

Productivity Techniques for Windows

Session 630

Abdul Latif and Robert Bowen
Texas Instruments

© Texas Instruments 1996

1



Productivity Techniques for Windows— Agenda

- Objectives
- Developer workstation deployment and management
- Composer Build Tool overview
- Performance tuning findings-code generation
- Performance tuning findings-installation
- Summary/Q&A

© Texas Instruments 1996

2



Objectives

- Reducing development downtime
- Improving developer productivity
- Creating a highly functional and consistent technical environment
- Building awareness of options to make the right choices (the **1st** Time!)



Productivity Techniques for Windows— Agenda

- Objectives
- ➔ • ***Developer workstation deployment and management***
- Composer Build Tool overview
- Performance tuning findings-code generation
- Performance tuning findings-installation
- Summary/Q&A



Developer Workstation Configuration– Overview

- Why do it?
 - Creates a consistent, reliable, and productive environment for Composer developers
 - Allows the developers to focus on design, coding, and testing rather than PC configuration
- Is it worth it?
 - The benefits will be evident the first time you generate code



Developer Workstation Configuration– Process

- Plan for one week of machine configuration and testing at project startup
- Try to use homogeneous machines
- Build and test one machine for each machine type (Take your time for testing!)
- Performance tune the configuration(s)
- Copy the configuration image to a file server (Save and maintain the image for later use)
- Replicate the image to all of the developer PC's
- Make PC-specific modifications (as necessary)
- Identify one person to coordinate all future modifications (maybe more for large projects)



Developer Workstation Configuration— Details

- Install all required software (operating system, network, tools, etc.)
- Configure Composer software (paths, options, etc.)
- Set the project standard Build Tool options (**WINITDEF.TGT**)
- Configure Composer Client Manager to 'talk' to all target servers
- Configure TCP/IP tools (Ping, Telnet, FTP) for maximum ease-of-use
- Set up icons for TI Books and Codeview trace example
- Modify Build Tool scripts (as necessary)



Developer Workstation Configuration— Files

- HOSTS and SERVICES files for TCP/IP (for CSE, RDA, and Client Manager)
- SYSTEM.INI variables (MaxBPS=768 under [386Enh], CommandEnvSize=1536 under [NonWindowsApp])
- AUTOEXEC.BAT variables (PATH, IEF, LIB, IEFGEN, AEHOME, AETEST, WIN3INC, WIN3LIB, WIN3EXE, IEF_BITMAP, INCLUDE, IEF_MDMODE, IEF_MDNAME, IEFUSER, IEFLAN, IEFCONST, IEF_RDSNODE, IEF_CCP, CSF_GUIM, EOLINK, DSUSER, DSPSWD)
- CONFIG.SYS variables (SHELL= /e:1024)



Developer Workstation Configuration– Debugging

- Utilize CodeView for Windows
 - **Warning:** CodeView for Windows has high resource requirements and usually takes over the machine!
- Refer to the Attached White Paper
- Build Tool allows for CodeView option (only for non-EAB's)
- Compile EAB code with debug option (-Zi)
- Link executable with CodeView (/CO)
- Execute Composer executable inside CodeView
- Set appropriate breakpoints
- Attend the EAB Technical Session for more info!!



Developer Workstation Configuration– Extras

- Try to use dedicated machines for specific tasks
 - MS Office, E-Mail, SQL reporting tools
 - Cooperative code generation
- Standardize locations of EAB's, bitmaps, RI triggers, and database loads on a file server
 - reference these directories in the Composer Build Tool (where appropriate)
- Write scripts for everything!
 - EAB Compiles, database loads, model backups, etc.



Developer Workstation Configuration– Summary

- PLAN, PLAN, PLAN!!
- Build quality machines in the beginning to reduce headaches later
- Communicate changes to all developers during the project
- Centralize configuration changes through one person (PTF's, enhancements, etc.)
- Be ready for success!!

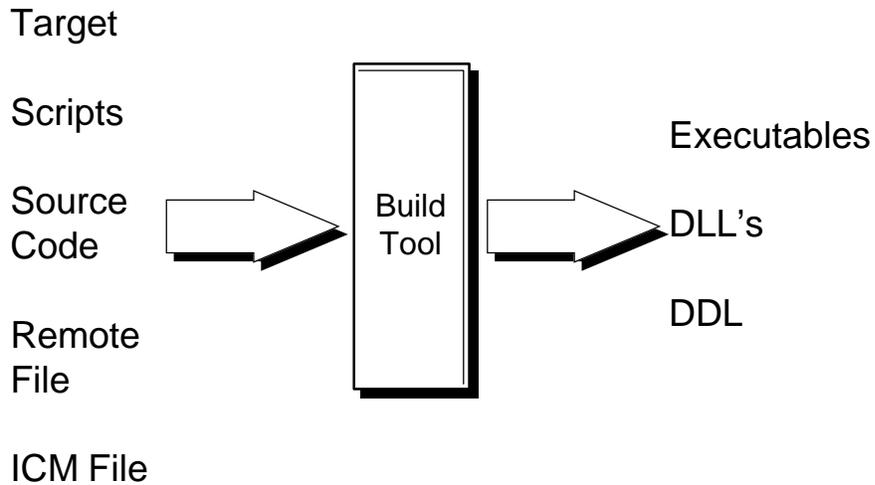


Productivity Techniques for Windows– Agenda

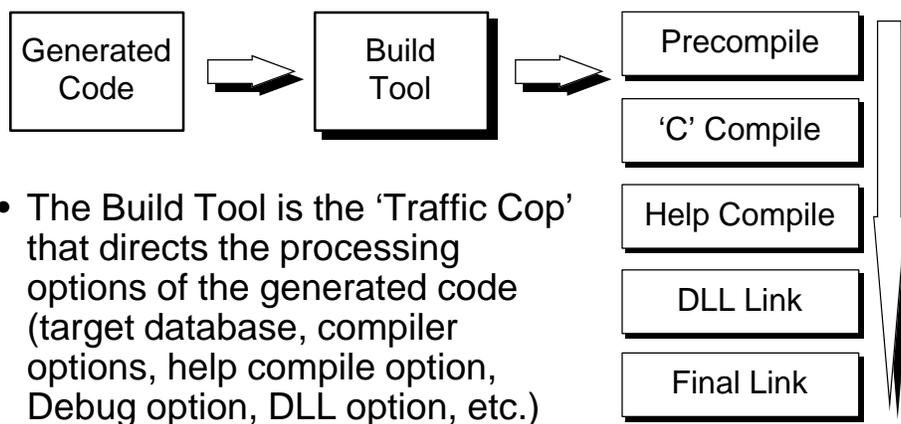
- Objectives
- Developer workstation deployment and management
- ➔ • *Composer Build Tool overview*
- Performance tuning findings-code generation
- Performance tuning findings-installation
- Summary/Q&A



Composer Build Tool Overview



Code Installation Process

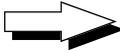


Composer Build Tool Overview–AIK

- What is the Advanced Installation Kit (AIK)?
 - An auxiliary tool used to tailor the installation of Composer-generated code
 - A “must-have” tool for the Windows target environment
 - Enabled by the Windows Build Tool
- What does it include?
 - Target files, scripts, easy-to-change installation options, automated DLL generation, and easy database portability
- Where is it?
 - In the AIK directory on the Composer 3 Installation CD. It is **NOT** on the installation menu!



Productivity Techniques for Windows– Agenda

- Objectives
- Developer workstation deployment and management
- Composer Build Tool overview
-  • *Performance tuning findings-code generation*
- Performance tuning findings-installation
- Summary/Q&A



Developer Workstation Performance Tuning

- Why do it?
 - Reduces the Code Generation / Installation 'Downtime'
(**L O N G** Coffee Breaks!!) 
 - Allows the developers to spend more time on designing, coding, and testing rather than generating and installing code
- Is it worth it?
 - The benefits will be evident every time you generate code



Performance Tuning Approach

- Created Generic Composer Workstations
- Benchmarked Code Generation and Installation Tasks using Windows for Workgroups
- Modified Configuration to Optimize Performance
- Repeated steps 2-3 
- Refer to Appendix for more details



Performance Tuning Findings

- Performance Tuning is Hard!
- It takes time to get an optimized configuration
- Configurations are never **perfectly** optimized
- It is still worth the work!
- Consider moving to Windows 95 or Windows NT



© Texas Instruments 1996

19



How to Generate Code Faster

Define Optimal Subset

- Use subset expansion (Design, Unit Test, System Test)
- Use several smaller subsets to maximize performance
- Subset size is very dependent on Load Module Packaging

© Texas Instruments 1996

20



How to Generate Code Faster (cont.)

Load Module Packaging

- For Unit Test:
 - Package Clients and Servers together in one load module
 - 1 client: 1 server: 1 load module
 - Multiple servers may be necessary (depending on design)
- For System Test:
 - Package common routines together (Error routines, List Boxes, etc.)



How to Generate Code Faster (cont.)

Regenerate Code Better

- Only regenerate the changed objects
- Schedule regeneration around meetings, lunch, etc.
- Don't regen after every change!



How to Generate Code Faster (cont.)

Manage Your Hard Drive

- Start with clean, formatted hard drive where possible
- Defragment your hard drive often (weekly)
- Increase your swapfile size to resolve some memory errors



Performance Tuning Findings— Code Generation

Things that Worked	Things that didn't Work
<ul style="list-style-type: none">• Defragment the hard drive• 32-bit file and disk access for Windows 3.11• Generating without Trace	<ul style="list-style-type: none">• Worrying about available Windows system resources• Changing swapfile settings



Productivity Techniques for Windows– Agenda

- Objectives
- Developer workstation deployment and management
- Composer Build Tool overview
- Performance tuning findings-code generation
- *Performance tuning findings-installation*
- Summary/Q&A



Performance Tuning Findings– Installation

- Refer to Appendix for detailed findings
- Modify configuration at three levels:
 - PC Hardware Configuration
 - Operating System Configuration (including Windows software)
 - Composer Software (Build Tool, Scripts, AIK)



Performance Tuning Findings– Installation (cont.)

PC Hardware Configuration

- Buy the Fastest, Biggest Machine Possible!
- Focus on Hard Drive Speed, Processor, and Memory
- Defragment the Hard Drive often (weekly)



© Texas Instruments 1996

27



Performance Tuning Findings– Installation (cont.)

Operating System Configuration

- Dedicate the machine configuration to Composer development
- Use slow, older machines for administrative tasks to minimize timeslicing (MS Office, E-Mail, Reporting Tools, etc.)
- Upgrade to Windows for Workgroups (v3.11)
- Use 32-Bit disk and file access (under 386 Enhanced in Control Panel)

© Texas Instruments 1996

28



Performance Tuning Findings– Installation (cont.)

Software Configuration

- Turn off the Help Compile option during development in the Build Tool (OPT.HELP)
- Configure the Build Tool PIF file (**WINITBLD.PIF**) for Full Screen execution
- Package small, generate fast!!
- Always generate from one local directory to minimize redundancy



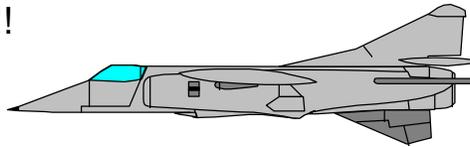
Performance Tuning Summary

- Manage the machine configurations at all levels (Hardware, Operating System, and Software)
- Manage the process to improve productivity (Subsetting, Code Generation, Test Data Maintenance, Testing, etc.)
- Plan, document, and communicate continually as the environment evolves



Summary

- Preparation pays off in developer productivity!!
- Many database options available for development and testing
- Use the tools available to reduce development and testing time
- Be ready to Fly!!



Productivity Techniques for Windows

Session 630

Abdul Latif and Robert Bowen
Texas Instruments

