Oberon in Real-Time Control Applications

Michael A. McGaw
McGaw Technology, Inc.

Introduction

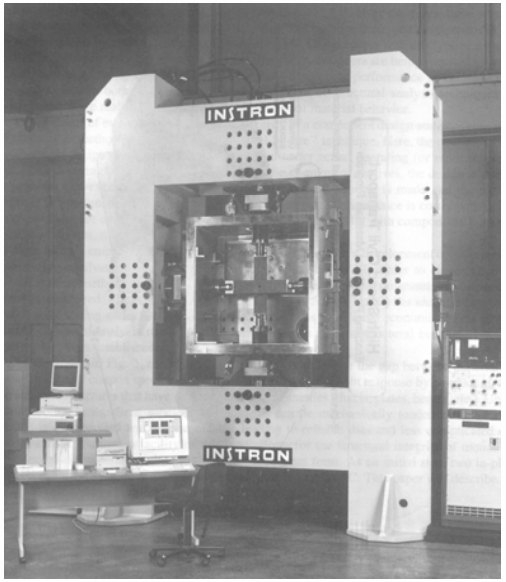
- Materials and structures testing, apparatus
- Design considerations
- Overview of digital servocontroller implementation in Native Oberon
- Aos porting

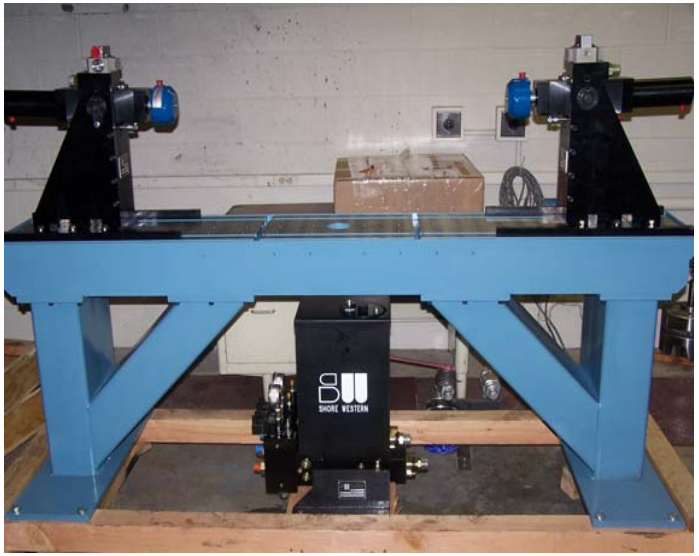
Evaluating Materials and Structures

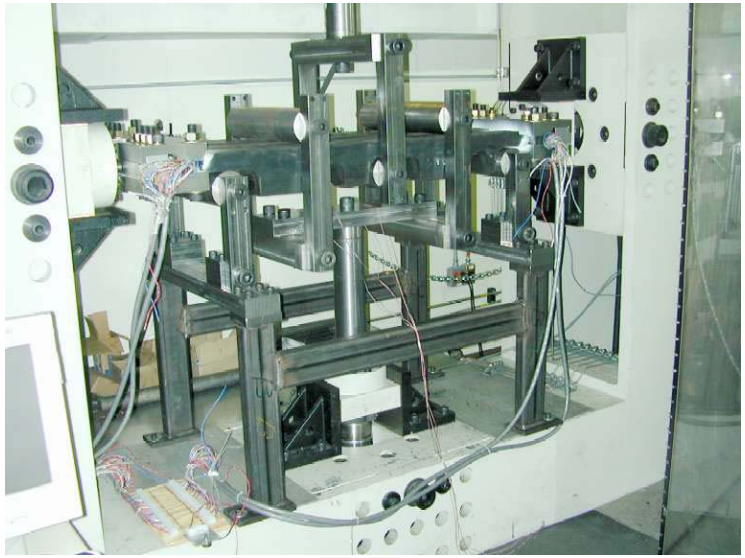
- Materials are tested to assess properties and behavior, e.g., strength, modulus, fatigue strength, etc.
- Structures are tested to assess performance and durability

Servohydraulic and Servomechanical testing systems are the preferred approaches

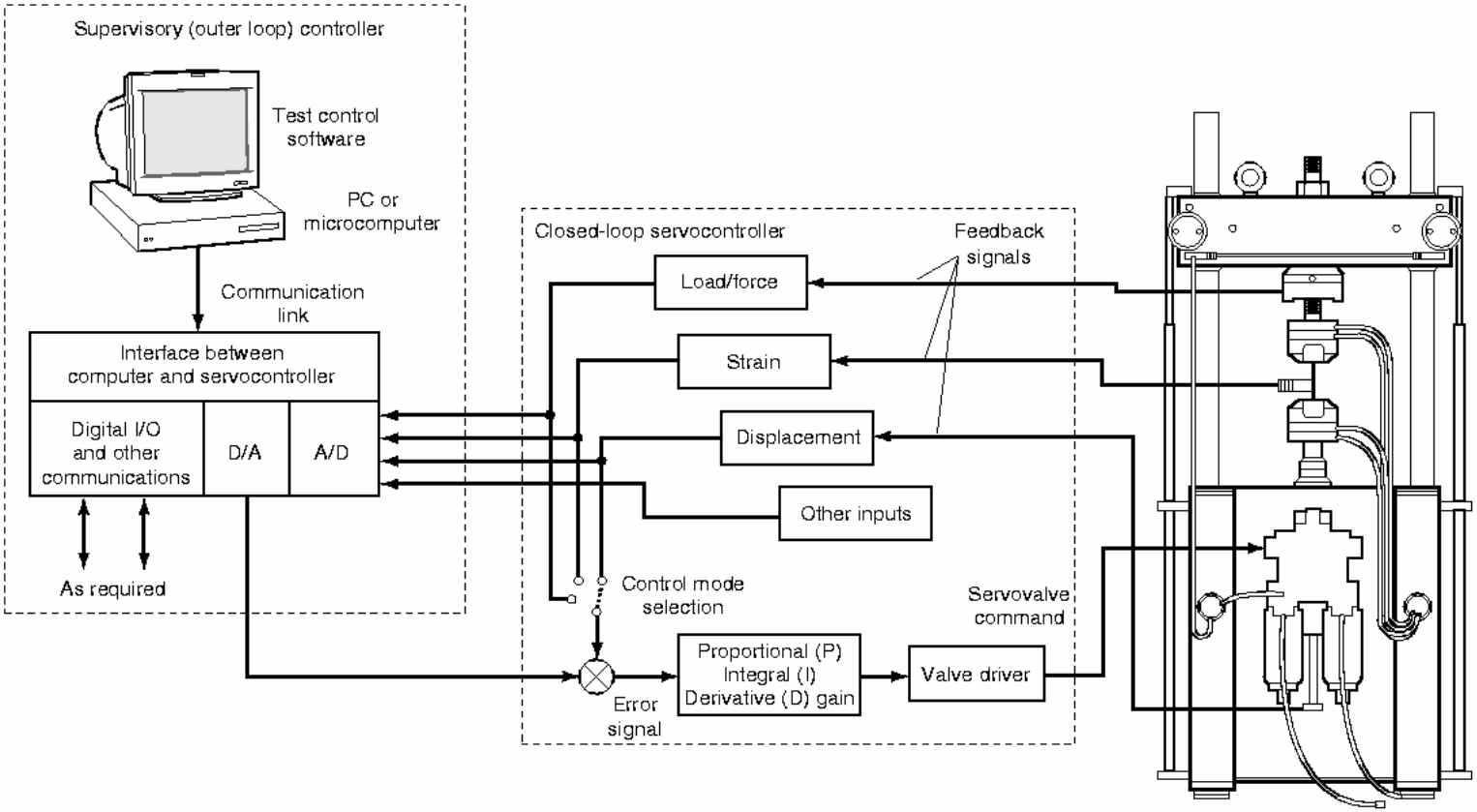
Testing Systems







Servocontroller System



Design Considerations Background

- History of Pascal and Modula-2 work, significant investments in M2 apps with preemptive multitasking (TRANSFER and IOTRANSFER)
- Needed system on bare PC hardware, embeddable, networking support and preemptive multitasking
- Watched Oberon work at ETH with great interest; a natural and obvious path from M2
  - Early 90's, experience with V3, Spirit, Linz V4;
  - Native's arrival in late 90's
    - Keys to Oberon selection (~2000):
      - Oberon on bare hardware
      - Ethernet NIC support
      - Licensing changes
      - Active Objects under development

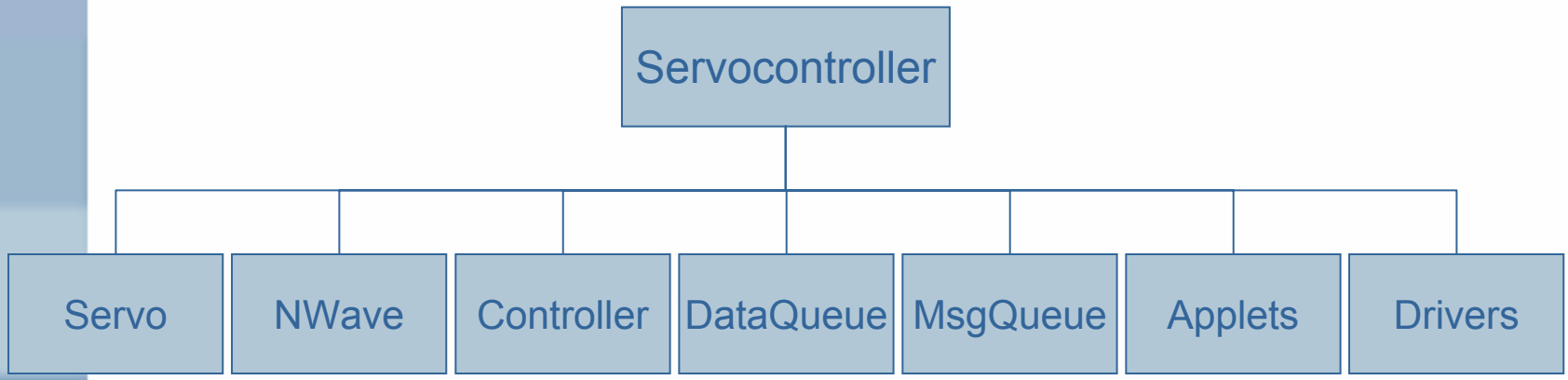
Project Goals

- Preparation: Studied similar apps in Oberon
  - System 7: Tasks vs. Threads: Wirth
  - Interferometer: Hrebabetzky
  - Model Helicopter: Wirth, Kottman, Sanvido
  - XOberon: Brega, et al.
- Operator interface in Windows
  - Commonly accepted GUI required
  - Operator must not be faced with Oberon System
- Leverage Oberon capabilities
  - Clear, concise language; fast compilation
  - Dynamic module loading
  - In-place command activation
- Realize an extendable controller

Servocontroller Design Approach

- Host-Target architecture
  - Windows hosted GUI
    - Implemented in Delphi
    - Leverage existing code base and experience with Delphi
  - Real-time on target
    - Reuse/porting of relevant M2 code
    - Oberon, in an ‘embedded’ format
    - Modifications as needed to facilitate RT use
  - High speed comms
    - Dedicated Ethernet
    - UDP
  - IDE

Target Software Architecture



- PID
- MSW
- Filtering
- Setpoint

- Run/Stop/ Reset
- Arming
- Functions:
- Sine
- Triangle
- Ramp
- Trapezoid

- Limits
- Intlks
- Hyd. On/Off
- Misc.

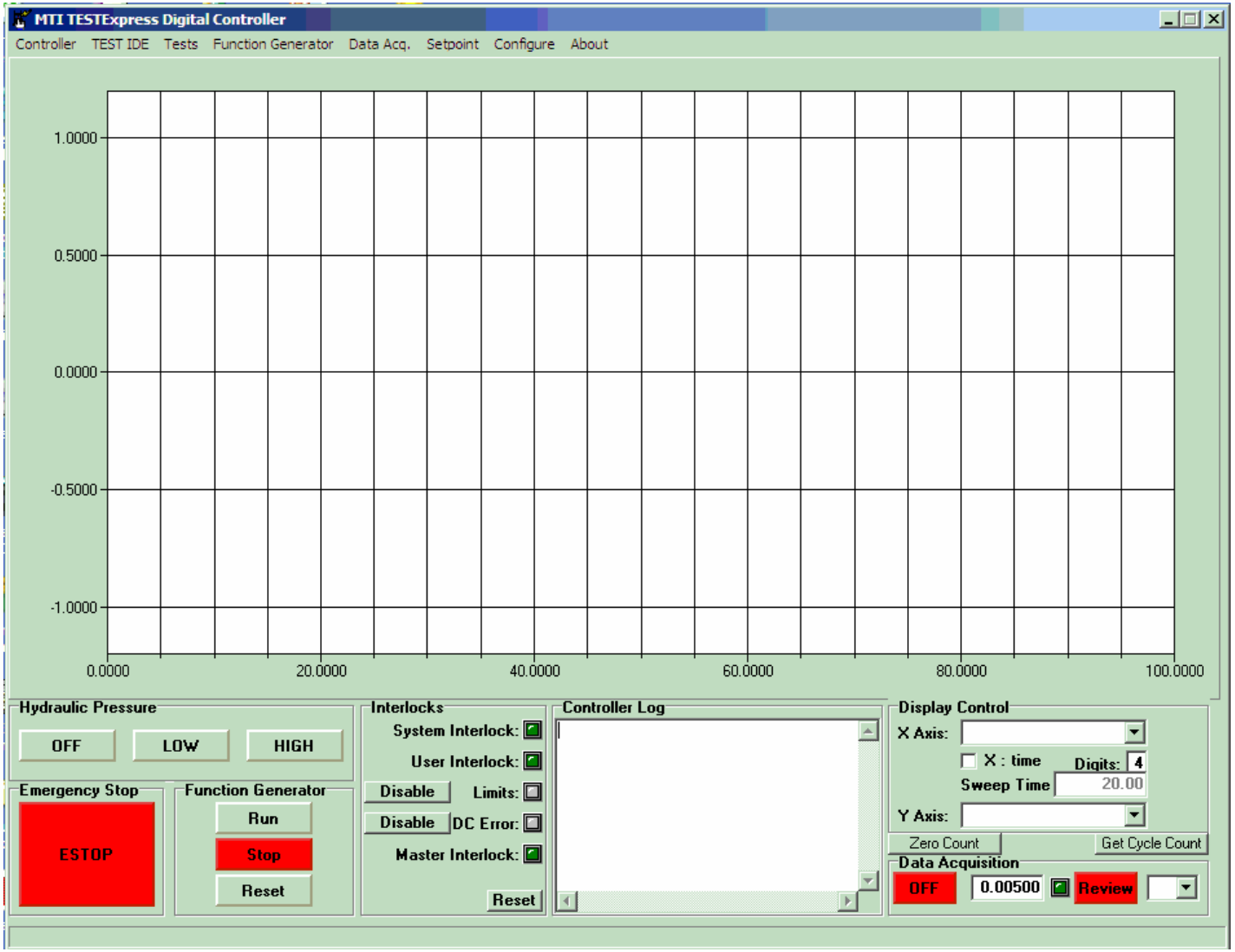
- Data Acq.

- Op. Test Mess'g

- Tensile
- Fatigue
- Slip
- AC133
- Margin
- .....

- PCI1602
- CTR05
- ....

DIGITAL Servocontroller



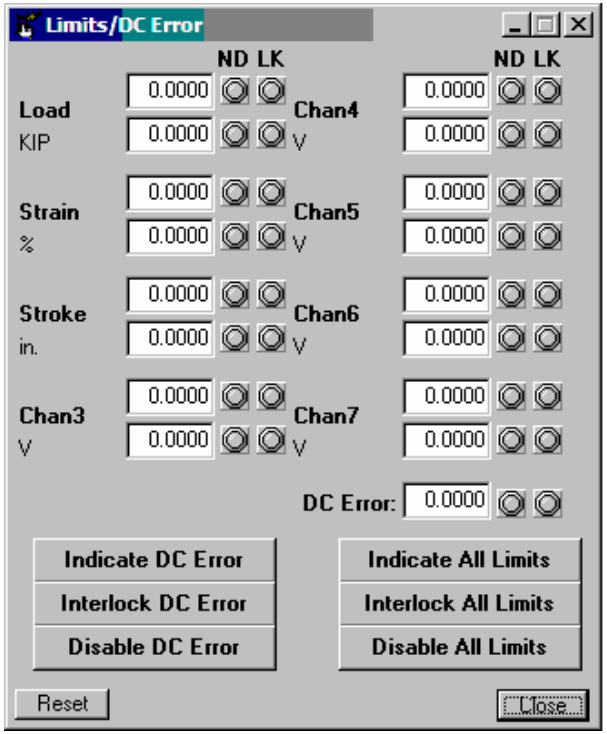
UDP comms handler

- To realize in-place command activation: Oberon.Par.text, Oberon.Call(…)
- On the host, we are sending command strings to the target: M.P {params}
  - Easy to debug
  - New features implemented quickly
- On the target, we are handling the commands in the usual way:

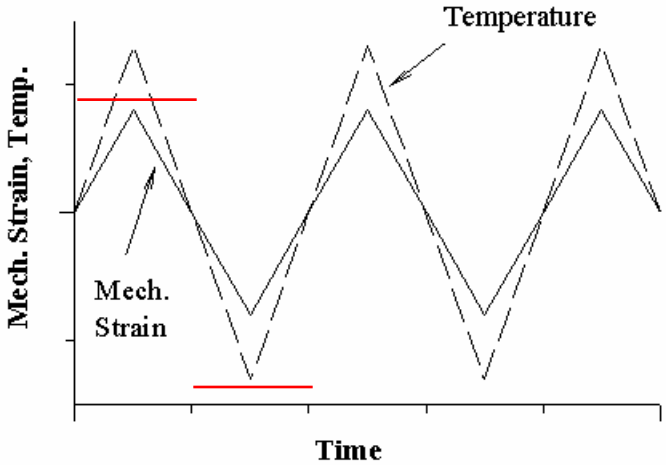
```
…
VAR S: Texts.Scanner;

BEGIN
Texts.OpenScanner(S, Oberon.Par.text, Oberon.Par.pos);
Texts.Scan(S);
…
```

Usage Example: Limit Detector




```
CommandString := 'Controller.EnableLimit ';
CommandString := CommandString +
                 IntToStr(Load) + ' ' +
                 IntToStr(Upper) + ' ' +
                 IntToStr(val) + ' ' +
                 IntToStr(Interlock);
SendCommand(CommandString);
```



McGaw Technology, Inc.

Windows-hosted IDE for Target

TESTExpress Program Editor 1.0.1.6

File Edit View Compile Options System Maintenance

demog.mod

```
1  MODULE Demo;
2
3  IMPORT Out := OCOut;
4
5  PROCEDURE Hello*(par ANY):ANY;
6  BEGIN
7    Out.String("Hello, World!");
8    RETURN NIL;
9  END Hello;
10
11 END Demo.
12
13
14 Demo.Hello
15
16 S.Free Demo~
17
18
19
```

```
06/21/07 11:43:20> Directory changed.
06/21/07 11:43:20> msg = "C:/ARCHIVE/TOSHIBA/WR2/TESTPRJ/WIN32GUI/MT/D4/MERGED7BUILD/" is current directory.
06/21/07 11:43:21> New remote dir is : C:/ARCHIVE/TOSHIBA/WR2/TESTPRJ/WIN32GUI/MT/D4/MERGED7BUILD/
06/21/07 11:43:21> DEMOG.MOD downloaded to DEMOG.MOD
06/21/07 11:43:21> DEMOG.MODi386 code generator installed
06/21/07 11:43:21> compiling Demo
06/21/07 11:43:21>    pos 57 err 20          ":" missing not done
```

5: 18 Insert

Oberon Changes

- Oberon boots off SSD with UDP comms handler running very early
- Traps must be handled, and System.Log data communicated
- Relatively minimal changes
  - GC may be app controlled, and may not be allowed to come into the system unless specifically called
  - Hrebabetzky's LibTask module found to be quite useful

Conversion to Aos

- Drivers:
  - ISRs under Oberon are prevented from using objects; this greatly limits what can be done.
  - Pre-emption and multitasking are required to implement next phase of work, e.g., recursively processed trees
  - Multiprocessor support enables radically new applications, if….
- Issues:
  - General learning curve…documentation…XML configuration…
  - Commands are very different:
    - Oberon: any procedure M.P {params} can be called in-place;
    - Aos: ONLY procedures with a special signature can be called in-place:

```
PROCEDURE DoThis*(par: ANY):ANY;
 BEGIN
  …
 RETURN NIL;
 END DoThis;
```

    - How to handle commands that must also be callable within sources?
    - Aos parameter processing is substantially different from Oberon's. How to convert code without thunking to Oberon in Aos?

Aos Conversion Example:

MyCalc.Add 5 7

Oberon:

```
MODULE MyCalc;
IMPORT Oberon,
        Out,
        Texts;

PROCEDURE Add*;
VAR S: Texts.Scanner;


    a, b: LONGINT;
BEGIN

  Texts.OpenScanner(S, Oberon.Par.text,
                       Oberon.Par.pos);
  Texts.Scan(S); a:= S.i;
  Texts.Scan(S); b:= S.i;
  Out.String("Result: ");
  Out.Int(a + b, 0); Out.Ln;

END Add;
END MyCalc.
```

Converted, Pure Aos:

```
MODULE MyCalc;
IMPORT AosCommands,
        Out    := OCOut,
        Texts := OCTexts;

PROCEDURE Add*(par: ANY):ANY;
VAR S: Texts.Scanner;
    s: AosCommands.Parameters;
    T: Texts.Text;
    a, b: LONGINT;
BEGIN
  s := par(AosCommands.Parameters);
  T := s.str;
  Texts.OpenScanner(S, T, 0);
  Texts.Scan(S); a:= S.i;
  Texts.Scan(S); b:= S.i;
  Out.String("Result: ");
  Out.Int(a + b, 0); Out.Ln;
  RETURN NIL;
END Add;
END MyCalc.
```

Conversion to Aos

- Serious Problem for RT work:
  - Active Object interrupt handling gives preemption, BUT:
  - During a GC call, Aos halts all CPUs and does *ONLY* GC on boot CPU.
  - So, we are back to an ISR in the Oberon style if we want predictability?
  - If we are to move to systems dominated by a proliferation of dynamically created objects, AND we wish to do real-time work that exploits this power, we must deal with the GC.
    - Preemptibility

Directions…

- ETH's long term view and commitment to Oberon is remarkable, and essential
- Aos: an excellent system…
  - The most serious impediment to the use of Aos in hard real time work remains the GC
    - GC preemptibility, or
    - A return to the use of Dispose as an option
  - Better command integration possible?
  - NIC support, esp. for board-level chip sets
  - Builder, w/conditional compilation
- Suggest to continue to enhance Native Oberon- a gem!
  - New class of application: microcontrollers…
  - Drivers: NICs, USB, RTC changes
  - GC changes would also be welcome here