Designing Motion Control Systems

Kevin Hull
Supervisor, Applications and Deployment
Yaskawa America Incorporated
Introduction

• Kevin Hull
  – 20 years in the automation industry
  – Over 15 years at Yaskawa America Incorporated
    • Manufacturer of VFDs, servos, robotics and controls
      – Motion Control Division
        » Supervisor, Applications & Deployment
        » Senior Applications Engineer
        » Product Engineer
  – Over 4 years at BHS-Torin
    • Manufacturer of servo controlled spring coilers
      – Service Technician, Training
      – Co-Op
        » Technical Writer
A topic with wide reaching scope!

- **Simple**
  - Single axis, point to point travel
- **Average**
  - Multi axis systems, synchronized motion
  - Communications between several devices
  - Transfer line, large number of axes working on process
- **Complex**
  - Special purpose
    - CNC multi axis part machining
    - Robotic motion, elaborate mechanical arms driven by complex kinematic equations.
Agenda

4 Topics

• Documentation
• Motion Design
• Software
• Production & Maintenance
Documentation
Documentation

Napkin Drawings
Define the scope of the system

- **Purpose of the equipment**
  - Critical acceptance criteria
  - Describe the operation
  - Number of axes

- **Required motion**
  - Speed, Torque, Position
  - Sequence
  - Special considerations
    - Adjustments

- **Other devices relevant to the system**
  - Upstream and downstream equipment
Basic Design Document

- A repository of all materials relevant to describing the technical characteristics of a machine.
  - Invaluable when discussing the application with other team members, equipment vendors, end users, maintenance personnel.
- No right or wrong way
- No strict template
- Capture all relevant information that may not exist elsewhere
  - Speed and tolerance specifications
  - Top level block diagrams
  - Mechanical component and product sketches
  - Timing charts
Basic Design Document

- Avoid reliance on application code as the primary means of documentation.
- Examples of useful documentation:
  - Component block diagram
  - Definitions, process specific “lingo”
  - Process flow diagrams
  - Control algorithms
  - Application code “layout”
Basic Design Example: Process Flow Diagram

Card Stock

Incoming Card Stock

Web (Master)

Servo #1

Lane #1

Servo #2

Lane #2

Kicker Axis folds tabs under on back end of Card Stock

Servo #3

Lane #3

Servo #4
Basic Design Example: Application Code Layout

**TEACH MODE**
- Sensor_Teach
- Sensor_Teach
- Sensor_Teach

**HEAD LOGIC**
- MC_WriteBoolParameter (1311)
- MC_WriteBoolParameter (1310)
- MC_MotoVelocity
- AxisControl
- MC_Stop
- MC_TouchProbe (Label home)
- AxisStatus
- MC_Reset
- MC_Reset

**CAM GENERATION**
- MC_SetPosition
- MC_MoveRelative
- MC_MoveAbsolute
- MC_AbortTrigger
- MC_TouchProbe
- MC_ResetParameter
- CalcAbbingSegments
- CamGenerator
- Y_CamStructSelect
- CamFilesManagement
- CamGenerator
- Y_CamStructSelect
- Y_CamIn
- MC_Stop

**TASK 3 (Low Speed)**
- Interval: 30 ms
- Priority: 3
- Watchdog Time: 50 ms

AUTOMATE® 2011
Basic Design Example: Component Block Diagram
Basic Design Example: Communication Diagram

- Include IP Address, default gateway
- Include hardware address
- Include password
Leading edge of card is reference for label placement (offset)

Relative placement of labels to card is affected by 3 main adjustments to the master engage position (rollover):

**EngageOffset** – Approx equal to label 1 offset - gap

**Manual Offset** – User adjustable fine tune for all heads on master

**HeadOffset** – User adjustable fine tune for individual head

Up to 6 labels can be placed on the card at positions referenced from the leading edge. Dwells will be automatically inserted between the label feeds to create a single cam per product.

Slave Speed

Cam Master Cycle Position

SensorDistance

Sensor
Basic Design Example: Timing Diagram

Walking Beam Example
## Basic Design Example: Timing Calculations

<table>
<thead>
<tr>
<th>Distance from product sensor to placement point (inches)</th>
<th>Product Length (inches)</th>
<th>Minimum Product Gap (inches)</th>
<th>Label repeat interval on backing (inches)</th>
<th>Label Length (inches)</th>
<th>Label Front-Edge Overhang (inches) (enter positive)</th>
<th>Line Speed (feet / minute)</th>
<th>Network update rate (msec)</th>
<th>Y_HSpd App Task Rate (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>9</td>
<td>3</td>
<td>1.625</td>
<td>1.5</td>
<td>0.75</td>
<td>500</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>fixed</td>
<td>3.5&quot; - 12&quot;</td>
<td>1.0625&quot; - 3.125&quot;</td>
<td>0.937&quot; - 3.0&quot;</td>
<td>0.0&quot; - 0.75&quot;</td>
<td></td>
<td>500 fpm max</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network samples / second</th>
<th>feet / second</th>
<th>inches / second</th>
<th>inches / msec</th>
<th>inches / network Update</th>
<th>inches / App Task Rate</th>
<th>Minimum time between cards (msec)</th>
<th>Label Accel Time (msec)</th>
<th>Total Label Move Time (msec)</th>
<th>Total network scans for label index</th>
</tr>
</thead>
<tbody>
<tr>
<td>333.3</td>
<td>8.33</td>
<td>100</td>
<td>0.100</td>
<td>0.300</td>
<td>0.600</td>
<td>120.0</td>
<td>0.625</td>
<td>16.25</td>
<td>5.4</td>
</tr>
</tbody>
</table>

- Pure time from sensor to placement point: 31.00 msecs
- Distance lost waiting for network scan delay of 2 scans. (To get latch data): 0.600 inches
- Distance lost waiting for network scan delay of 1 scan. (Send Command): 0.300 inches
- Distance remaining to perform CamShift: 1.450 inches
- Time left to perform CamShift: 14.5 msecs
- Number of application scans (Y_HSpd): **2.42**

**NOTE:** Must be Greater than 2 scans
Motion Design
Motion Design

Mechanically driven motion

- Mechanical configurations can naturally create smooth motion

Source: www.20sim.com
Early days of electronically generated motion

• Trapezoidal
  – Simple to calculate
  – Downside: Jerk
    • Bad for mechanical components
    • Rapid changes in acceleration can lead to premature wear
    • For a given load, the higher the jerk, the greater the amount of unwanted vibration energy that will be generated.
Motion Design

- Speed
- Acceleration
- Jerk
Motion Design

Optimizing Performance

• S-Curve
  – Even a very small amount of S-Curve reduces the jerk component significantly

• Electronic cam profiles
  – Select from a range of motion profiles best suited to the application
    • Vibration reduction
    • RMS torque reduction
Motion Design

**Speed**

**Acceleration**

**Jerk**

75 mSec move

3 mSec filter = 8 x jerk reduction
Motion Design

Cam Profiles

- Use a virtual master (time clock) to drive the prescribed motion pattern
- Many formulas are readily available in the many motion control packages
- Added advantage: Same synchronization and profile at any speed
Motion Design

Simple Harmonic

Lowest RMS torque
Motion Design

Modified Sine

Generally good overall
Motion Design

Asymmetrical Cycloidal

Good for vibration reduction
Motion Design

Good for limiting velocity on longer moves
Motion Design

Asymmetric profile for vibration control
Motion Design

Cam Curve Characteristics

Sorted by Calculated Max Inertia Torque

<table>
<thead>
<tr>
<th>Curve</th>
<th>Calculated</th>
<th>Actual test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Velocity Max</td>
<td>Accel Max</td>
</tr>
<tr>
<td>Simple Harmonic</td>
<td>1.57</td>
<td>4.93</td>
</tr>
<tr>
<td>No Dwell Modified Constant Velocity</td>
<td>1.33</td>
<td>7.68</td>
</tr>
<tr>
<td>One Dwell Modified Sine</td>
<td>1.66</td>
<td>5.21</td>
</tr>
<tr>
<td>No Dwell Modified Trapezoid</td>
<td>1.72</td>
<td>4.20</td>
</tr>
<tr>
<td>Modified Sine</td>
<td>1.76</td>
<td>5.53</td>
</tr>
<tr>
<td>Modified Constant Velocity</td>
<td>1.28</td>
<td>8.91</td>
</tr>
<tr>
<td>One Dwell Cycloidal (m=1)</td>
<td>1.76</td>
<td>5.53</td>
</tr>
<tr>
<td>One Dwell Trapezoid</td>
<td>1.74</td>
<td>4.91</td>
</tr>
<tr>
<td>One Dwell Modified Trapezoid (m=1)</td>
<td>1.92</td>
<td>4.44</td>
</tr>
<tr>
<td>One Dwell Modified Trapezoid (Ferguson)</td>
<td>1.92</td>
<td>4.48</td>
</tr>
<tr>
<td>One Dwell Cycloidal (m=2/3)</td>
<td>1.72</td>
<td>6.75</td>
</tr>
<tr>
<td>Parabolic</td>
<td>2.00</td>
<td>4.00</td>
</tr>
<tr>
<td>Modified Trapezoid</td>
<td>2.00</td>
<td>4.89</td>
</tr>
<tr>
<td>Cycloidal</td>
<td>2.00</td>
<td>6.28</td>
</tr>
<tr>
<td>NG2 Curve</td>
<td>1.79</td>
<td>5.88</td>
</tr>
<tr>
<td>Asymmetrical Modified Trapezoid</td>
<td>2.00</td>
<td>6.11</td>
</tr>
<tr>
<td>Asymmetrical Cycloidal</td>
<td>2.00</td>
<td>7.85</td>
</tr>
<tr>
<td>Trapezoid</td>
<td>2.19</td>
<td>6.17</td>
</tr>
</tbody>
</table>

Significant impact on RMS torque
The remarkable result here is that the same move time of 705ms is achieved, using 253Nm of motor torque (slightly less than before), but using 45kW of infeed peak power instead of over 60kW. **This is a 23% reduction in peak power compared to the existing S-curved motion profile.** The Motor Torque vs Speed and Motor Power vs Speed charts (see Figures 6 and 7) show how the torque and power distribution is ‘front loaded’ at lower speeds using the Modified Sine profile. Even the peak current was reduced by 7% from 189.8A to 178.5A. So where is the tradeoff? For 1925mm of the 1929mm motion profile, the jerk value is less than 366m/s² (see Figure 6). It is just the final 4mm where the jerk value ramps up above 366m/s² to 513m/s². But this is a none issue since the jerk value remains well below 366m/s² for almost the entire motion profile, thus loading the mechanical system more softly than the S-curve profile, and therefore allowing the axes to quickly position the final 4mm with relative ease. So the net improvement is dramatic. A 23% reduction in peak power while maintaining the same move time!
Motion Design

Motion Design - Performance Summary

• Good
  – Trapezoidal
    • Profiles are limited
    • No transition between acceleration regions

• Better
  – S-Curve
    • Even a small amount of S-curve provides significant improvement
      – Always apply S-Curve filtering if using a triangular or trapezoidal move

• Best
  – Electronic Cam Profile
    • Usually a pre calculated, table-based approach to motion.
    • Select a profile for optimized smoothness, reduced wear, lowest transfer times, and acceptable RMS torque
Software
Software

Topics

• Background
• Application programming software
• Resulting application code that runs on the controller
• Interface to other devices
Software

Brief review of automation programming

• Very unique, vendor specific behavior
• Program mnemonics closely tied to the hardware
• Programming was very cryptic
• Nearly impossible for maintenance personnel to troubleshoot
  – Ladder logic preferred
Variety of programming styles

• Ladder logic
  – Solved left to right, or top to bottom?
  – Hybrid ladder
• Structured text
  – BASIC
  – Mnemonic language
  – Scripting language
• Sequential function chart
• Flow chart
Program Execution

- **Procedural**
  - Execute once (unless deliberately repeated)
- **Scan Based**
  - Repeating sequence
- **Multi Tasking**
  - FIFO
    - Drawback - One process can monopolize CPU
  - Round Robin
    - Drawback - Assumes all processes are equally important
  - Priority Based Scheduling
    - Drawback - Lower priority tasks can be starved.

How do you know which method your control system employs?
Software

Programming Standards

- **IEC-61131**
  - Programming and code execution
  - Hardware and software organization
  - Modular code
  - Multi controller
  - Multi tasking

- **PLCopen**
  - Motion control
  - Motion State Diagram
  - Device and vendor independent

- **PackML**
  - Machine states of operation
  - Facilitates interoperability
  - Improved maintenance and troubleshooting

Motion Control System
Software

IEC 61131

• **Benefits**
  – An inherent understandability by individuals who are proficient programmers
  – Object orientated, many tools found in most higher level programming languages
  – Multitasking capability
    • Cyclic, event, or system tasks
  – Mix and match programming styles:
    • Structured Text
    • Ladder Logic
    • Sequential Function Chart
  – Compartmental and reusable code
  – Strictly enforced datatyping
  – Complex user defined datatypes can be created.
IEC 61131 Specification

- Logic
  - Programming Languages
    - Ladder
    - Structured Text
    - Function Block Diagram
    - Sequential Function Chart
  - Program Execution Behavior
    - Cyclic
    - Event
  - Strict Datatyping

- Hardware
  - Resource(s) (Controllers)

Logic and hardware are separate
IEC 61131

- Logic
  - Programs
  - Function
  - Function Blocks
  - Data Types
  - Structures

- Hardware
  - Multi device project
  - Selectable task execution priorities and intervals
  - Synchronize external data with specific tasks
IEC 61131

• What do users hope to gain with IEC 61131?
  – Reduced learning curve
  – Reduced development time
  – Easily reusable code
  – Expanded pool of trained engineers
  – Consistency in behavior
  – Maintenance level familiarity

Reduced cost without sacrificing performance
Benefits

• Provides an abstraction layer between the user and vendor specific idiosyncrasies
• Functions blocks designed to standardize motion behavior
• Provides more predictability by following common operation patterns
• A subset of IEC 61131
Simple Specification

- Read once upon the rising edge of ‘Execute’
- Consistent interface
- Consistent output behavior
Software
PLCopen

- Consistent Interface
  - Servos
  - Virtual Axes
  - External Encoders
  - Inverters
  - Steppers
Software

Program Organization

• Basic modes of operation
  – Manual
    • Homing
    • Setup
    • Maintenance
  – Auto
    • Run
PackML

- Created by OMAC
  - Organization for Machine Automation and Control
  - PackML stands for Packaging Machine Language
  - PackML defines a standard state model for machine operation
  - Two working groups: Packaging and Machine Tool
  - Produce consensus guidelines to improve flexibility, improve capability, and reduce system integration costs
  - The primary goals are to encourage a common "look and feel" across the plant floor

Download the “PackML Implementation Guide” from www.omac.com
PackML

- Three main takeaways
  - Define the machine logic according to activity
  - Organize the application code according to various components
  - Structure the data for consistency among various equipment
PackML
State Model
Software

• PackML
  - Organize the application code according to various components
Software

Machine: Form-Fill-Sealer

- Weigh Scale
  - Equipment Module: Weigh Scale
    - Control Module: Infeed Gate
    - Control Module: Scale
    - Control Module: Dump Gate

- Bag Forming
  - Equipment Module: Bag Forming
    - Control Module: Axis Left
    - Control Module: Axis Right

- Sealer/Cutter
  - Equipment Module: Sealer/Cutter
    - Control Module: Heater
    - Control Module: Axis Sealer

- Feed Rollers
  - Equipment Module: Feed Rollers
    - Control Module: Servo 1
    - Control Module: Servo 2
**Software**

*Structure the data for consistency among various equipment*

```
ProductStruct: STRUCT
  ProductID: DINT;
  ProcessVariable: ProcessValueArray;
  Ingredient: IngredientArray;
END_STRUCT;
ProductArray: ARRAY[0..31] OF ProductStruct;
```

```
PMLnStruct: STRUCT
  UnitNode: DINT;
  UnitNodeChangeRequest: BOOL;
  ProcNode: DINT;
  ProcNodeChangeRequest: BOOL;
  CurrentSpeed: REAL;
  MatReady: WORD;
  State: DINT;
  StateChangeRequest: BOOL;
  CtrlCmd: DINT;
  Node: NodeArray;
  ProcessVariable: ProcessValueArray;
  Product: ProductArray;
  Limits: LimitsArray;
END_STRUCT;
```

```
PMLnArray: ARRAY[0..31] OF PMLnStruct;
```

```
/* PackML Structured COMMAND Array for Coordinating Machine Nodes */
PMLnStruct: STRUCT
  Status: DINT;
  Command: STRING;(* Command string *)
END_STRUCT;
```

```
PMLnArray: ARRAY[0..31] OF PMLnStruct;
```

```
/* PackML Structured ALlRM Array for Coordinating Machine Nodes */
PMLnStruct: STRUCT
  Alarm: AlarmArray;
  ModeCurrent: DINT;
  ModeCumulativeTime: DINT;
  StateCurrent: StateTimeArray;
  StateCumulativeTime: StateTimeArray;
  ProdProcess: DINT;
  DefectProd: DINT;
  ReWorkProd: DINT;
  ResetTimer: DINT;
END_STRUCT;
```

```
PMLnArray: ARRAY[0..31] OF PMLnStruct;
```

```
/* PackML Structured ALlRM Array for Coordinating Machine Nodes */
PMLnStruct: STRUCT
  Alarm: AlarmArray;
  Status: DINT;
  Command: STRING;(* Command string *)
END_STRUCT;
```

```
PMLnArray: ARRAY[0..31] OF PMLnStruct;
```

*Image of a diagram showing a control module with various elements such as Unit Machine, Equipment Module, Control Module, and Line with respective data flows.*
Software

OMAC Machine Template PackML HMI Screen

- Machine State: Stopped
- State Elapsed Time [min]: 0.32
- Reset Current Mode Times
- Mode Elapsed Time [min]: 0.32
- Reset All Mode Times

PackML State Diagram

- States: Idle, Starting, Execute, Completing, Complete
- Transitions: Clearing, Stopped, Started, Idle, Suspended, Executing, Completing, Complete

<table>
<thead>
<tr>
<th>State</th>
<th>Cumulative Time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clearing</td>
<td>0.00</td>
</tr>
<tr>
<td>Stopped</td>
<td>100.03</td>
</tr>
<tr>
<td>Starting</td>
<td>25.50</td>
</tr>
<tr>
<td>Idle</td>
<td>230.69</td>
</tr>
<tr>
<td>Suspended</td>
<td>0.09</td>
</tr>
<tr>
<td>Execute</td>
<td>270.33</td>
</tr>
<tr>
<td>Stopping</td>
<td>0.37</td>
</tr>
<tr>
<td>Aborting</td>
<td>5.95</td>
</tr>
<tr>
<td>Aborted</td>
<td>192.40</td>
</tr>
<tr>
<td>Holding</td>
<td>0.00</td>
</tr>
<tr>
<td>Hold</td>
<td>0.23</td>
</tr>
<tr>
<td>Un-Holding</td>
<td>0.19</td>
</tr>
<tr>
<td>Suspending</td>
<td>0.08</td>
</tr>
<tr>
<td>Un-Suspending</td>
<td>0.09</td>
</tr>
<tr>
<td>Resetting</td>
<td>0.09</td>
</tr>
<tr>
<td>Aborting</td>
<td>0.08</td>
</tr>
<tr>
<td>Completing</td>
<td>0.09</td>
</tr>
<tr>
<td>Complete</td>
<td>0.09</td>
</tr>
</tbody>
</table>

- Home
- Machine Warnings
- Machine Alarms
- Machine Setpoints
- Recipe

AUTOMATE™ • 2011
Software

PackML Vision

- Improve machine to machine interoperability
- Improve enterprise integration
- Improve maintainability
Production & Maintenance
Factory Acceptance Testing

• Test E-Stop conditions
  – Verify the machine can recover and resume production

• Test partial power loss to machine components
  – Multi panel systems
  – HMI, remote I/O, and controller on separate power supplies
  – Handled gracefully and recoverable
Production & Maintenance

Provide for maintenance support

• Add code to check battery status
  – Simple to include when using digital motion networks
  – Display warning on HMI
    • Include instructions
• Automatic parameter restoration
  – Speeds component replacement
  – Backup amplifier settings in controller
• Enable remote internet connection
Production & Maintenance

Provide for maintenance support

• Recalibration of axes with absolute encoders
  – Provide traditional homing method as backup
    • Move to sensor at known location

• Absolute encoder battery failure
  – Create a concise recalibration procedure
  – Verify position has been recalibrated before allowing production
Production & Maintenance

Factory Acceptance Testing

• Meeting Production speeds? Part Tolerances?
  – Benchmark the servo tuning parameters that facilitate these goals
  – Include in Basic Design Document

• Mechanical checkup
  – Vibrations?
  – Electrical Noise?
Production & Maintenance

Check for vibration!

3.7g  0.7g  8.0g
4.1g  3.2g  7.4g

200% peak torque
Vibration analysis result

62 Hz resonance
Review – Top 5 Takeaways

- Documentation
- Motion Design
- Production & Maintenance
- Software
- Hardware
#1 Invest the time into creating quality documentation
   A picture is worth a thousand words

#2 Make the highest performance machinery using the best motion profiles.
   What if your competitor does first?

#3 Use a hierarchical approach for maximized reusability!
   You’ll end up reusing code that’s been “road tested”

#4 Embrace the automation programming standards
   Increase efficiency, robustness, reusability, and complexity at once

#5 Design for simplified maintenance and maximum uptime
Contact Information:

Kevin Hull  
Supervisor, Applications and Deployment  
Yaskawa America Incorporated  
2121 Norman Drive South  
Waukegan, IL. 60085

(847) 887-7029  
kevin_hull@yaskawa.com