UEFI Driver Development Training Events

Leon Li
UEFI Development
Intel

Agenda

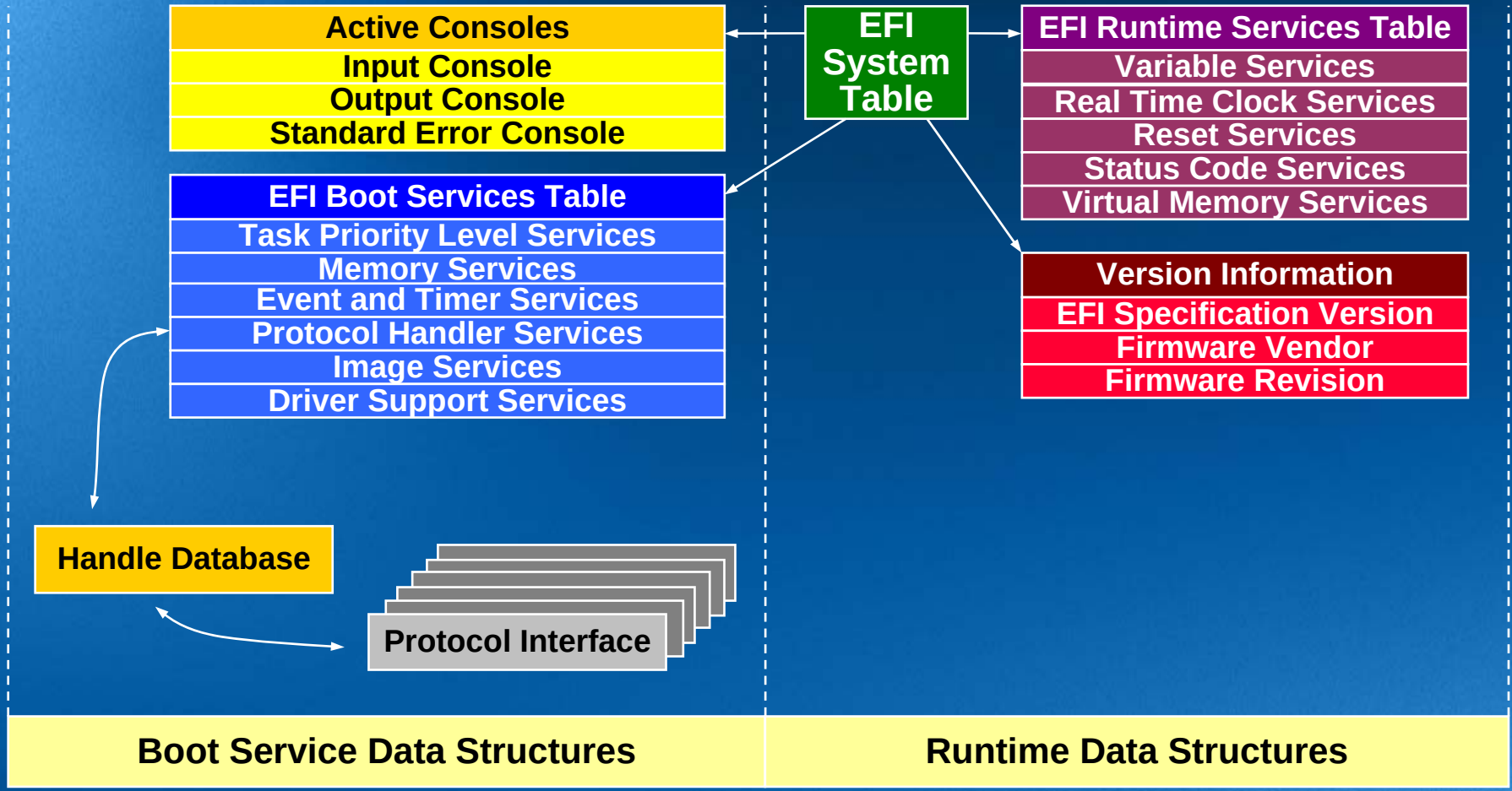
- System Boot Services
- Event Definition
- Event Types
- Event Usage
- Timer Usage
- Task Priority Level

System Boot Services?

- System services are interfaces that all UEFI compliant systems offer.
- Boot Services are a subset of these that are available only before ExitBootServices() is called.
- Runtime Services are the other subset and they are available both before and after ExitBootServices() is called.

See § 6 UEFI 2.1 Spec.

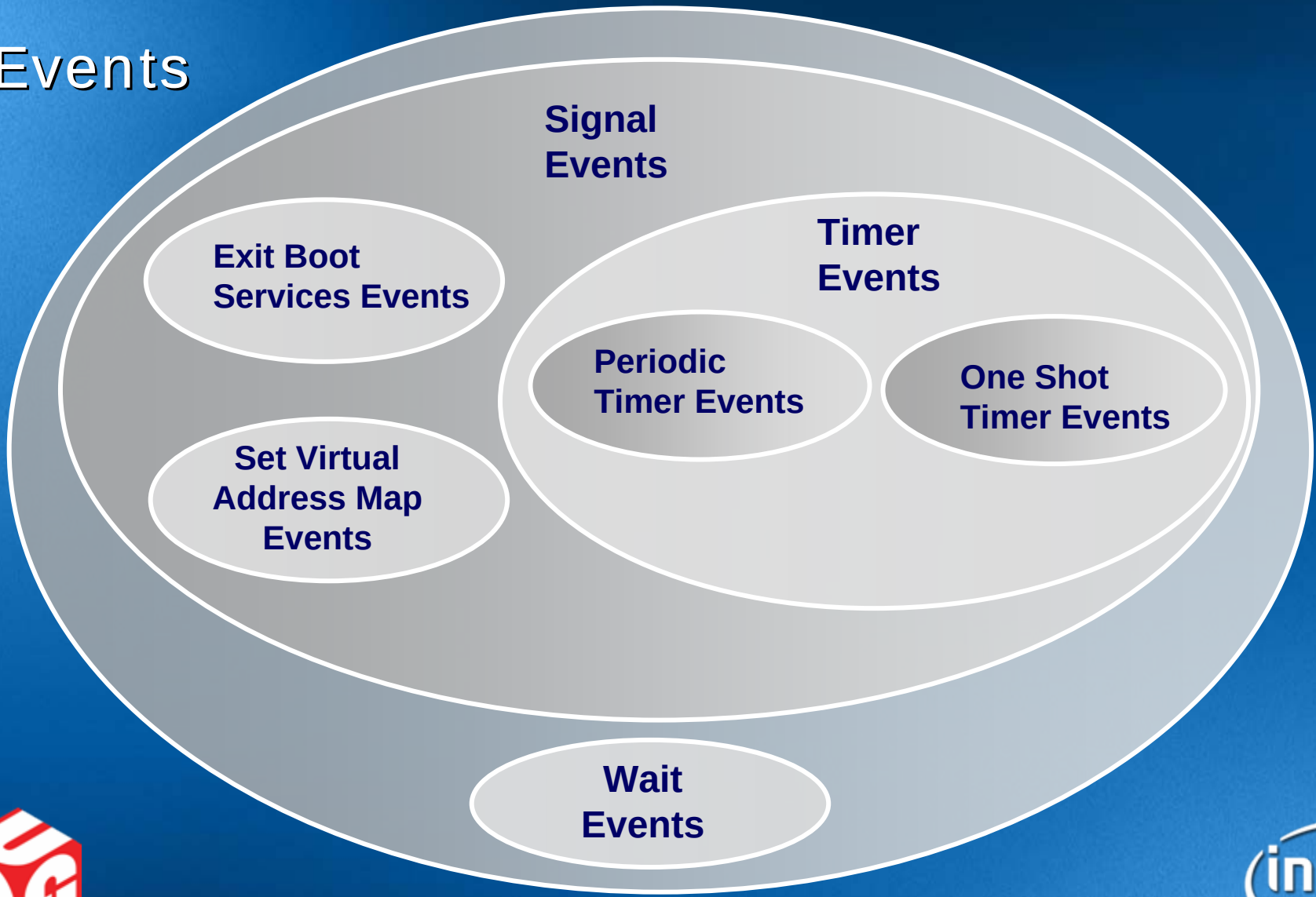
UEFI Data Structures - EFI System Table



Event Definition

- A part of Boot Services
- A messaging method
  - Returns control to a specific function
  - When the event is Signaled
    - After a specified time lapse
      - Useful for polling (i.e. Device Drivers)
    - When SignalEvent() is called with the event handle
      - Useful for controlling order of events
  - Note: special event for ExitBootServices()

Event Types and Relationships



Event Types

- Timer event – used to delay by a certain time
- Runtime event – an event that will be used after ExitBootServices()
- Notify Wait event – an event whose function is not spontaneously called
- Notify signal event – an event whose function is spontaneously called
- Exit Boot Services event – the special event that signals that ExitBootServices() has been called.

Three Elements of all Events

- **The Task Priority Level** (TPL) of the event
  - Priority at which the notification function is executed
- **A notification function**
  - Is executed when the state of the event is checked or the event is being waited upon.
  - The notification function of a signal event is executed whenever the event transitions from the waiting state to the signaled state
- **A notification context**
  - is passed into the notification function each time the notification function is executed

Event Usage

- CreateEvent() – creates an event structure
- CreateEventEx() – creates an event in a group
- CloseEvent() – closes and free event structure
- SignalEvent() – sets event to signaled state
- WaitForEvent() – stops execution until signaled
- CheckEvent() – checks the state of an event

Event Usage (continued)

- Call CreateEvent() to create an event handle
  - Or use CreateEventEx() to make a set of events
- Wait on or periodically check the state of the event
  - Or wait for the function to be called automatically
- Signal the event

	CreateEvent()	CreateEventEx()	WaitForEvent()	Calls a function
Wait Event type	Yes	Yes	Yes	No
Signal Event type	Yes	Yes	No	Yes
Set of events	No	Yes	OR	OR
Single event	Yes	No Effect	OR	OR

Event Usage (continued)

- CheckEvent() vs WaitForEvent()
  - Use CheckEvent() for a single event
  - Use WaitForEvent() for an event list

Timer Usage

1. Create an event using CreateEvent() with EVT_TIMER bitmask.
2. Set the timer using SetTimer()
   1. Use TimerCancel to cancel an existing timer
   2. Use TimerPeriodic to set a repeating timer
   3. Use TimerRelative for a single event
3. If this is a “one-shot” item close the event with CloseEvent() immediately.

Timer Usage (sample code)

```
// Create the event.
Status = gBS->CreateEvent (
     (EFI_EVENT_NOTIFY_SIGNAL | EFI_EVENT_TIMER),
     EFI_TPL_NOTIFY,
     EFI_EVENT_NOTIFY* FunctionToCall,
     NULL,
     &Event
     );
// Set off event for every ½ second.
Status = gBS->SetTimer (
     Event,
     TimerPeriodic,
     5000000
     );
// note that the time is in 100ns so 5000000 is ½ second.
```

Event/Timer Examples:

- What functions should be called in each of these cases:
  - Drivers can use events to poll for new hot-plug devices
  - Drivers can use events to wait for completion
  - Drivers can react to ExitBootServices event

Polling for new devices

- USB example
  - Creates an periodic timer event in Start()
  - Start the timer in Start()
  - Inside timer function
    - Are there any new devices on controller (or hub)
      - If the new device is a hub -> call timer function for the hub
      - Install driver for new device

Waiting for completion

- SCSI example
  - In the SCSI Passthrough protocol the PassThru() function has an (optional) event parameter.
    - If an event is passed in and the device supports non-blocking I/O operations then the call will return immediately and the event will be signaled when the I/O operation has finished.
  - Code can call the function and will be notified when the operation is complete.

Reacting to ExitBootServices event

- To get the notification:
  - Create an event in Start() with a notification function
  - React when the function is called
- When you get the notification
  - Many drivers may react to this by completing communication with devices or closing children.
  - UNDI drivers can react specially
    - They can call SetVirtualAddressMap() to switch from a boot service driver into a runtime service driver.
    - No other drivers can be runtime service drivers.

Task Priority Levels (TPLs)

- to define the priority in which notification functions are executed
  - Higher priority notifications are processed first
  - Notifications with higher priority can interrupt
- To create locks
  - Make data access in a driver atomic by temporarily raising the TPL

See §2.6 of EFI Driver Writer's Guide.

Task Priority Levels (TPLs)

- EFI supports 3 TPLs for use by drivers
  - Note: There are other levels for use by the firmware.
- In ascending order
  - TPL_APPLICATION (default TPL) – User I/O must be here
  - TPL_CALLBACK – can do file system and Disk I/O
  - TPL_NOTIFY – No blocking operations, usually low level

See §6.1 of UEFI 2.1 specification.

TPL Functions

- RaiseTPL() – used to raise the TPL
  - Note: specifying a value lower than the current TPL results in indeterminate behavior. Do Not Do This!
  - Returns the TPL as it was before the operation so the driver can restore the original value after a temporary switch.
- RestoreTPL() – used to lower the TPL
  - Note: specifying a value higher than the current TPL results in indeterminate behavior. Do Not Do This!
- No documented way to get current TPL.
- Keep track yourself

TPL Example from EHCI driver

```
EFI_TPL OldTpl;
OldTpl = gBS->RaiseTPL (EFI_TPL_NOTIFY);

<check the bus>

gBS->RestoreTPL(OldTpl);
```


intel®
Leap ahead™