Version 1.1
Porting Guide to Certification

EMBC™

Porting Guide to Certification
Agenda

Version 1.1 Topics

- Overview of Version 1.0 Issues
- Fundamental Concepts Defining Version 1.1
- Source Tree Standards for All Applications
- Test Harness Regular and Lite
- Developer Environments
- Porting and Certification

Porting Guide to Certification *Overview*

Porting/Certification is a Manual Process

- High Initial Development Cost
- Low Maintenance Cost
- Custom makefiles, no examples or tools
- Microsoft Visual C++ IDE Workspace Example
- Original Developers often Unavailable

eCL Certified!
Benchmarks Certified by ECL, an Independent Testing, Analysis, and Verification Company
http://www.ebenchmarks.com

Benchmark Source Tree Aging

- All Applications have
  - Different Harnesses, Different File/Directory Conventions
  - Compiler Warnings and Errors require Code changes
- 120 Open Bugs, Barriers to closure, Declining Activity

Version 2 is COMING. This is OUR Foundation.

EEMBC™

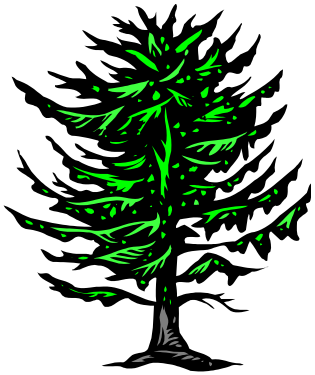
Porting Guide to Certification
Fundamental Concepts

With several years experience...

- Add relevant Developer Environments
- Enhance support for Certification
- Modernize tools, Remove barriers to cc,gcc (ANSI C)

Repair and Share Tools

- Makefile from Office
- TH Lite with CRC from 8-16 Bit
- TH Regular and Makerule.pl from Networking
- Verification from Telecom



Restore Source Tree

- Standard Directories, File Names
- Top down build of all benchmarks

EEMBC™

Porting Guide to Certification
Source Tree Standards

All File and Directory Names Lowercase

- Removes Unix/Windows naming problems

Standard Test Harness in Each Application

- TH Regular -> th, TH Lite -> th_lite
  - Application Layer - al
  - Functional Layer - src

Standard Empty Benchmark in Each Application

- Builds Lite and Regular Versions using ported harness

Standard Utilities in Each Application

- Toolchain definition, Data Collection, Makefile Dependencies

EEMBC™

Porting Guide to Certification
Source Tree Standards

Benchmark File Names

- TH Regular -> bmark.c
- TH Lite -> bmark_lite.c

VC++ Workspace Files

- One per application using application name
- World make workspace named makeworld.dsw

VC++ Project Files

- Two per benchmark TH Regular -> <benchmark>.dsp, TH Lite <benchmark>_lite.dsp
- IDE intermediate files in a sub-directory below the benchmark source.
- win32 used for Visual C++

EEMBC™

Porting Guide to Certification
Source Tree Standards

Compiler Include Path Required

- No path names in source code

Documentation

- May reside at any level in directory -> doc

Makefile builds

- A single makefile at application root
- Files included by makefile have filetype -> <xxx>.mak
- Intermediate files in directories at application root level.

Test Harness Output

- Output files from harness have filetype -> <xxx>.log

Simple Changes Allow Global Build, Run, Results Collection

EM BC™

Porting Guide to Certification
Test Harness Regular and Lite

Starting Point for TH Regular

- THV32E1 was the Standard
- All applications evolved with variations
- Multiple compiler warnings, bugs, and inconsistencies
- Networking had initial version 3.3
  - Initial work was good, but didn't work outside Networking or build under Windows

Starting Point for TH Lite

- th_lite developed as the Standard, no CRC
- 8-16 Bit had own TH Lite, with CRC
- Partial ports of Consumer and Telecom from Sergei Larin
- ARM Project with Patrick Webster

EEMBC™

Porting Guide to Certification
Test Harness Regular and Lite

TH Lite Project

- Completed Consumer and Telecom
- Propagated to other Applications
  - Required enhancements for each application
  - Enhancements rolled back to all other applications
- Enhancements derived from TH Regular when available.

Benchmark Changes

- Non Intrusive CRC Check for each Benchmark
  - 8-16 bit excluded, but standard filenames applied.
- Started standard file naming and comment blocks.
- Avoided changes within Algorithms

Demonstrated Feasibility of Standard TH Lite

EEMBC™

Porting Guide to Certification
Test Harness Regular and Lite

TH Regular Continuation

- Repaired Windows Build
- Implemented in all other applications.
- Code changes rolled back to all applications
- Common eembc_dt.h, and thcfg.h

Timing Tests

- Match TH Regular - TH Lite
- Compare TH Regular V1.0 - TH Regular V1.1
- Testing all changes to <1% timing difference

A Standard TH for All Applications

EEMBC™

Porting Guide to Certification
Developer Environments

Two Classes of Developers

- Integrated Development Environment (IDE)
- Makefile

IDE Users

- Initial testing and checkout compile
- Profiling, Interactive Debug, Tuning, Optimization

Makefile Users

- Automate build/run/data collection
- Quickly apply and measure changes to all benchmark
- May be only environment for Unix Hosts, Target RTOS

EEMBC™

Porting Guide to Certification
IDE Development

Traditional Microsoft Visual C++

- New Workspace makeworld.dsw
  - Builds All Benchmarks For All Applications
    - one click verification of a Release
- New Application Workspaces
  - Build all Benchmarks within Application
- Unique Project Names, consistent directory structure

Multiple IDE Tool Chains

- Consistent win32 directory for IDE example
- Code moved from win32 to Benchmark root
- New IDE, Sub-Directory at each benchmark.

The Visual C++ Example now models top down build with IDE

EEMBC™

Porting Guide to Certification
Makefile Development

New Support for Gnu C

- Cygwin used for unified Windows/Unix Build
- All benchmarks compiled with -ansi -pedantic
  - Possible to port without code changes

New Batch files and Scripts

- Build, Run, and Collect Results for Certification
- Windows and Unix without changes

Multiple Tool Chains

- Create a simple definition file
- All benchmarks build with new tool chain
- Visual C++ support for makefile environment included

EEMBC™

Porting Guide to Certification
Makefile Development

Perl, Awk, and uudecode utilities

- Automatically generate complex dependency files
- Extract Timing data from Test Harness output.
- Extract Verification Files from Test Harness output

Makefile Includes for Adaptation

- Tools, Directories, Compile/Link Targets, Run, and Results
- Each can be independently customized

Porting Guide to Certification
Porting Abstractions

- Abstractions for Porting
  - Platform/Processor
  - Compiler/Tool Chain
  - Subcommittees
  - Test Harness
    - Abstraction Layer
    - Functional Layer

Porting Guide to Certification Build Process - V1.1 Beta 1C

- Makefile Control
  - Tools Definitions
  - Compile
    - DIRS
    - TARGETS
  - Execution
    - RUN
  - Analysis
    - RESULTS

Porting Guide to Certification Build Process - V1.1 Beta 2

- Tools Control
- Makefile
  - Tools Definitions
    - Compile
      - DIRS
      - TARGETS
        - ITERATIONS (optional)
    - Execution (optional)
      - RUN
        - ITERATIONS (TH Regular optional)
    - Analysis (optional)
      - RESULTS

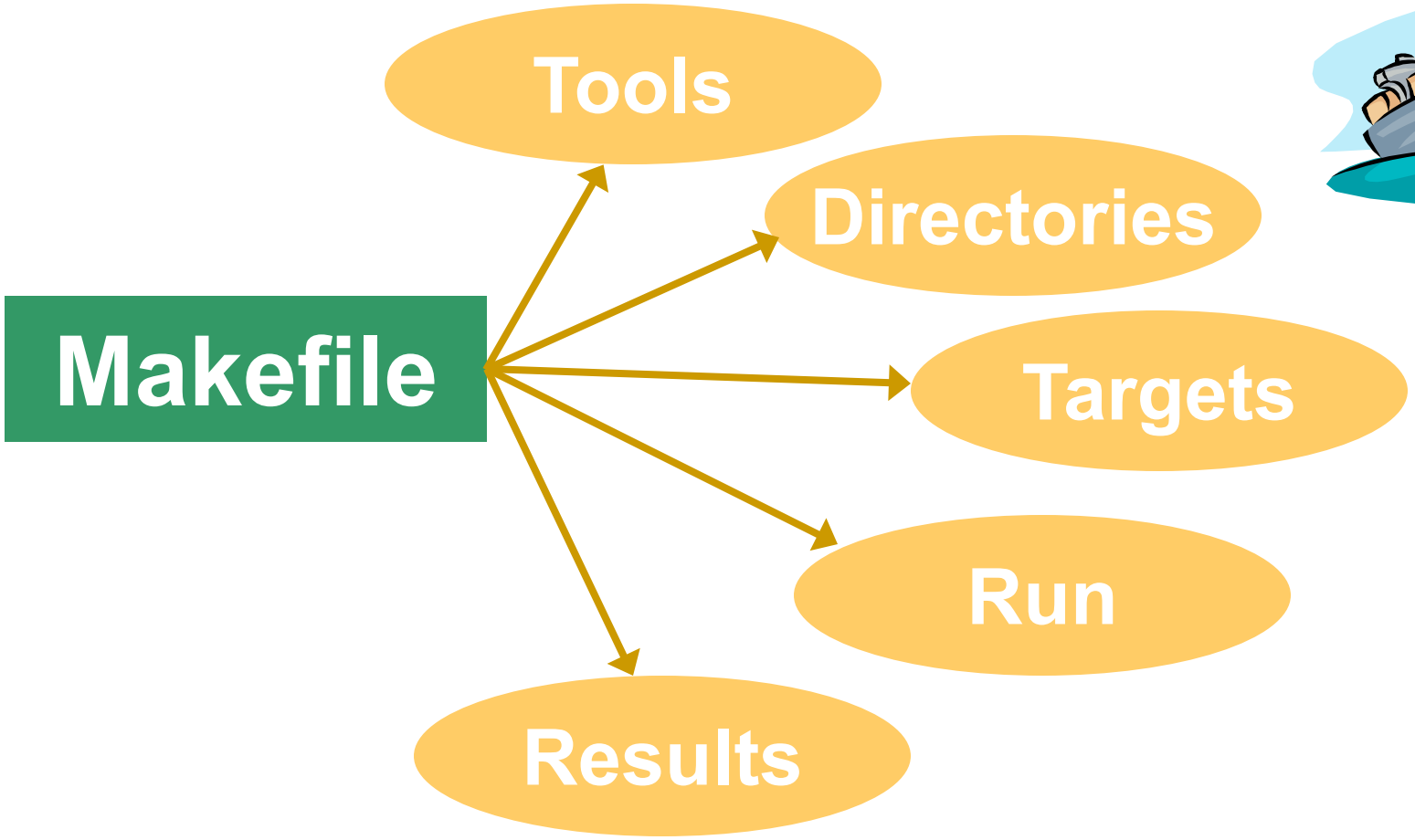
Porting Guide to Certification Source Tree - V1.1 Beta 1C

- Subcommittee
  - Benchmarks
  - util
    - make
      - Platform/Processor
      - Compiler/Tool Chain
    - awk
      - Results Processing
  - th
    - al - Abstraction Layer
    - src - Functional Layer

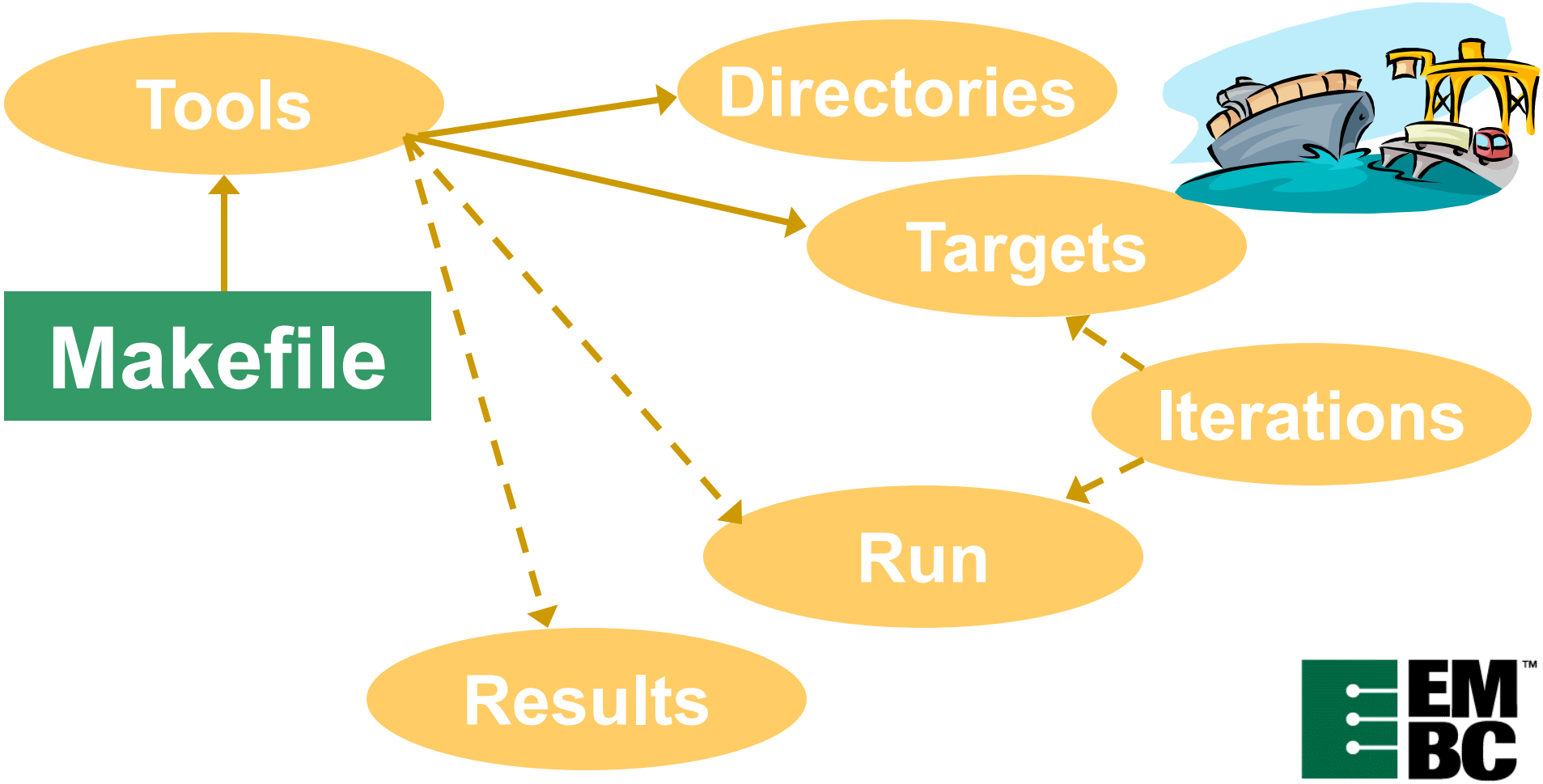
Porting Guide to Certification Source Tree - V1.1 Beta 2

- Subcommittee
  - Benchmarks
- util
  - make
    - Platform/Processor
    - Compiler/Tool Chain
  - awk
    - Results Processing
- th
  - <tools>
    - al - Abstraction Layer
  - src - Functional Layer

Porting Guide to Certification
Porting - V1.1 Beta 1C



Porting Guide to Certification Porting - V1.1 Beta 2



Porting Guide to Certification
Porting

Tools

Tool Chain Definition File

<toolchain>.mak in util/make

Definitions

- Path to tools
- Compiler, Options, and object file type
- Assembler and Options
- Linker, Options, and Executable file type
- Run, command to execute Benchmark

Makefile Variable

TOOLCHAIN=util/make/<toolchain>.mak

Porting Guide to Certification
Porting

Directories

- Directory Definition File
  - dirs.mak
- Commands to create and remove Directories
  - Object Files, Regular and Lite
  - Executable files
  - Results files

EMBC™

Porting Guide to Certification
Porting

Targets

Targets Definition File

depgen.cml

Makerule.pl scans C files generating targets.mak

Makefile ensures targets.mak is up to date

Porting Guide to Certification
Porting

Run

Execution definition file

- run.mak

Commands to run, and size Benchmark

- Direct execution, or via program defined in Tools
- TH Output re-directed to log file
- Size information re-directed to separate log file

Porting Guide to Certification
Porting

Results

Results definition file

- results.mak

Commands to extract Timing and Size information

- AWK script for Timing provided
- Custom script for sizing, dependant on tool chain format

Results are stored in log file at top of application

Porting Guide to Certification
Certification

Wherever your starting point, Version 1.1 provides a new foundation for certification

EEMBC™