and Memory Table

NI LabVIEW software offers an I/O option called the NI RIO Scan Interface. When you discover your CompactRIO controller from the LabVIEW project, you-have the option to program the CompactRIO using the NI RIO Scan Interface or using a LabVIEW FPGA interface (if you do not have LabVIEW FPGA installed, LabVIEW defaults to NI RIO Scan Interface).

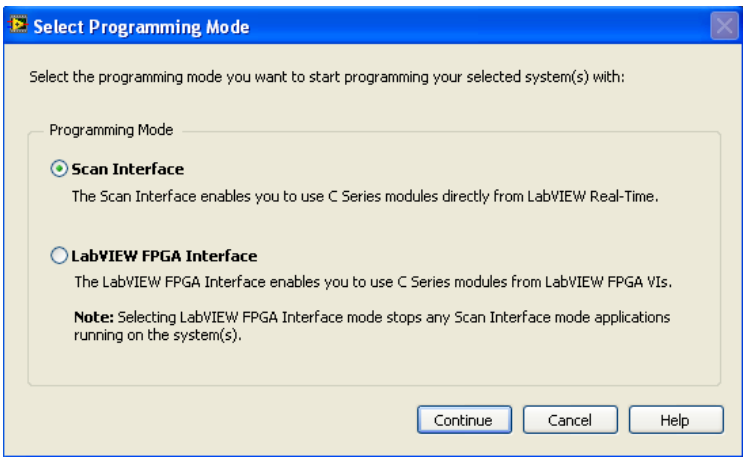


Figure 6. Select the programming mode for the CompactRIO controller.

When the controller is accessing I/O via the Scan Interface, module, I/O is automatically read from the modules and placed in a memory table on the CompactRIO controller. Based on I/O variable aliases, you can then develop the embedded application with user-defined control logic.

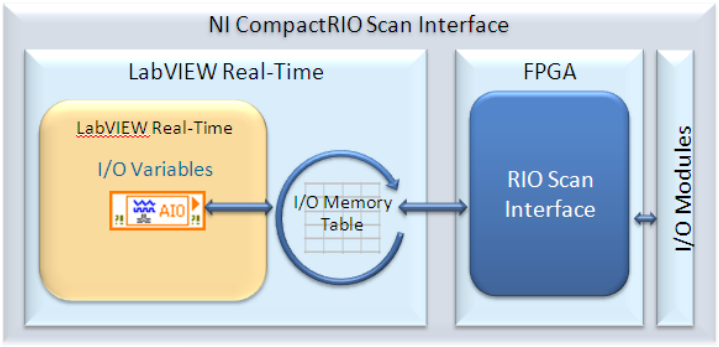


Figure 7. Block Diagram Description of the CompactRIO Scan Interface Software Components

Motion

Whether it is uniaxial, biaxial, or triaxial, motion is typically a vital part of the machine functionality. Understanding the different components including motion controllers, drives, and motors is important. National Instruments has made motion control easy by providing motion control software in LabVIEW. Along with simplifying motion control programming, NI delivers easy integration with NI data acquisition and vision products. To learn more about motion control fundamentals including software, motion controllers, drives, motors, feedback devices, and I/O, view the "Fundamentals of Motion Control[1] tutorial. After selecting the proper motion control hardware, including motion drives and motors, you need to design the software. To learn more about the software architecture, visit the Motion Control Technical Library[2]. Finally, for product specifications and pricing, visit ni.com/motion[3].

Figure 8. National Instruments offers a broad range of motion control products to implement powerful stepper or servo motor applications.

Vision

Vision inspection is another important part of machine design. Vision is used for a variety of applications such as packaging inspection, motion guidance, and assembly verification. It is important that a vision system not only perform the necessary functionality but also integrate easily with motion and other

components to provide closed-loop feedback. When choosing a vision system, you should consider your requirements for features such as camera resolution, necessary signal processing, and system integration. To learn more about these concepts, view the “What Is NI Vision?”[4] tutorial.
After determining the necessary components of a vision system, you need to design the software architecture. NI vision software is built on the LabVIEW platform for easy development and integration. For more technical information on how to automate a vision inspection system, view “Customizing the Inspection Interface in NI Vision Builder for Automated Inspection – Part I.” [5]
For product specifications and pricing, visit ni.com/vision.[6]

Figure 9. National Instruments is a leading supplier of machine vision software and hardware tools.

Machine Condition Monitoring

Machine condition monitoring is one of the most widely used techniques for monitoring the health of a machine. It includes analyzing machine vibrations to determine defects such as shaft imbalance, bearing faults, or gear misalignments. Each of these defects corresponds to specific vibration frequencies that are usually an integer multiple of the fundamental rotational frequency. By monitoring these vibrations, you can pinpoint faults and monitor their vibration levels to detect failure before it happens.

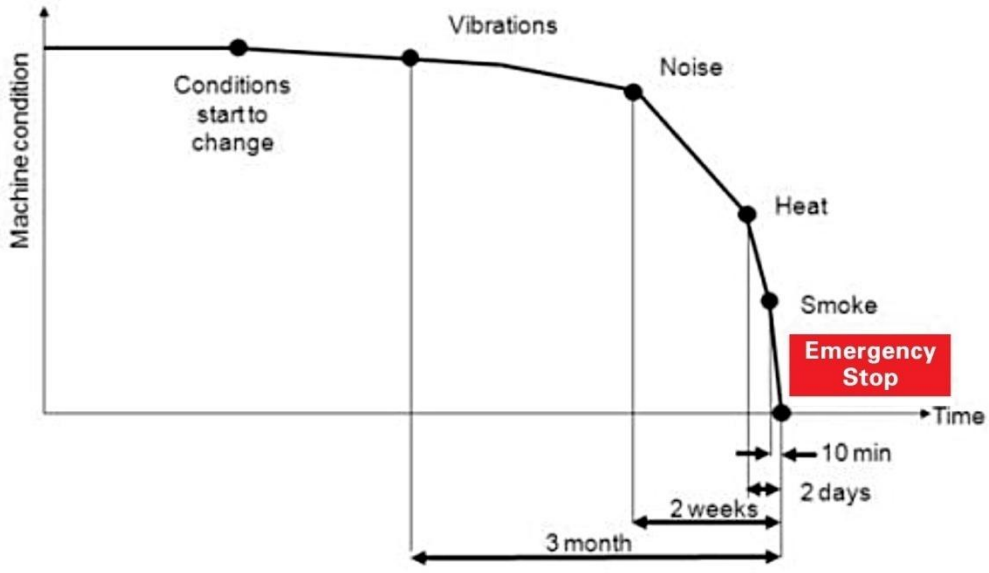


Figure 10. Describing Machine Condition Monitoring

Figure 10 shows a typical scenario of a developing machine defect. The first warning sign is always vibrations, so machine builders design vibration analysis into the machine to detect failure in advance. Monitoring the machine vibration gives you time to properly schedule maintenance and plan for the outage. To learn more about machine condition monitoring, including system architectures, sensors, hardware, software, and case studies, visit the Machine Condition Monitoring Technical Library.[7] For product specifications, visit ni.com/soundandvibration.[8]

HMI

The final component to any machine is a display to either show the machine data or provide the user with a real-time machine interface. National Instruments offers software based on LabVIEW as well as hardware to architect a display ranging from flat panel monitors and PDAs to industrial touch screens and rack-mount options. To learn more specific information about an overall HMI architecture or more LabVIEW touch panel information, read "An HMI Architecture for the LabVIEW Touch Panel Module."[9] For product specifications, visit "HMIs and Industrial Touch Panels."[10]

Web Links

[1] Fundamentals of Motion Control (http://zone.ni.com/devzone/cda/tut/p/id/3367)
[2] Motion Control Technical Library (http://zone.ni.com/devzone/cda/tut/p/id/3054)
[3] Motion (http://www.ni.com/motion)
[4] What Is NI Vision? (http://zone.ni.com/devzone/cda/tut/p/id/6618)
[5] Customizing the Inspection Interface in NI Vision Builder for Automated Inspection – Part I (http://zone.ni.com/devzone/cda/tut/p/id/6714)
[6] Vision (http://www.ni.com/vision)
[7] Machine Condition Monitoring Technical Library (http://zone.ni.com/devzone/cda/tut/p/id/6511)

[8] Sound and Vibration (http://www.ni.com/soundandvibration/)
[9] An HMI Architecture for the LabVIEW Touch Panel Module (http://zone.ni.com/devzone/cda/tut/p/id/6121)
[10] HMIs and Industrial Touch Panels (http://sine.ni.com/nips/cds/view/p/lang/en/nid/202377)

Chapter 4

Control Design

Chapter 4: Control Design

Introduction

Machine designers today are dealing with new challenges. A few decades ago, time to market was not an impending issue and different design areas had different teams with specific expertise. Teamwork was not enforced because every design team took the design inputs and generated a design output. But with the fierce competitive environment in which companies have to operate today, a modern design approach must incorporate efficiencies and symbiosis as well as a high-level view of the machine requiring design team interaction.

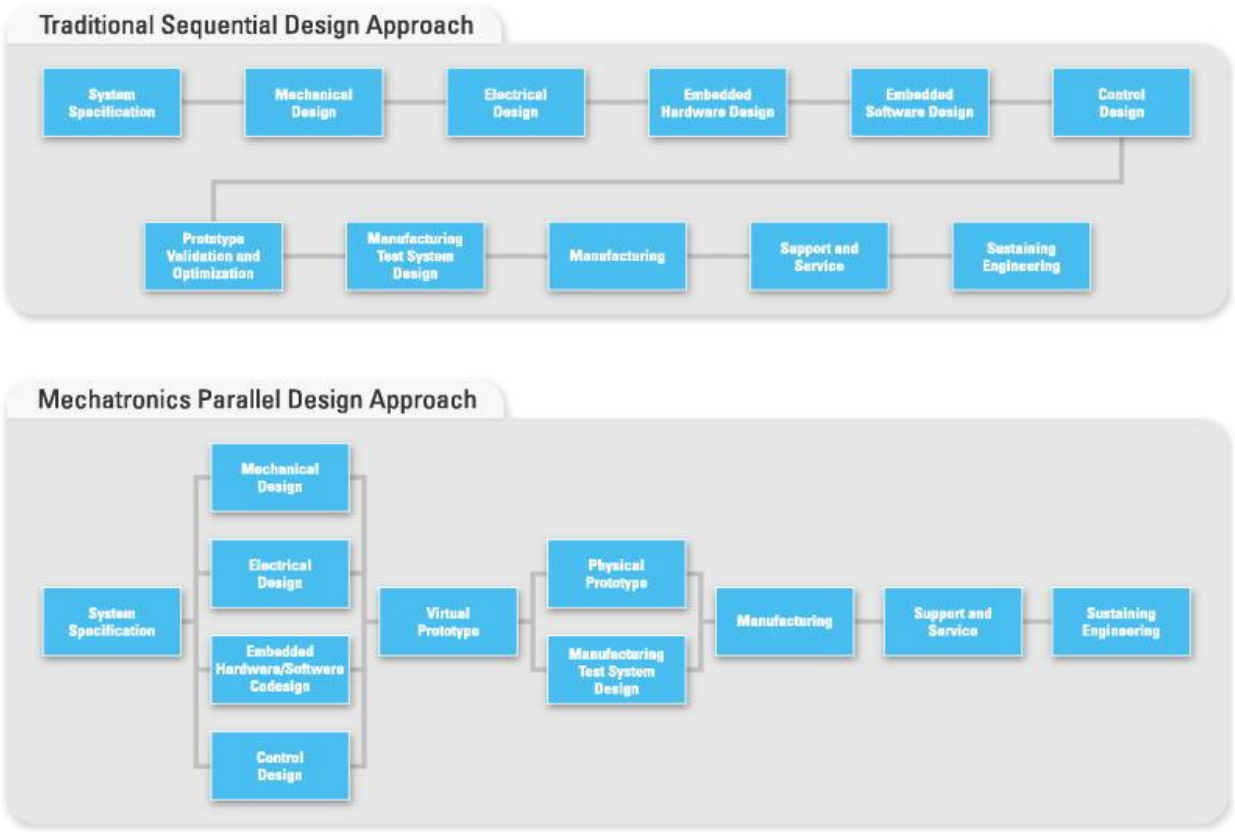


Figure 1.Traditional sequential design approaches do not meet today's machine design challenges, so designers are switching to mechatronics.

To make the design process even more challenging, machines are expected to be more complex with higher precision and tolerances. As machines require higher throughput, faster loop rates with more complex algorithms are no longer limited to high-performance, high-cost machines – they have become a commodity.

Virtual Prototyping

The need to design faster and more efficiently has changed the way design tools are perceived. In the past, mechanical engineers used finite element technology (FET)-based tools to study strain and deformation while the machine was operating, electrical engineers dimensioned motors and sensors

required for the machine to operate, and control engineers used dynamic system simulation to design control algorithms that were later recreated by embedded engineers. This process was long, with little interaction between the different design groups, and prone to errors and inefficiencies because each team failed to consider the requirements of the other design groups. It wasn't until costly prototypes were built that those errors became apparent. By then, it was too expensive and time-consuming to go back into the design cycle to track down and correct those design flaws. Software tools like NI LabVIEW for SolidWorks are driving a new way of design called virtual prototyping. With this new approach, engineers can integrate different design tools to achieve a faster view of how design changes affect the overall performance. They can then build prototypes not to check the performance but to validate the overall design. To use virtual prototyping, engineers need open design tools that offer the necessary standard technology to pass information back and forth (see chapter 1).

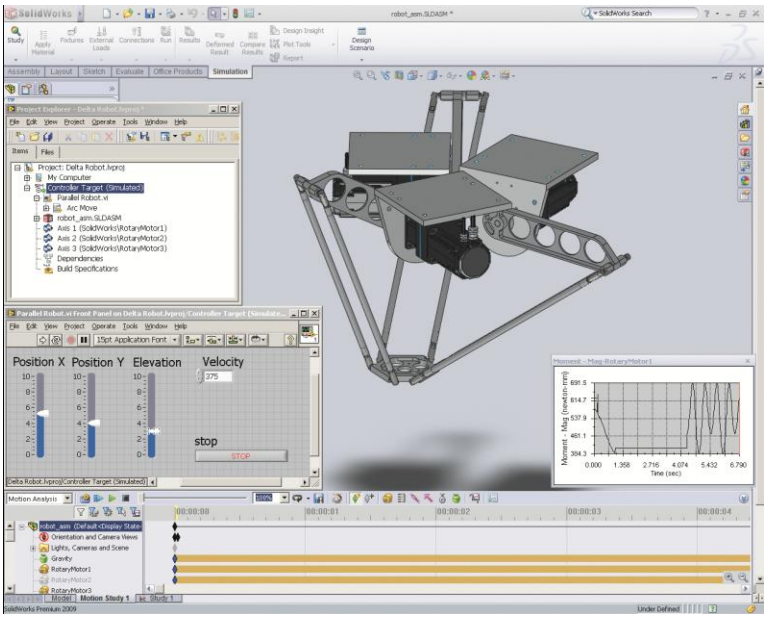


Figure 2. Machine designers can integrate NI and SolidWorks virtual prototyping tools to simulate machine operation and check for errors.

With this approach, designers reduce the time they need to go from design to a working prototype, which increases their chances of completing the project on budget and on deadline.

Advanced Control

Because proportional-integral-derivative (PID) controllers are reliable and easy to implement, PID has become the most commonly used control algorithm for machine control. National Instruments provides industrial-grade PID algorithms in the LabVIEW PID and Fuzzy Logic Toolkit and the LabVIEW NI SoftMotion Module. But with virtual prototyping, control designers now have even more options when designing advanced control algorithms for higher-performance demands. Machine designers can use their previous PID algorithms to improve current controllers by adding advanced features such as gain scheduling, integrator anti-windup, and PID gains autotuning, which are included in the LabVIEW PID and Fuzzy Logic Toolkit.

- Autotuning helps designers find a set of PID gains automatically.

- Gain scheduling helps designers use different gains on the PID algorithm depending on where the machine is operating at a given point in time.

But PID algorithms have limitations. Even when designers are taking advantage of autotuning features, they still have to tweak PID gains manually in a process that may take up to days and cause system instability. To make things worse, many system characteristics shift over time, thus requiring PID gain recalibrations. Also, PID algorithms cannot handle multiple input, multiple output (MIMO) systems, and they assume the system is linear, which might not be true, especially nowadays when components execute on the limits of specifications. The LabVIEW Control Design and Simulation Module delivers extra functionality to overcome PID limitations and offers designers absolute freedom to design their own algorithms. Module features include the following:

- Dynamic system simulation that is capable of reproducing sensor functionality, system nonlinearities, and so on
- Analytical PID that provides at design time PID gains to ensure system stability
- Model predictive control that can account for future changes in reference and disturbance

For example, Timken Bearings asked the Georgia Institute of Technology to design an active control system to automate the manual process of centering a finished bearing needed to measure the mechanical dimension. Manual centering was not only used for bearing metrology but also in production processes. Therefore, manufacturing engineers could also use an automated centering approach to reduce the cycle time of various production processes for cylindrical parts.

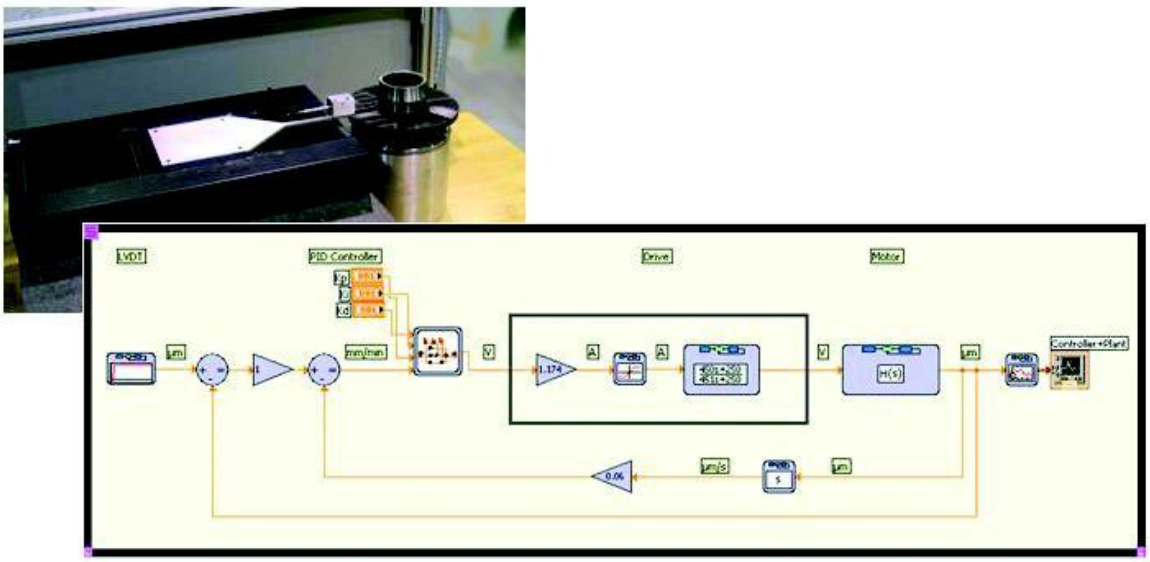


Figure 3. Georgia Tech used simulation to design an active control system to automate the manual process of centering a finished bearing.

Designers used the LabVIEW Control Design and Simulation Module to design and analyze the higher-level control loops in the system using advanced control algorithms like Kalman filters for the noisy measurement output. They modeled the filtered data with a single-lobe sine wave using least-squares curve fitting. In addition, they used simulation to explore various control loops in the system. For

example, the designers modeled a motion control loop with a subsystem for the PID control law used in the motion controller, a transfer function for the motor drive along with a saturation block, and the transfer function representing the motor dynamics. They included both position and velocity feedback in the model.

Control Implementation

Designers can choose from several hardware platforms such as programmable logic controllers (PLCs), programmable automation controllers (PACs), digital signal processors (DSPs), and microprocessors for control algorithm implementation. Before selecting one of these platforms, they need to compare the reliability, hardware availability, and cost of the different options. Hardware performance, like desired loop rate, often is a key evaluation parameter when choosing from a broad range of targets. Once they have designed and validated the control algorithm, designers must figure out how to bridge the gap between this design and validation and the implementation of a hardware solution that performs like the aforementioned design algorithm. Today, most companies have a dedicated team with embedded programming expertise that takes the algorithms developed by the control department and replicates their behavior, typically in a lower-level programming environment and without any code reuse. In addition to providing both the software-based design tools and the hardware-based deployment platform, National Instruments offers unprecedented integration between these two different realms, generally incorporating code reuse with little to no modification. Engineers can readily deploy algorithms developed using LabVIEW to a wide variety of targets, such as high-performance PXI hardware, desktop PCs, FPGA-enabled NI CompactRIO hardware, or industrial NI Compact FieldPoint hardware.

Oftentimes, platform designers choose to deploy a prototype to check system behavior, and functionality testing is different from the final deployment target. When evaluating prototyping hardware, designers may consider controller algorithm flexibility and I/O to be more important than cost. But when choosing a final deployment platform, cost becomes a main concern. Fortunately, engineers can reuse code developed in LabVIEW in different National Instruments reconfigurable I/O (RIO) platforms. When the production volume is high enough that off-the -shelf hardware cost is prohibitive, designers can save their investment in software development by using C code generation provided by the LabVIEW Microprocessor SDK. Boston Engineering provides multidisciplinary engineering services to companies that need to design electrical and mechanical systems using microprocessor-based embedded systems to control mechanical movement. The company applied this approach, using CompactRIO for prototyping an ADE Blackfin processor for final deployment, when it designed a controller for film development in a digital printing kiosk.

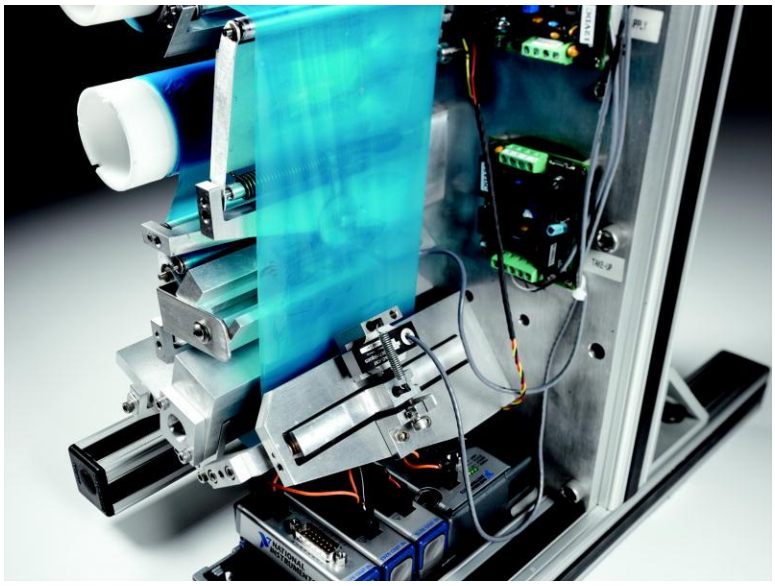


Figure 4. Boston Engineering used LabVIEW tools to design a controller for film development in a digital printing kiosk.

Summary

There is little doubt that the landscape for machine designers has changed. The demands to reduce the design cycle and design more complex machines with increased functionality have profoundly changed the design approach. Design tools are now capable of offering an unprecedented level of flexibility and speed. Algorithms and tools that were available only for high-end research a few years ago are now breaking into the industrial market along with the increased capabilities to cycle in the design between hardware and software.

National Instruments provides software tools that seamlessly interact with other design tools and can be used to deploy to hardware targets to quickly generate prototypes and evaluate and modify different controller configurations. NI also delivers the necessary path to migrate all the code written in the prototyping phase to a more custom, flexible platform without the hassle and the cost associated with rewriting code.