EFI Application Development

Intel Corporation

Software and

Services Group

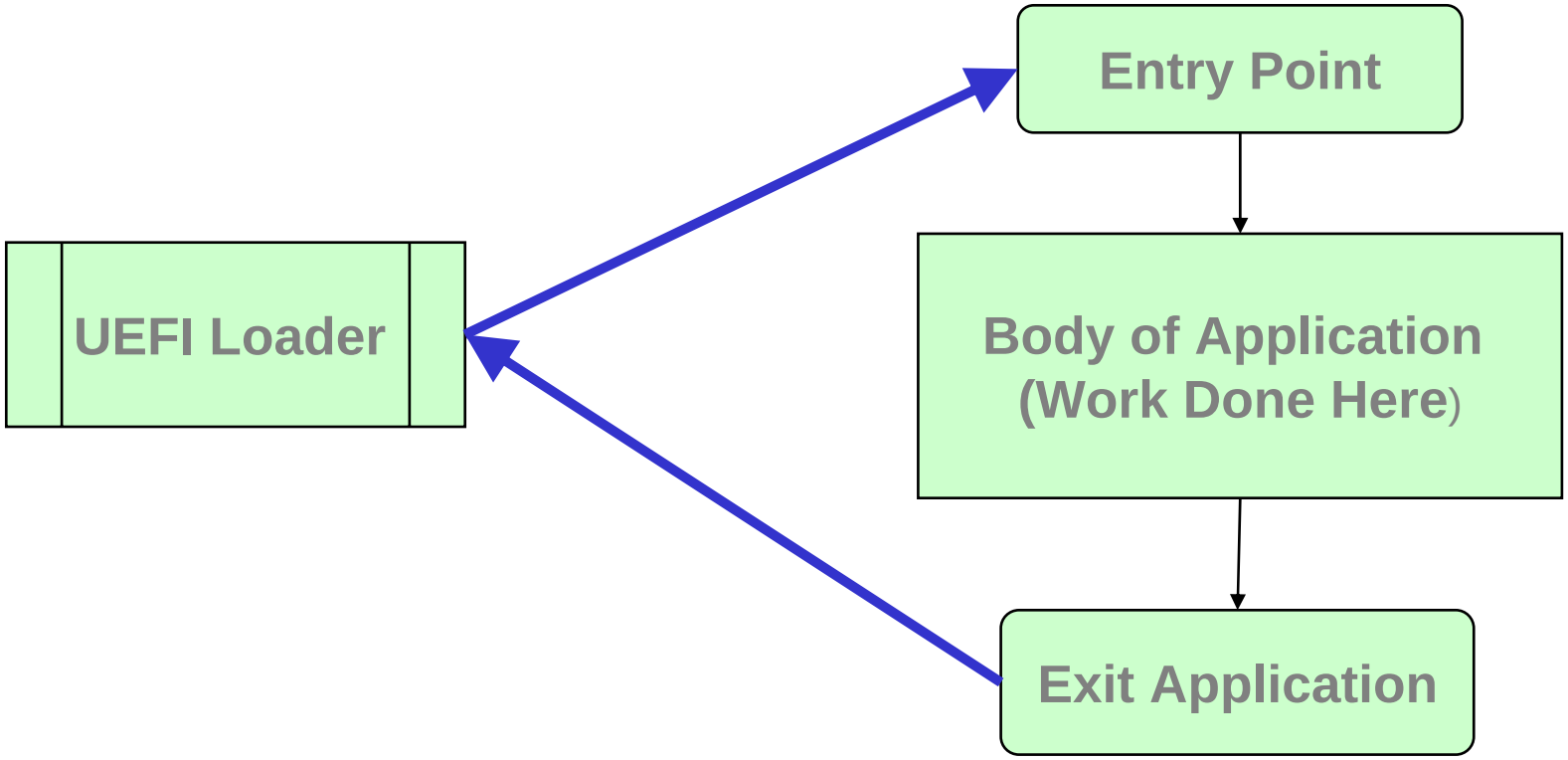
Agenda

- Introduction
- EFI Applications
- EDK DUET
- EFI Toolkit
- 3rd Party Libraries

What do UEFI Applications do?

- Extend firmware abstractly
  - Without hardware or OS dependence
- Portable across platforms
  - IA32, IA64, Intel-64, XScale, Apple*, NT32 emulator
- Enable rapid application development

Application Execution



What is an EFI Application?

- An EFI Loadable Image
  - Loaded by EFI loader just like drivers
  - Does not register protocols like drivers do
  - Consumes protocols
  - Typically user driven (exits when task completed)
  - Same set of interfaces available as drivers have
- Can be used for
  - Platform diagnostics
  - Factory diagnostics
  - Utilities
  - Driver prototyping
  - ‘Platform’ applications

Driver vs. Application

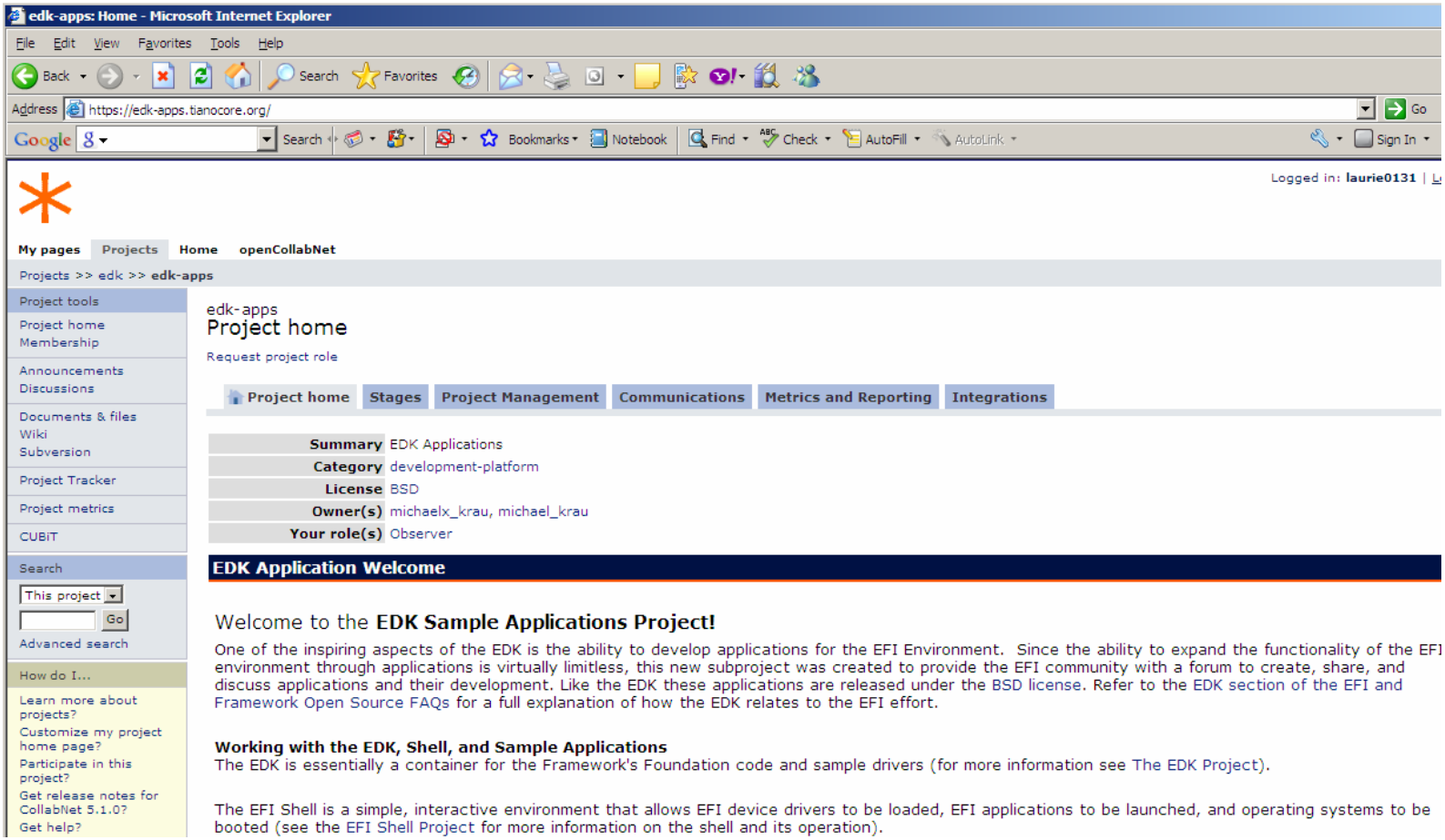
	Driver	Application
Loaded by:	EFI Loader	EFI Loader
Interfaces available:	ALL	ALL
Consume protocols?	YES	YES
Produce protocols?	YES	NO
Typically driven by?	System	User
Typical use	Support HW	Any

EFI Shell

- An EFI Application
- Interactive Console Interface
- Application Launch
- Load EFI Drivers
- Scripting Capability
- Automatic execution of startup script file
- Console redirection to files

EDK Applications

- EDK Sub-project http://edk-apps.tianocore.org



EDK-APPS Examples

- Under Documents and Files

Files

Filter this list

Name	Status	Modified by	Description
PciIoTest	Draft	michaelx_krau at 12:38:45 AM	PciIoTest will simply detect all PCI devices and print out the related information.
MemTest	Draft	michaelx_krau at 12:37:54 AM	The MemTest application just simply tests the specified memory.
HelloWorld	Draft	michaelx_krau at 12:34:12 AM	A simple EFI application, just printing the string "Hello World"
EdbCfg	Draft	yshi8 on Monda	EdbCfg is a shell application, which could change the EBC Debugger attributes in the EFI Shell environment.
EchoServer	Draft	michaelx_krau at 12:37:07 AM	The simple network application just sends back whatever received from the connected client.
EchoClient	Draft	michaelx_krau at 12:36:37 AM	A simple network application just sends a specified string to the echo server and prints the string sent back by the server to the console.
ConTest	Draft	michaelx_krau at 12:34:58 AM	A console test application, which simplely get the input char and print its char code to the screen.

What is Developers UEFI Emulation (DUET)

- DUET – UEFI Over Legacy BIOS
- Why DUET?
  - Provide IHV an EFI/UEFI environment above legacy BIOS, to help them develop and debug their native EFI/UEFI drivers.
- Enter condition:
  - Hardware Initialization done. Legacy interfaces available.
  - Legacy boot to DUET.
- Exit condition:
  - Provide pure EFI/UEFI environment. (IA32/X64)
  - Boot to EFI/UEFI Shell/OS.

Goal of DUET

- Goal is …
  - Export EFI/UEFI interface
  - Support IA32 and X64 architecture
  - Chipset/Platform independent
  - Boot from Floppy
  - Boot from USB (Legacy Free Consideration)
  - Boot from Hard Disk
  - Support boot to EFI/UEFI Shell

DUET Goal (Cont'd)

- Goal is not:
  - Not all Framework interfaces are supported, (for example: PEI-CIS, DXE-CIS)
  - Not support Itanium® Processor architecture
  - Not support CSM, INTx call (except Video), and 16bit code
  - Not support boot to Legacy OS
  - Not support boot to OS

EDK DUET

- How to use DUET
- DUET release notes on Tianocore.org:
  - https://edk.tianocore.org/files/documents/16/320/DuetRelNotes.txt

EFI Toolkit Components

- Utilities
- C Library
- Network Stack
- Platform Management
- Compression
- Database

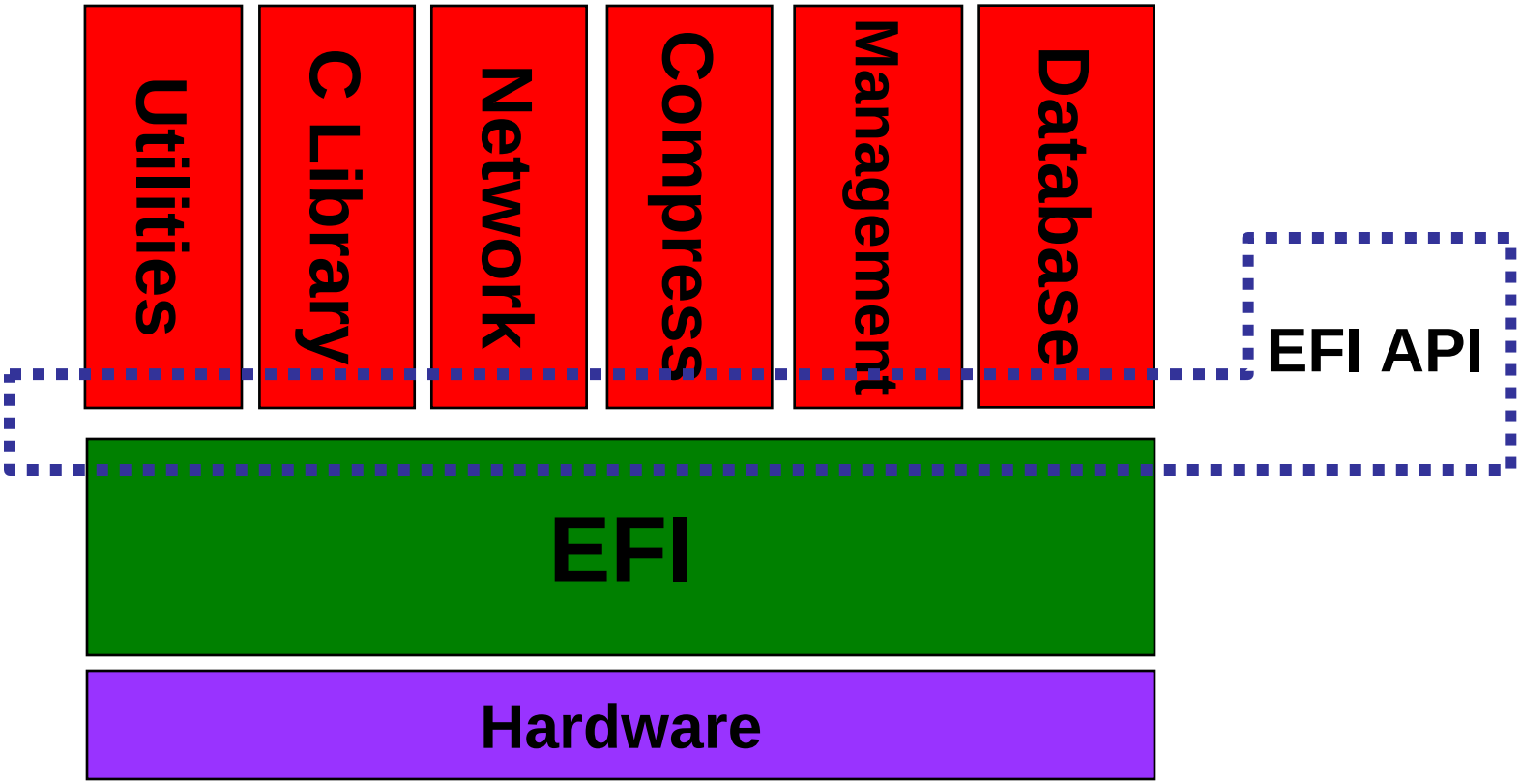
Source Included

Useful tools for EFI application development

Programming Models

- Native EFI Model
  - Uses only EFI constructs
  - Access to all EFI constructs
  - Smaller code size
- Portability Model
  - Familiar programming interfaces
  - Easier to port ANSI/POSIX based programs
  - Larger binary image
- A single program can use both

EFI Toolkit Integration



C Library

- FreeBSD Port
- ANSI/POSIX compliant
- System I/O - open(), read(), write(), close(), stat()
- Standard I/O - fopen(), printf(), gets(), …
- String/Char - strcmp(), isascii(), atoi(), …
- Memory - malloc(), free(), realloc(), …
- Time/Date - time(), asctime(), ctime(), …
- Math - sqrt(), pow(), sin(), log(), …

EFI Library

- "Lite Weight" C Library like functions
  - String Functions
  - Memory Support Functions
  - CRC Support Functions
  - Text I/O Functions
  - Math Functions
  - Spin Lock Functions
- Specific EFI functions
  - Handle and Protocol Support Functions
  - Device Path Support Functions

Network Components

- Port of FreeBSD TCP/IP stack
- Supports standard protocols
  - IPv4, ICMP, ARP, UDP, TCP
- Socket library interface
- Implemented as an EFI protocol

Miscellaneous

- SMBIOS Library
  - Library routines for parsing SMBIOS tables
- Database
  - btree
  - Hashing
- Compression
  - General purpose compression/decompression
  - Gzip functionality

Utilities

- Network utilities
  - FTP client and server, ping
- Text editor
- Scripting interpreter (Python)
- Sample applications

EFI Hello.c

```
#include "efi.h"

EFI_STATUS
InitializeHelloApplication (
   IN EFI_HANDLE          ImageHandle,
   IN EFI_SYSTEM_TABLE     *SystemTable
   )
{
   UINTN Index;

   SystemTable->ConOut->OutputString(SystemTable->ConOut,
              L”Hello application started\n");
   SystemTable->ConOut->OutputString(SystemTable->ConOut,
              L"\n\r\n\r\n\rHit any key to exit this image\n\r");
   SystemTable->BootServices->WaitForEvent(
         1, &(SystemTable->ConIn->WaitForKey), &Index);
   SystemTable->ConOut->OutputString(SystemTable->ConOut,
              L"\n\r\n\r");
   return EFI_SUCCESS;
}
```

EFI Library Hello.c

```
#include "efi.h"
#include "efilib.h"

EFI_STATUS
InitializeHelloLibApplication (
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
    )
{
    InitializeLib (ImageHandle, SystemTable);
    Print(L"\n\n\nHelloLib application started\n\n\n");
    Print(L"\nHit any key to exit this image\n");
    WaitForSingleEvent(ST->ConIn->WaitForKey,0);
    ST->ConOut->OutputString (ST->ConOut, L"\n\r\n\r");
    return EFI_SUCCESS;
}
```

C Library Hello.c

```
#include <atk_libc.h>
#include <stdio.h>

EFI_STATUS
InitializeHelloLibCApplication (
   IN EFI_HANDLE          ImageHandle,
   IN EFI_SYSTEM_TABLE    *SystemTable
   )
{
    InitializeLib(ImageHandle, SystemTable);
    printf("Hello LibC application started\n\n\n");
    printf("Hit C/R to exit this image\n");
    return( getchar() );
}
```

C Library Hello.c

```
#include <atk_libc.h>
#include <stdio.h>

int main (int argc, char **argv  )
{
    printf("Hello LibC application started\n\n\n");
    printf("Hit C/R to exit this image\n");
    return( getchar() );
}
```

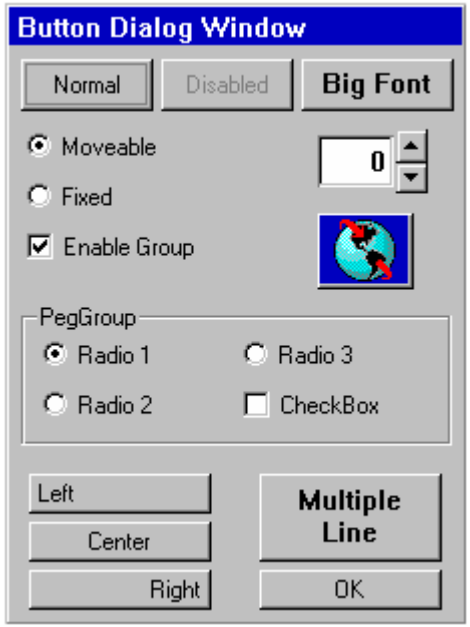
C++ Support

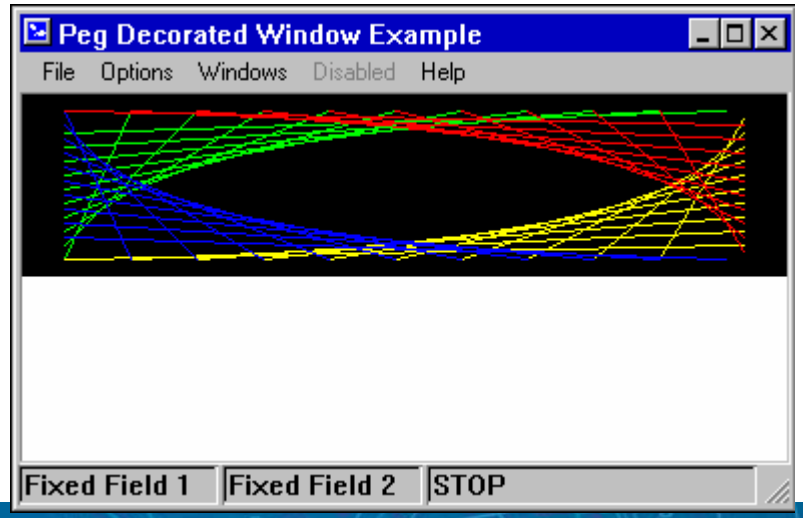
- No direct support
  - No Global constructors and destructors
- New and Delete can be mapped to malloc/free

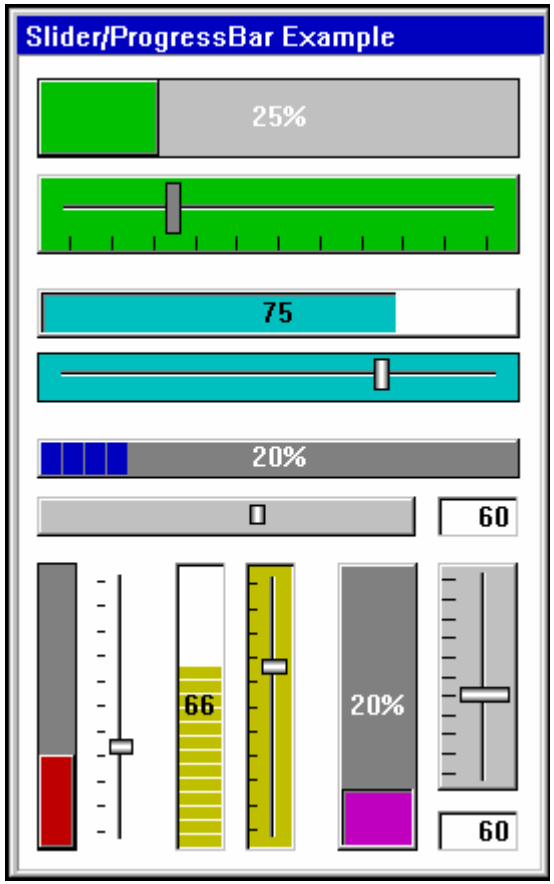
Portable Embedded Graphics

- Portable Embedded Graphics
  - Portable graphics library for EFI
  - Similar windowing components (widgets)
    - Dialog boxes
    - Progress bars, scroll bars
    - Text boxes
    - Window Management
    - Fonts
    - Bitmaps, JPEG, …
- Contact Swell Software
  - http://www.swellsoftware.com

PEG Components







Summary

- EFI Applications extend firmware
  - Provides system independence in the pre-boot space
    - Hardware
    - Operating System
    - Platform
      - Intel® IA-32, EM64T, Itanium® Architecture and XScale® technology
- Large library support
- EFI Shell provides convenient launch point

Further Information

- https://www.TianoCore.org
  - Website for EFI open source resources
    - EFI Developer Kit (EDK)
      - Nt32 emulation environment
      - EDK-APPS https:/edk-apps.tianocore.org/
    - EFI toolkit https://efi-toolkit.tianocore.org/
- http://www.swellsoftware.com
  - Portable Embedded Graphics toolkit

Q & A

Back up