Driver Module Framework (DMF)

Sam Tertzakian
Principal Software Engineer
Introduction and Agenda

Session Introduction
What is DMF and what are its goals?

Session Agenda:
Traditional driver diagram and discussion
Introduce DMFMODULE and DMF Core
DMF driver diagram and discussion
Modules discussion
How to make a DMF driver?
Look at sample source code
Plans for DMF Going Forward
DMF Resources
What is DMF and what are its goals?

Framework that makes it easier to write better drivers

“DMF Drivers” are “WDF Drivers”. Programmers use WDF and DMF together.

Goals

Make it easier and more intuitive to write modular, layered code inside drivers.
Make it possible to directly reuse (by linking) driver code without using “copy/paste/modify”.
Make it easier to properly architect drivers by eliminating improper dependencies and code paths.
Make it easier for driver writers to think using high-level constructs.
Make it easier for driver writers to create their own high-level constructs and let others reuse them.
Make driver programming easier, faster, cheaper and more satisfying.

Result when the above goals are met:

Programmers spend more time thinking, writing and debugging new code that accomplishes their specific requirements and less time writing code that has been written many times before.
Traditional Driver Diagram

WDF

WDF Callbacks

Device Context

Client Specific Code

Traditional Driver Model

BUTTON

LIST

THREAD

STREAM

ACPI

FIFO
C++ helps...but still needs “glue”

NTSTATUS MyDriver_D0Entry(...) {
    DEVICE_CONTEXT* deviceContext=DeviceContextGet(Device);

    deviceContext->Button.D0Entry(deviceContext);
    deviceContext->Thread.D0Entry(deviceContext);
    /*... some other code */
    deviceContext-> Stream.D0Entry(deviceContext);
}

HBI: Microsoft Confidential
For WinHEC 2018
Shared under NDA
11-05-18
© 2018 Microsoft
Traditional Driver Diagram
Introduction to DMF

DMFMODULE

DMF Core
DMF Core

• The DMF Core is a framework is statically linked to the Client driver.
• It resides between the Client driver and WDF.
• The Client driver gives DMF Core a list of each DMFMODULE it wants to use.
• When WDF calls into the Client driver via a callback, DMF dispatches the callback to every DMFMODULE it has created on behalf of the Client.
Goal: Make it easier to **properly architect** drivers by eliminating improper dependencies and code paths.
Goal: Make it easier to **properly architect** drivers by eliminating improper dependencies and code paths.
Other Properties of a Module (1)

Any Module can instantiate **as many instances of other Modules as Child Modules as it wants**. There is no limit to the number of descendants or siblings a Module can have (except for memory).

Programmers can use Modules and create new Modules.

When a programmer creates a new DMFMODULE, DMF automatically creates the DMFMODULE using an underlying WDFOBJECT.

A Module’s functionality is always accessed using a handle that is created by DMF and given to the Client.

Internally, the Module uses its handle to retrieve its private context. This is similar to how a driver uses its WDFDEVICE to access its device context.
Other Properties of a Module (2)

As with any object-oriented programming paradigm, Modules are agnostic about their parent which can be either the Client Driver or another Module.

DMF Modules can abstract any kind of code, from simple object like a list of buffers to an entire algorithm, data structure, programming pattern and even a full device driver.

Important: Like WDF, DMF’s interfaces to the Client driver are in C. Like WDF, DMF can also be used in C++ drivers.
DMF Driver Diagram

WDF

DMF

BUTTON

Client Specific Code

STREAM

WDF Callbacks
Module Context

WDF Callbacks
Device Context

WDF Callbacks
Module Context

WDF Callbacks
Module Context

WDF Callbacks
Module Context

WDF Callbacks
Module Context

WDF Callbacks
Module Context

Note: Modules can access all WDF Primitives

DMF Driver Model

WDF callbacks
Parent-Child Communication
Goal: Make it easier for driver writers to think using high-level constructs.
Goal: Make it easier for driver writers to think using high-level constructs.
Goal: Make it easier for driver writers to think using *high-level constructs*. 
Goal: Make it easier for driver writers to think using high-level constructs.
Modules have a Common Interface

Module’s Create Function is called by DMF

Create() function is agnostic regardless of Module type so any Module can use any other Module.

Goal: Make it easier for driver writers to create their own high-level constructs and let others reuse them.
Goal: Make it easier and more intuitive to write **modular, layered code** inside drivers.
How to make a DMF Driver?

What makes a normal WDF driver a DMF driver?

1. A WDF Client driver must call APIs to hook DMF into the Client driver:
   - DMF_DmfDeviceInitAllocate()
   - DMF_DmfDeviceInitHookPnpPowerEventCallbacks()
   - DMF_DmfDeviceInitHookFileObjectConfig()
   - DMF_DmfDeviceInitHookPowerPolicyEventCallbacks()
   - DMF_DmfDeviceInitHookQueueConfig() (optional)
   - DMF_DmfFdoSetFilter(); (optional)

2. A WDF Client driver must call this API to initialize DMF.
   - DMF_ModulesCreate()

3. The Client driver receives one additional callback where the list of Modules the driver uses is given to DMF.
How to use a Module

• You need two things:
  • The Module’s CONFIG structure.
  • The Module’s Methods.

• Every Module has a .md File
  • This file contains documentation about the purpose of the Module, its CONFIG, structures, enumerations, callbacks and Methods.
DMF Sample Code Tour

Enough theory...Let’s look at code!

There are several samples on Github. Many of the samples use the OSR USB FX2 board because it is accessible and well known.

(Detailed information about the OSR USB FX2 board is available in the MSDN device driver samples repository.)
The board has a bank of 8 switches, a bank of lights, a button and an LED segment display.

A sample function driver for the board is provided on MSDN.

The driver exposes a device interface and IOCTL codes that allow an application to control the lights on the lightbar as well as get notification that the switches have changed and what the switches are set to (on/off).

Using the application a user can read the switches and set the lights.
SwitchBar4 (Filter Driver)

- IOCTL_OSRUSBFX2_GET_INTERRUPT_MESSAGE
- IOCTL_OSRUSBFX2_READ_SWITCHES
- IOCTL_OSRUSBFX2_SET_BAR_GRAPH_DISPLAY

OSR FX2 Function Driver

SwitchBar4 (Filter Driver)
High Level Tasks for New Driver

• When D0Entry() occurs read the state of switches and set lights to match.
• Send IOCTL requests that are completed every time the switches are changed.
• Every time switches are changed, the driver should read the state, convert the bit mask and set the lights to match.
Initializing DMF in a Client Driver (1)

```c
NTSTATUS MyDriver_DeviceAdd(PDEVICEINIT DeviceInit) {
    NTSTATUS ntStatus; WDFDEVICE device; PDMFDEVICE_INIT dmfDeviceInit;
    DMF_EVENT_CALLBACKS dmfCallbacks;
    WDF_OBJECT_ATTRIBUTES objectAttributes; WDF_PNPPOWER_EVENT_CALLBACKS pnpPowerCallbacks;

    dmfDeviceInit = DMF_DmfDeviceInitAllocate(DeviceInit);

    WDF_PNPPOWER_EVENT_CALLBACKS_INIT(&pnpPowerCallbacks);
    pnpPowerCallbacks.EvtDeviceD0Entry = SwitchBarEvtDeviceD0Entry;
    DMF_DmfDeviceInitHookPnpPowerEventCallbacks(dmfDeviceInit, &pnpPowerCallbacks);
    WdfDeviceInitSetPnpPowerEventCallbacks(DeviceInit, &pnpPowerCallbacks);

    DMF_DmfDeviceInitHookFileVersionConfig(dmfDeviceInit, NULL);
    DMF_DmfDeviceInitHookPowerPolicyEventCallbacks(dmfDeviceInit, NULL);

    WdfDeviceInitSetDeviceType(DeviceInit, FILE_DEVICE_UNKNOWN);
    WdfDeviceInitSetExclusive(DeviceInit, FALSE);
    WdfFdoInitSetFilter(DeviceInit);
    DMF_DmfFdoSetFilter(dmfDeviceInit);

    WDF_OBJECT_ATTRIBUTES_INIT_CONTEXT_TYPE(&objectAttributes, DEVICE_CONTEXT);
    ntStatus = WdfDeviceCreate(DeviceInit, &objectAttributes, &device); { if (!NT_SUCCESS(ntStatus) goto Exit; }
    dmfCallbacks.EvtDmfDeviceModulesAdd = DmfDeviceModulesAdd;
    DMF_DmfDeviceInitSetEventCallbacks(dmfDeviceInit, &dmfCallbacks);
    ntStatus = DMF_ModulesCreate(device, &dmfDeviceInit);

    Exit:
    if (dmfDeviceInit != NULL) DMF_DmfDeviceInitFree(&dmfDeviceInit);
    return ntStatus;
}
```
Initializing DMF in a Client Driver (2)

```c
NTSTATUS MyDriver_DeviceAdd(PDEVICEINIT DeviceInit){
    NTSTATUS ntStatus; WDFDEVICE device; PDMFDEVICE_INIT dmfDeviceInit;
    DMF_EVENT_CALLBACKS dmfCallbacks;
    WDF_OBJECT_ATTRIBUTES objectAttributes; WDF_PNPPOWER_EVENT_CALLBACKS pnpPowerCallbacks;
    dmfDeviceInit = DMF_DmfDeviceInitAllocate(DeviceInit);
    WDF_PNPPOWER_EVENT_CALLBACKS_INIT(&pnpPowerCallbacks);
    pnpPowerCallbacks.EvtDeviceD0Entry = SwitchBarEvtDeviceD0Entry;
    DMF_DmfDeviceInitHookPnpPowerEventCallbacks(dmfDeviceInit, &pnpPowerCallbacks);
    WdfDeviceInitSetPnpPowerEventCallbacks(DeviceInit, &pnpPowerCallbacks);
    DMF_DmfDeviceInitHookFileObjectConfig(dmfDeviceInit, NULL);
    DMF_DmfDeviceInitHookPowerPolicyEventCallbacks(dmfDeviceInit, NULL);
    WdfDeviceInitSetDeviceType(DeviceInit, FILE_DEVICE_UNKNOWN);
    WdfDeviceInitSetExclusive(DeviceInit, FALSE);
    WdfFdoInitSetFilter(DeviceInit);
    DMF_DmfFdoSetFilter(dmfDeviceInit);
    WDF_OBJECT_ATTRIBUTES_INIT_CONTEXT_TYPE(&objectAttributes, DEVICE_CONTEXT);
    ntStatus = WdfDeviceCreate(DeviceInit, &objectAttributes, &device); { if (!NT_SUCCESS(ntStatus) goto Exit; }
    dmfCallbacks.EvtDmfDeviceModulesAdd = DmfDeviceModulesAdd;
    DMF_DmfDeviceInitSetEventCallbacks(dmfDeviceInit, &dmfCallbacks);
    ntStatus = DMF_ModulesCreate(device, &dmfDeviceInit);
    if (dmfDeviceInit != NULL) DMF_DmfDeviceInitFree(&dmfDeviceInit);
    return ntStatus; }
```
VOID DmfDeviceModulesAdd(WDFDEVICE Device, PDMFDEVICEINIT DmfDeviceInit) {
    DEVICE_CONTEXT* deviceContext = DeviceContextGet(Device);
    DMF_MODULE_ATTRIBUTES moduleAttributes;
    DMF_CONFIG_DefaultTarget moduleConfigDefaultTarget;

    DMF_CONFIG_DefaultTarget_AND_ATTRIBUTES_INIT(&moduleConfigDefaultTarget, &moduleAttributes);
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.BufferCountOutput = 1;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.BufferOutputSize = sizeof(SWITCH_STATE);
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestCount = 1;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.PoolTypeOutput = NonPagedPoolNx;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.PurgeAndStartTargetInD0Callbacks = FALSE;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestTargetIoctl =
        IOCTL_OSRUSBFX2_GET_INTERRUPT_MESSAGE;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.EvtContinuousRequestTargetBufferOutput =
        SwitchBarSwitchChangedCallback;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.RequestType = ContinuousRequestTarget_RequestType_Ioctl;
    moduleConfigDefaultTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestTargetMode =
        ContinuousRequestTarget_Mode_Automatic;
    moduleAttributes.PassiveLevel = TRUE;
    DMF_DmfModuleAdd(DmfModuleInit, &moduleAttributes, WDF_NO_OBJECT_ATTRIBUTES, &deviceContext->DmfModuleDefaultTarget);

    // Add more Modules as needed.
    // }
}
DMF calling the Client Driver’s D0Entry

NTSTATUS SwitchBarEvtDeviceD0Entry(WDFDEVICE Device, WDF_POWER_DEVICE_STATE PreviousState) {
    DEVICE_CONTEXT* deviceContext = DeviceContextGet(Device);
    NTSTATUS ntStatus;

    // Read the state of switches and initialize lightbar.
    //
    ntStatus = SwitchBarReadSwitchesAndUpdateLightBar(deviceContext->DmfModuleDefaultTarget);

    return ntStatus;
}
ContinuousRequestTarget_BufferDisposition
SwitchBarSwitchChangedCallback(
    _In_ DMFMODULE DmfModuleDefaultTarget,
    _In_reads_(OutputBufferSize) VOID* OutputBuffer,
    _In_ size_t OutputBufferSize,
    _In_ VOID* ClientBufferContextOutput,
    _In_ NTSTATUS CompletionStatus
) {
    ContinuousRequestTarget_BufferDisposition returnValue;
    if (!NT_SUCCESS(CompletionStatus)) {
        returnValue = ContinuousRequestTarget_BufferDisposition_ContinuousRequestTargetAndStopStreaming;
        goto Exit;
    }
    SwitchBarReadSwitchesAndUpdateLightBar(DmfModuleDefaultTarget);
    returnValue = ContinuousRequestTarget_BufferDisposition_ContinuousRequestTargetAndContinueStreaming;
    Exit:
    return returnValue;
}
NTSTATUS SwitchBarReadSwitchesAndUpdateLightBar(_In_ DMFMODULE DmfModuleDefaultTarget) {
    NTSTATUS ntStatus;
    SWITCH_STATE switchData;
    ntStatus = DMF_DefaultTarget_SendSynchronously(DmfModuleDefaultTarget, NULL, 0, (VOID*)&switchData, sizeof(SWITCH_STATE), ContinuousRequestTarget_RequestType_Ioctl, IOCTL_OSRUSBFX2_READ_SWITCHES, 0, NULL);
    if (! NT_SUCCESS(ntStatus)) goto Exit;
    ntStatus = DMF_DefaultTarget_SendSynchronously(DmfModuleDefaultTarget, &switchData.SwitchesAsUChar, sizeof(UCHAR), NULL, 0, ContinuousRequestTarget_RequestType_Ioctl, IOCTL_OSRUSBFX2_SET_BAR_GRAPH_DISPLAY, 0, NULL);

    Exit:
    return ntStatus;}

SwitchBar2 (Remote Target)

SwitchBar2
(Remote Target)

OSR FX2
Function
Driver

IOCTL_OSRUSBFX2_GET_INTERRUPT_MESSAGE
IOCTL_OSRUSBFX2_READ_SWITCHES
IOCTL_OSRUSBFX2_SET_BAR_GRAPH_DISPLAY

WinHEC 2018

For WinHEC 2018 Shared under NDA

© 2018 Microsoft
Tell DMF the List of Modules to Use

VOID DmfDeviceModulesAdd(_In_ WDFDEVICE Device, _In_ PDMFMODULE_INIT DmfModuleInit) {
    DMF_MODULE_ATTRIBUTES moduleAttributes;
    DMF_CONFIG_DeviceInterfaceTarget moduleConfigDeviceInterfaceTarget;
    DMF_MODULE_EVENT_CALLBACKS moduleEventCallbacks;

    DMF_CONFIG_DeviceInterfaceTarget_AND_ATTRIBUTES_INIT(&moduleConfigDeviceInterfaceTarget, &moduleAttributes);
    moduleConfigDeviceInterfaceTarget.DeviceInterfaceTargetGuid = GUID_DEVINTERFACE_OSRUSBFX2;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.BufferCountOutput = 1;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.BufferOutputSize = sizeof(SWITCH_STATE);
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestCount = 1;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.PurgeAndStartTargetInD0Callbacks = FALSE;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestTargetIoctl = IOCTL_OSRUSBFX2_GET_INTERRUPT_MESSAGE;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.EvtContinuousRequestTargetBufferOutput = SwitchBarSwitchChangedCallback;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.RequestType = ContinuousRequestTarget_RequestType_IOCTL;
    moduleConfigDeviceInterfaceTarget.ContinuousRequestTargetModuleConfig.ContinuousRequestTargetMode = ContinuousRequestTarget_Mode_Automatic;
    moduleAttributes.PassiveLevel = TRUE;

    DMF_MODULE_ATTRIBUTES_EVENT_CALLBACKS_INIT(&moduleAttributes, &moduleEventcallbacks);

    DMF_DmfModuleAdd(DmfModuleInit, &moduleAttributes, WDF_NO_OBJECT_ATTRIBUTES, NULL);
}
VOID
SwitchBar_OnDeviceArrivalNotification(
    _In_  DMFMODULE DmfModule
)
{
    SwitchBarReadSwitchesAndUpdateLightBar(DmfModule);
}
Other Notes about DMF

• DMF supports:
  • Function, Filter and Bus drivers.
  • Kernel-mode and User-mode drivers.
  • C and C++ drivers.
• DMF Modules always expose a C interface, but internally can use C++. Some of our User-mode drivers use COM.
• DMF developers should **always work with DEBUG build** as that is DMF’s “Verifier” which uses ASSERT() heavily.
• Microsoft Surface Team has used DMF for over three years.
• Some of our drivers have as much as **99% code reuse** (based on lines of code in Client driver). These drivers consist of 2 or 3 callback functions and the ModulesAdd() function.
• Driver with **least** amount of code reuse still reuses 69%.
• Remember: The true power of DMF is that it allows you to build your own Modules!
Plans for going forward

DMF is in Open Source

- To become a contributor, simply create a pull request on Github. Obviously, we will use great care when we update the code base.
- We have been updating the code base frequently and will continue to do so.
- We are adding more samples, especially more complex samples.
- We are adding more Modules to the Library.
- It is possible to create issues in the repository and we will respond.
- **DMF is not officially supported by Microsoft as a product. DMF is not supported by Customer Support Services.**
DMF Resources

Links

Github repository : https://github.com/Microsoft/DMF

Blog post is here : http://aka.ms/DMF

Documentation is here : https://github.com/Microsoft/DMF/tree/master/Dmf/Documentation

Email contacts : dmf-feedback@microsoft.com

Component Firmware Update : http://aka.ms/CFU

I will be available throughout WinHEC for any questions, feedback or more in depth discussion.
Summary

Goal: Make driver programming **easier, faster, cheaper** and more **satisfying**.

Goals of DMF
Spend time writing and debugging new code to **solve new problems**!

Call to Action

- Download the latest release of DMF and build it.
- Read the list of available Modules.
- Read the samples.
- Read the documentation.
- Try to use a Module in a driver.
- Try to write a Module you need and use it in a driver.
- Try to write a whole driver using Modules.
- Send us questions and feedback!