

Introduction to OpenVMS AST Programming

Session
435

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
<http://www.rlgsc.com>

Friday, October 6, 2000
1:00 pm – 2:15 pm
Room 504

Compaq Enterprise Symposium 2000
Los Angeles Convention Center
Los Angeles, California

Introduction to OpenVMS AST Programming
Slide 1

© 1992, 2000, Robert Gezelter, All Rights Reserved

Robert Gezelter
Software Consultant

35 – 20 167th Street, Suite 215, Flushing, New York 11358 – 1731 USA

+1 (718) 463 1079

Agenda –

- This session will teach you how to use OpenVMS ASTs
- The rules presented here ARE stricter than many of the rules presented in COMPAQ manuals.
- These rules are designed to ensure correct, efficient applications.

Introduction

Basics

- Synchronization roughly equivalent to IPL level synchronization in the VMS Executive.
- High Efficiency
- Fewer limits than Event Flags

When Should You Use ASTs?

Realtime Applications
Control
Transaction Processing
Monitoring
Terminal Applications
Network Applications
Time related applications

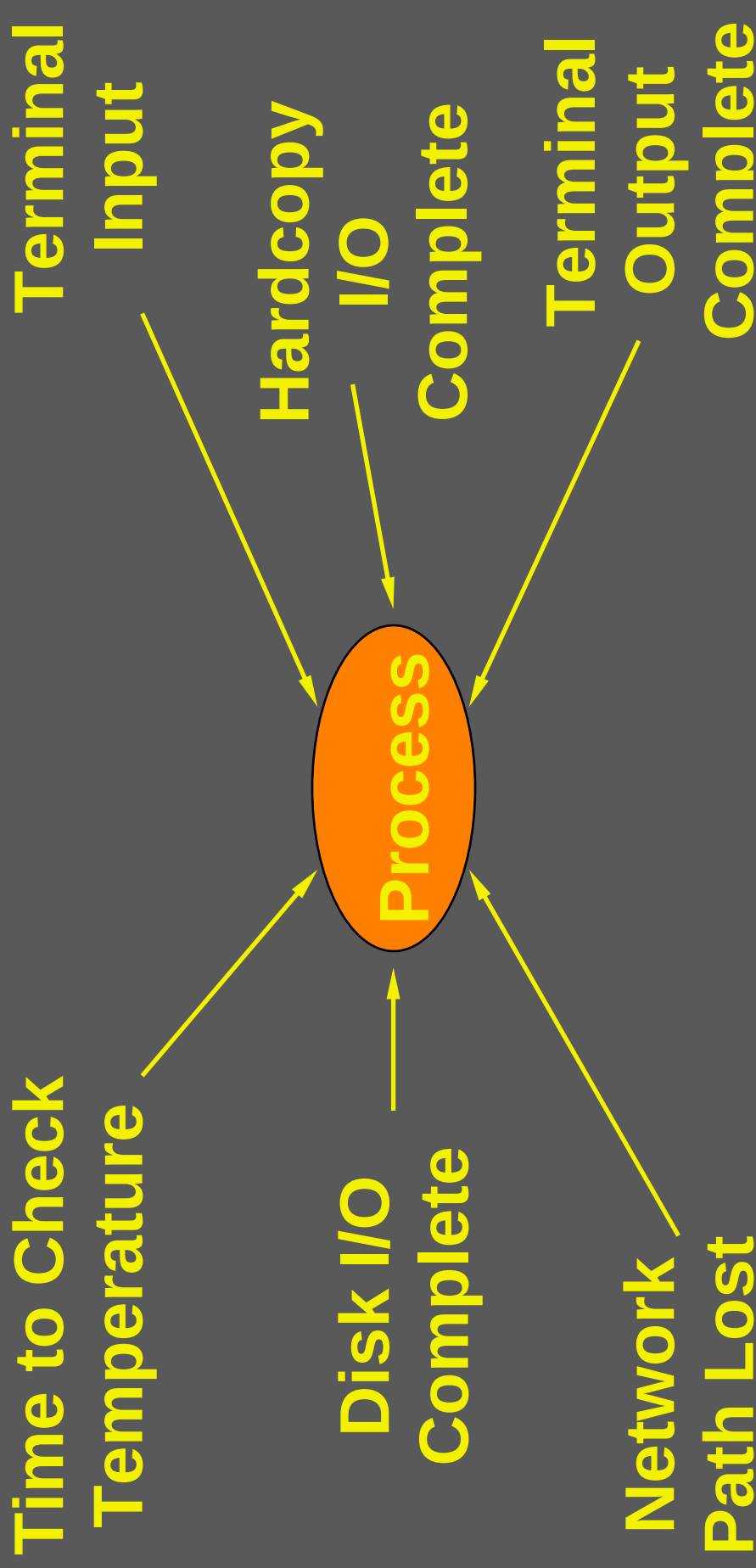
General AST Concepts

- Non-interruptable by other ASTs at same or lesser Access Modes.
- FIFO Execution.
- AST Entry is via an asynchronous(!), simulated, CALLS instruction.

Typical Event Driven Computer Applications

- Printing
- Terminal Management
- Process Control

Typical Event Driven Computer Applications



Common Root —

- External events control program
- Programs need to be efficient
- External event sequence is not under program control
- No Dispatch Routine

Generating and Processing ASTs

- Asynchronous System Services
 - \$QIO
 - \$ENQ
- Record Management Services
- Timer Services (\$TIMER)
- Declare AST Service (\$DCLAST)
- Mailboxes
- Unsolicited I/O Events

Programming Benefits

- Event Flags are limited
 - 64 Local Event Flags
 - 64 Common Event Flags (remappable)
- No limit on ASTs. AST limits enforced by
 - ASTLIM (from SYSUAF)
 - System Resources
- Capable of supporting multiple, alternative sequences without polling or increases in complexity

Keep Programs Simple

Best main program for AST based application is extremely simple.

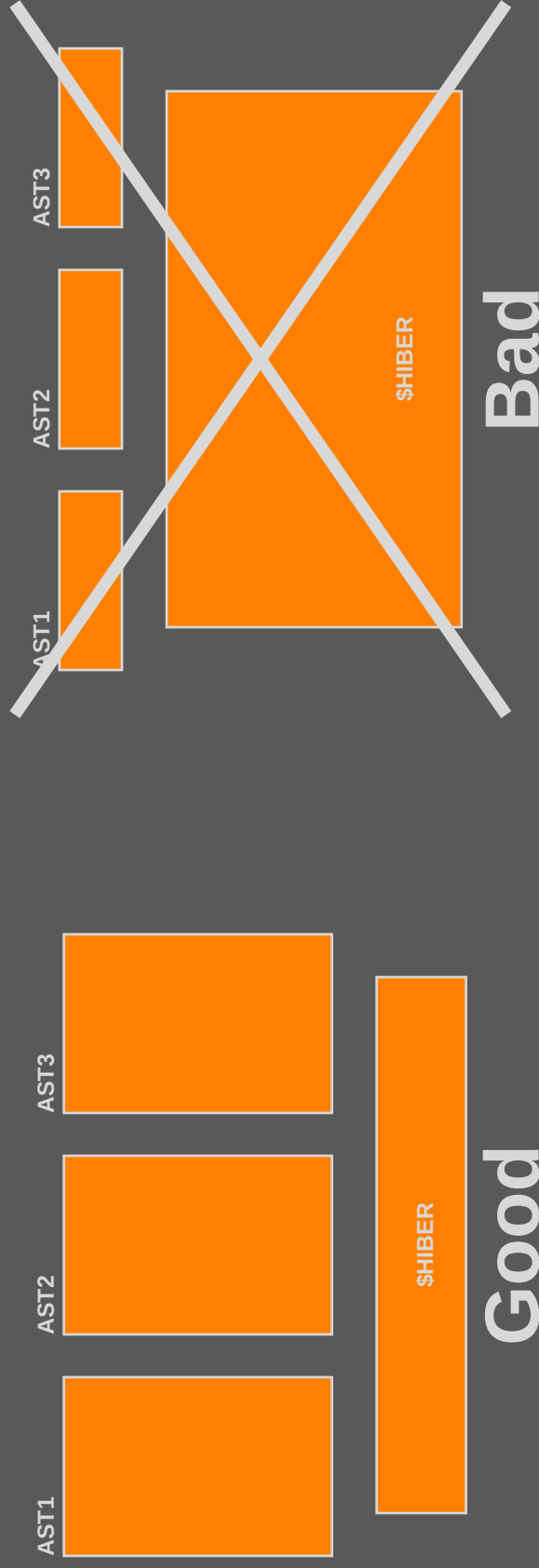
```
PARAMETER NO = 0  
CALL INIT  
EXIT_FLAG = NO  
DO WHILE EXIT_FLAG .EQ. NO  
    CALL SYS$HIBER()  
END DO
```

Keep Programs Simple

- Get in — GET OUT!
- Never use System Service
WAIT forms
- Keep Logic simple

Tricks to Getting It Right

**Do ALL Processing in ASTs.
Avoid Performing Processing at AST level
and normal Process level.**

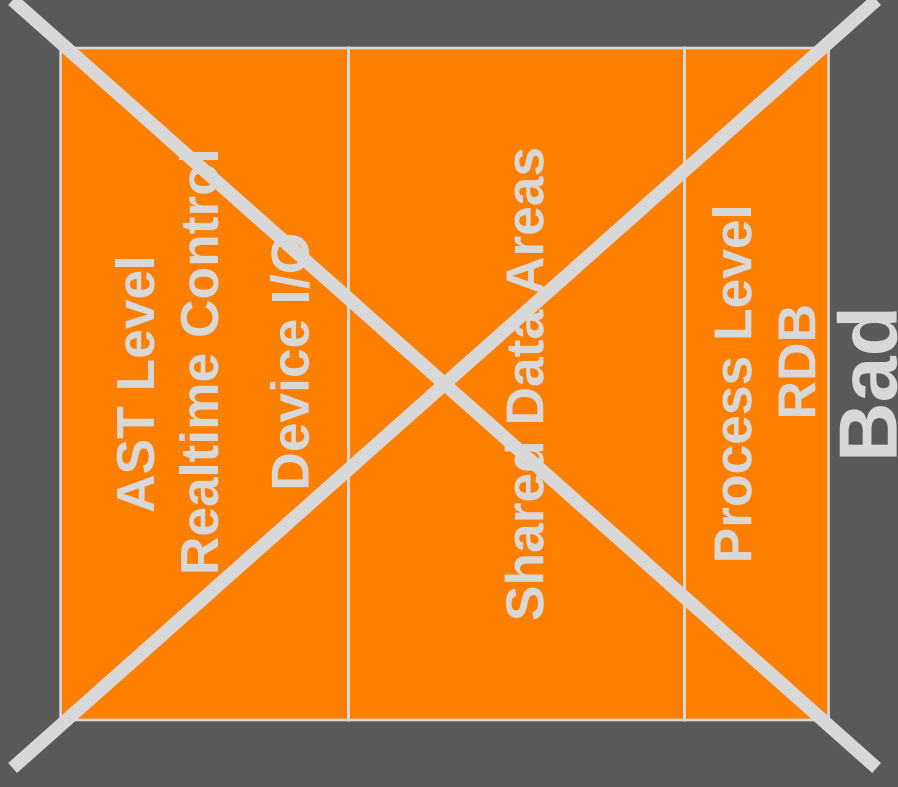
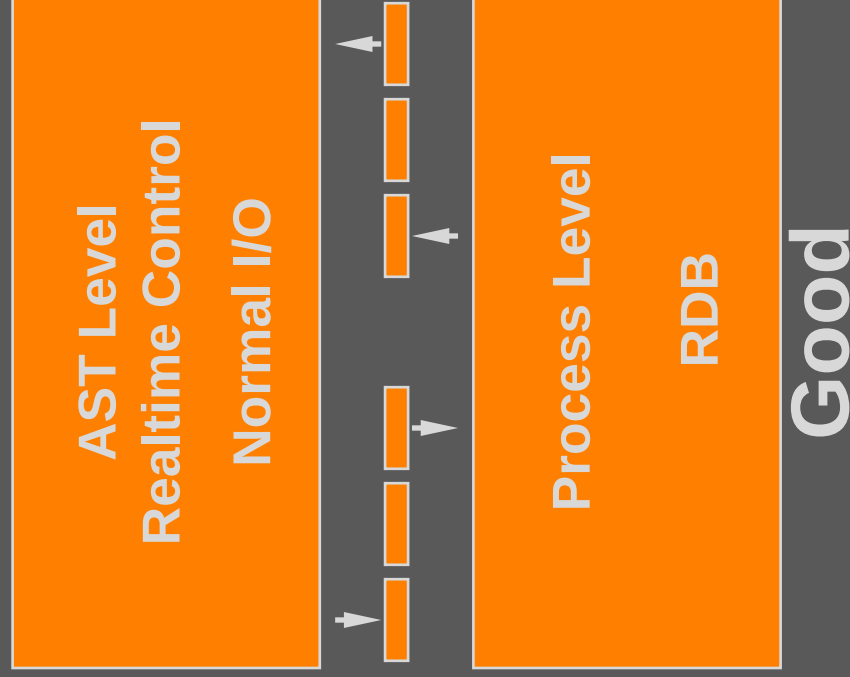


Tricks to Getting It Right

Some packages (e.g. RDB)
expect to be used only
from normal level,
NOT AST level.

Tricks to Getting It Right

Use Work, Answer, and Free Queues to communicate.



Communications Between AST Level and Process Level

- Use Queues, Insert/Remove Queue or LIB\$ routines (for HLLs)
- Be careful of queue overflows, handle overflows gracefully
- Remember to ALWAYS issue \$WAKE call!

Communications Between Process Level and AST Level

- Use queues, Insert/Remove Queue or LIB\$ routines (HLLs)
- Use \$DCLAST service to switch to AST level
- Allow ASTs to be processed in the order they are generated, DO NOT process multiple items at a time!

Initialization

Do as much initialization as possible from AST level to reduce risk of race conditions.

```
SUBROUTINE INIT                                SUBROUTINE INIT
X = SYS$DCLAST(INITAST, PARM)                  :
END                                              .
                                              .
SUBROUTINE INITAST(PARM)                      END
```

Good

Bad

Avoid Problems

- Kill bugs before they occur
- DO NOT inhibit ASTs.
Use \$DCLAST to avoid interruptions.

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
<http://www.rlgsc.com>

Session Notes & Materials:

<http://www.rlgsc.com/cets/2000/index.html>