Outline - Roadmap

- Roadmap of Digital Control
- Control Hardware
  - Micro-controllers
  - Motion Control Cards
  - PCs
  - PLCs
- Control Software
  - Polled
  - Interrupt-driven

Roadmap of Computer Control

- Process Modeling
  - Physical System Model → Mathematical Model → Discrete transfer functions
- Control System Design
  - Select sensors and actuators
  - Choose controller topology
    - Determine its parameters to satisfy the technical specifications
  - Simulation
- Implementation
  - Build hardware and develop code
- Testing and Debugging

Control Hardware

- Custom Solutions
  - Microcontrollers
  - DSPs / DSCs
  - Mixed Signal Processors
  - Field Programmable Gate Arrays (FPGAs)
- Programmable Logic Controllers
- Motion Control Cards
- Personal Computers
  - Data acquisition cards

Micro-Controllers ("μC")

- A single chip includes
  - Fast micro-processor
  - RAM
  - Flash Memory (EEPROM)
  - Peripheral Units (ADC, DAC, PWM, counters, etc.)
- Control program is uploaded to the flash memory of the μC
- Very suitable for most industrial applications.
  - High performance,
  - Cost effective.
Motion Control Cards

• Stand-alone card developed for multiple-axis motion control applications.
• Includes usually a DSP performing all the calculations in real-time.
  – Large I/O
  – Advanced sensor interfacing
• Cost ~ 2000...3000 dollars.

Personal Computer

• Personal Computer along with a DAQ board is a good alternative for some applications with limited scope.
• PC is used to carry out the control computations in real-time.
• DAQ board(s) serves as I/O interface for the PC.
• Easy to implement but not very cost effective!

Programmable Logic Controllers

• Industrial control units.
  – Small to large scale applications,
  – Very rugged and reliable,
  – Mostly suitable for sequential control.
• In terms of functionality, they are very similar to micro-controllers except that PLCs include
  – large number of I/Os
  – Built-in units for filtering and signal isolation.
• Usually cost effective for large scale applications.
• Can be interfaced with PCs.

Control Software Development

• Assembly language
  – Generates efficient machine code
  – Developing code is a slow- and painstaking process.
• High-level languages
  – C/C++
  – Pascal
  – Basic
• Software Packages
  – Matlab
    • Real-time Windows Target (Test)
    • xPC (Deployment)
  – Labview
Matlab Real-time Windows Target

• Matlab RWT is primarily designed for rapid development / test of control-system prototypes.
  – Not suitable for deployment!
• Works in conjunction with a digital I/O card supported by Matlab.
  – RWT toolbox must be separately purchased.
  – C/C++ compiler is also needed.
    • Matlab 7.x comes with the open-source Watcom C/C++ compiler.
    • Earlier versions require an external C/C++ compiler.
• Control algorithm is then developed in Simulink environment employing the resources (I/O blocks) of this card.

MATLAB RWT (Cont’d)

• Once the Simulink diagram has been completed, the Matlab RWT converts it into a C/C++ code:
  – Compiled code runs in real-time using a modified version of the Win32 kernel.
• User can monitor the states using “Scope” as if a Simulink model is running.
• There are restrictions on RWT:
  – Only certain simulink blocks can be used;
  – Sampling rates are limited.

RWT (Cont’d)

Control Programs

• Polled Software
  – After the execution of control functions, the main program awaits the end of each sampling period.
  – Suitable for simple control systems where precise timing is not essential.
• Interrupt-driven Software
  – Main program is disrupted when a periodic interrupt is detected.
  – An interrupt service routine (ISR) is invoked to execute control functions.
  – As soon as ISR ends, the main program will resume.
  – Suitable for time-sensitive control tasks.
### Polled Control Algorithm

1. Generate or fetch command $r(k)$.
2. Read measurement $y(k)$.
3. Calculate error: $e(k) = r(k) - y(k)$.
4. Compute the correction signal:
   $$m(k) = a_1m(k-1) + a_2m(k-2) + \ldots + b_0e(k) + b_1e(k-1) + \ldots$$
5. Send $m(k)$ to the output interface.
6. Wait till the end of period $T$.
7. Let $k \leftarrow k + 1$; Go to Step 1.

### Interrupt-Driven Control Program

1. Generate or fetch command $r(k)$.
2. Read measurement $y(k)$.
3. Calculate error: $e(k) = r(k) - y(k)$.
4. Compute the correction signal:
   $$m(k) = a_1m(k-1) + a_2m(k-2) + \ldots + b_0e(k) + b_1e(k-1) + \ldots$$
5. Send $m(k)$ to the output interface.
6. Let $k \leftarrow k + 1$.
7. Return to main program.

Timer or some external event generates an interrupt triggering the ISR.