Contents

1 design pattern

1.1 OnStudioOperation : the op done by ds DRILL

- note : someone launch operation event and let it be affected

1.1.1 code

```
private void SubscribeToEvents()
{
    DsEvents.Instance().OnStudioOperation += InterruptServices_OnStudioOperation;
}

private void InterruptServices_OnStudioOperation(object sender, StudioOperationArgs ar
{
    switch (args.Operation)
    {
        case StudioSpecialOperation.HID_FINGER_MODE:
            if (args.State == StudioOperationState.STARTED)
            {
                enableDisableUIDuringHIDFingerMode(false);
            }
            else if (args.State == StudioOperationState.FINISHED)
            {
                enableDisableUIDuringHIDFingerMode(true);
            }
            break;
    }
}
```

1.2 if it's called from main stream to update the ui DRILL

- note :

1.2.1 code

```
private void enableDisableUIDuringHIDFingerMode(bool enable)
{
    if (this.InvokeRequired)
    {
        BeginInvoke(new Action<bool>(enableDisableUIDuringHIDFingerMode), enable);
        return;
    }

    //todo: port when need
    //this.splitContainer1.Panel2Collapsed = !enable;
}
```

1.3 get notice from OperationEvent to change the UI status in dashboard DRILL

- note :

1.3.1 code

```
private void OnStudioOperation(object sender, StudioOperationArgs args)
{
    if (args.Operation == StudioSpecialOperation.FLASH)
    {
        if (args.State == StudioOperationState.STARTED)
        {
                EnableMe(false);
            }
            else if (args.State == StudioOperationState.FINISHED)
            {
                EnableMe(true);
            }
            else if (args.State == StudioOperationState.ERROR)
            {
                EnableMe(true);
            }
        }
        else if (args.Operation == StudioSpecialOperation.TUNING)
        {
            if ((args.Category == StudioOperationCategory.PENTUNING) && (args.State == Stu
            {
                EnableMe(false);
            }
            else if ((args.Category == StudioOperationCategory.PENTUNING) && (args.State =
            {
                EnableMe(true);
            }
        }
        else if (args.Operation == StudioSpecialOperation.PREPRODUCTION)
        {
            if (args.State == StudioOperationState.STARTED)
            {
                EnableMe(false);
            }
            else if (args.State == StudioOperationState.FINISHED)
            {
                EnableMe(true);
            }
        }
    }
```

1.4 UIEventService **DRILL**

- note :

the para of delegate function must be

1. object sender
2. ToolEventArgs e

1.4.1 code

```
public delegate void ToolValueChangedEvent(ToolEventArgs e);
public delegate void ToolClickEvent(ToolClickEventArgs e);
public delegate void RegisterValueChangedEvent(object sender, RegisterType type);

public delegate void ReadAllRegistersNeededEvent(object sender);
public delegate void DoNotReportNextFwResetEvent(object sender);

public delegate void SolutionFileSelectionChangedEvent(object sender);
public delegate void ActivityWindowSelectionChangedEvent(object sender, bool hasSelect
public delegate void StatusTextUpdateEvent(string msg, Color msgClr);
public delegate void ProgressBarEvent(bool start);
```

1.5 delegate event function DRILL

- note :

1.5.1 code

```
public delegate void ToolValueChangedEvent(ToolEventArgs e);
public static event ToolValueChangedEvent OnToolValueChanged;

public static void FireToolValueChanged(ToolEventArgs e)
{
    if (OnToolValueChanged != null)
    {
        OnToolValueChanged(e);
    }
}


// usage
// studiomain.cs
UIEventsService.FireToolValueChanged(e);
```

```
// what things it would do?
// see what function is +to the delegate function
UIEventsService.ToolValueChangedEvent += xxx
```

1.6 method invoker DRILL

- note :

1.6.1 code

```
public static event MethodInvoker OnSolutionOpen;

public static event MethodInvoker OnSolutionClosing;

public static event MethodInvoker OnSolutionClosed;
```

1.7 StudioEventService DRILL

- note :

1.7.1 code

```
public static IAsyncResult FireConnectionModeChanged(object sender, StatusEnum state)
{
    ConnectionModeEvent connectionModeDelegate = FireSyncConnectionModeChanged;
    return connectionModeDelegate.BeginInvoke(sender, state, null, null);
}
```

1.8 get InvocationList DRILL

- note :

1.8.1 code

```
public static void FireSyncConnectionModeChanged(object sender, StatusEnum state)
{
    L12n.UseInvariantCulture();

    ConnectionModeEvent handler = OnConnectionModeChanged;
    if (handler != null)
    {
        foreach (ConnectionModeEvent f in handler.GetInvocationList())
        {
            try
            {
                f(sender, state);
            }
            catch (Exception ex)
            {
                DsMessage.FireOnMessage(StudioPackageType.Biz, StudioMessageType.Error
                    "FireConnectionModeEvent" + ":" + ex.Message);
                Logger.WriteError("FireConnectionModeEvent" + ": " + ex.Message + " : "
            }
        }
    }
}
```

1.9 get resources from resource manager DRILL

- note :

1.9.1 code

```
_resMgr = new ResMan("Synaptics.DSNG.UI.Res", Assembly.GetExecutingAssembly());
```

1.10 get/set F11Touchpad in solutionFacade DRILL

- note :

1.10.1 code

```
// dp : fly weight
        [XmlElement("Touchpads", typeof(F11TouchpadItem))]
        public List<F11TouchpadItem> Touchpads { get; set; }

         public F11TouchpadItem GetTouchpad()
         {
             DesignLayout layout = StudioSolutionManager.Instance.Experiment.Layout;
             if (layout.Touchpads.Count() == 0)
```

```
            {
                layout.Touchpads.Add(new F11TouchpadItem());
            }
            return layout.Touchpads[0];
        }

        public F11TouchpadItem GetTouchpad(string name)
        {
            DesignLayout layout = StudioSolutionManager.Instance.Experiment.Layout;
            F11TouchpadItem result = layout.Touchpads.FirstOrDefault(item => item.Ite
            if (result == null)
            {
                result = new F11TouchpadItem { ItemName = name };
                layout.Touchpads.Add(result);
            }
            return result;
        }

        public void SetTouchpad(F11TouchpadItem item)
        {
            DesignLayout layout = StudioSolutionManager.Instance.Experiment.Layout;
            layout.Touchpads.RemoveAll(x => x.ItemName == item.ItemName);
            layout.Touchpads.Add(item);
        }
```

1.11 gethashcode template **DRILL**

- note :

1.11.1 code

```
public override int GetHashCode()
{
    unchecked
    {
        int result = base.GetHashCode();
        result = (result * 397) ^ (XSensorMap != null ? CollectionIdentity.GetHashCode
        result = (result * 397) ^ (YSensorMap != null ? CollectionIdentity.GetHashCode
        result = (result * 397) ^ (XSensitivities != null ? CollectionIdentity.GetHash
        result = (result * 397) ^ (YSensitivities != null ? CollectionIdentity.GetHash
        result = (result * 397) ^ VisualFeedback.GetHashCode();
        result = (result * 397) ^ Orientation.GetHashCode();
        result = (result * 397) ^ Origin.GetHashCode();
        result = (result * 397) ^ FingerDimension.GetHashCode();
        result = (result * 397) ^ XScalingFactor.GetHashCode();
        result = (result * 397) ^ YScalingFactor.GetHashCode();
        result = (result * 397) ^ TouchPadScale.GetHashCode();
        result = (result * 397) ^ ActivePenInkingWidth.GetHashCode();
        return result;
    }
}
```

1.12 restore layout from xml file DRILL

- note :

1.12.1 code

```
public void RestoreLayout()
{
    try
    {
        if (File.Exists(Application.StartupPath + "\\" + "DashboardLayout.xml"))
        {
            using (var reader = new StreamReader(Application.StartupPath + "\\" + "Das
            {
                var gp = (GroupProperty)new XmlSerializer(typeof(GroupProperty)).Deser
                this.ultraExplorerBarDashboard.NavigationMaxGroupHeaders = gp.MaxGroup
                foreach (GroupValue gv in gp.GroupSetting)
                {
                    int grpId = this.ultraExplorerBarDashboard.Groups.IndexOf(gv.Name)
                    if (0 <= grpId)
                    {
                        UltraExplorerBarGroup g =
                            this.ultraExplorerBarDashboard.Groups[gv.Name];
                        if (g != null)
                        {
                            g.Visible = gv.Visible;
                    }
                }
            }
            IComparer myComparer = new myReverserClass(gp.GroupSetting);
            this.ultraExplorerBarDashboard.Groups.Sort(myComparer);
        }
    }
}
catch (Exception ex)
{
    DsMessage.FireOnMessage(StudioPackageType.Ui, StudioMessageType.Error, "Dashbo
}
}
```

1.13 save layout **DRILL**

- note :

1.13.1 code

```
public void SaveLayout()
{
    try
    {
        using (StreamWriter writer = new StreamWriter(Application.StartupPath + "\\" +
        {
            GroupProperty gp = new GroupProperty();
            gp.MaxGroupHeaderValue = this.ultraExplorerBarDashboard.NavigationMaxGroup
            foreach (UltraExplorerBarGroup group in this.ultraExplorerBarDashboard.Gro
            {
                gp.GroupSetting.Add(new GroupValue(group.Key, group.Index, group.Visib
            }
            new XmlSerializer(typeof(GroupProperty)).Serialize(writer, gp);
        }
    }
    catch
    {
        DsMessage.FireOnMessage(StudioPackageType.Ui, StudioMessageType.Error,
            "Can't save the Dashboard layout.");
    }
}
```

1.14 list firstOrDefault DRILL

- note :

1.14.1 code

```
xIndex = _GroupValueList.FirstOrDefault(g => g.Name == ((UltraExplorerBarGroup)x).Key)
yIndex = _GroupValueList.FirstOrDefault(g => g.Name == ((UltraExplorerBarGroup)y).Key)
```

1.15 serrialize DRILL

- note :

1.15.1 code

```
[Serializable]
public class GroupValue
{
    [XmlElement("Name")]
    public string Name;
    [XmlElement("Index")]
    public int Index;
    [XmlElement("Visible")]
    public bool Visible;

    public GroupValue()
    {
    }

    public GroupValue(string name, int index, bool visible)
    {
        Name = name;
        Index = index;
        Visible = visible;
    }
}
```

```
[Serializable]
[XmlInclude(typeof(GroupValue))]
public class GroupProperty
{
    private List<GroupValue> _groupsetting = new List<GroupValue>();
    public int MaxGroupHeaderValue { get; set; }

    [XmlElement("GroupSetting")]
    public List<GroupValue> GroupSetting
    {
        get { return _groupsetting; }
        set { _groupsetting = value; }
    }
}
```

1.16 sort csv by list **DRILL**

- note :

1.16.1 code

```
private string SortCsv(string csv)
{
    List<string> items = new List<string>();

    string[] strTokens = csv.Split(new char[] { ',' });
    foreach (string item in strTokens)
    {
        items.Add(item);
    }
    items.Sort();

    string output = string.Empty;
    foreach (string item in items)
    {
        output += item + ",";
    }
    if (output != string.Empty)
    {
        // Remove the trailing ","
        output = output.Substring(0, output.Length - 1);
    }
    return output;
}
```

1.17 get flash identify value in dashboard DRILL

- note :

1.17.1 code

```
ReflashParameters reflashParam = new ReflashParameters(
    SolutionDataFacade.Instance.GetDeviceProtocol(),
    SolutionDataFacade.Instance.GetHostInfo());

reflashParam.MPCSerialNumber = serial;

touchModuleInfo tInfo = null;

Dictionary<ReflashInfoBase.InfoKey, string> touchModInfo =
    FWMamager.Instance().FirmwareInfo(reflashParam);
```

1.18 throw exception in func DRILL

- note :

1.18.1 code

```
if (serialNumber == "")
    throw new Exception("Can't find Serial Number of MPC");
else
    throw new Exception(string.Format("Can't find MPC of serial number {0}", serialNum
```

1.19 singleton DRILL

- note :

1.19.1 code

```
public static DashBoardController GetInstance()
{
    if (theCtrler == null)
    {
        theCtrler = new DashBoardController();
    }
    return theCtrler;
}

private static DashBoardController theCtrler = null;
```

1.20 read interrupt status **DRILL**

- note :

1.20.1 code

```
public ulong CommandReadInterruptStatus()
{
    try
    {
        var f01Helper = RMIFunctionFacade.Instance._helpersFromDevi.GetF01Helper();
        if (f01Helper != null)
        {
            return f01Helper.GetInterrupt(true);
        }
    }
    catch (DsException ex)
    {
        DsMessage.FireOnMessage(StudioPackageType.Ui, StudioMessageType.Error, "Dashbo
    }
    return ulong.MaxValue;
}
```

1.21 mvc, controller get data from RMIFacade **DRILL**

- note :

1.21.1 code

```
public byte GetSleepMode()
{
    return RMIFunctionFacade.Instance.GetSleepMode();
}
```

1.22 mvc, view : get notice from observer and get value from controller DRILL

- note :

1.22.1 code

```
private void UpdateControlRegisterValues()
```

1.23 launch one UIservice to notify other UI DRILL

- note :

1.23.1 code

```
private void ButtonForceUpdateClick(object sender, EventArgs e)
{
    buttonForceUpdate.Enabled = false;
    _Controller.SetForceUpdate();
    buttonForceUpdate.Enabled = true;
    // Bring back input focus, otherwise, the input focus go to the next control in th
    buttonForceUpdate.Focus();
    UIEventsService.FireExecuteDashboardCommandDuringDiagnostic();
}
```

1.24 another way to read resiter by _deviceHelper DRILL

- note :

1.24.1 code

```
_DeviceHelper.Read(0x1, RegisterTypeEnum.Data, 0, GetData(0x1));
return _F01.GetInterrupt(false);
```

1.25 RunTimeState : get status update from OnStudioOperation DRILL

- note :

1.25.1 code

```
private void OnStudioOperation(object sender, StudioOperationArgs args)
```

1.26 RMIFunctionFacade, helper from solution.instance.registermap DRILL

- note :

1.26.1 code

```
_helpersFromSoln = new HelpersFromSolution();
_f01Hlpr = new RMIFunction01Helper(StudioSolutionManager.Instance.RegisterMap);
```

1.27 RMIFunctionFacade from device DRILL

- note :

1.27.1 code

```
_helpersFromDevi = new HelpersFromDevice();
_f21Helper = new RMIFunction21Helper();
```

1.28 fire Studio Operation example DRILL

- note :

1.28.1 code

```
DsEvents.Instance().FireStudioOperation(this, new StudioOperationArgs(StudioSpecialOpe
```

1.29 Ilist example DRILL

- note :

1.29.1 code

```
public IList<RegisterInfo> FindRegistersByAddr(IList<ushort> addrList)
{
    Debug.Assert(addrList != null);
    RegisterMap map = GetCurrentRegisterMap();
    if (map == null)
    {
        return null;
    }
    RegisterMapHelper helper = new RegisterMapHelper(map);
    return addrList.Select(addr => helper.FindByAddress(addr)).ToList();
}

public IList<RegisterInfo> GetAllControlRegisters()
{
    return _RegisterMap.AllRegisters.Where(reg => reg.Type == RegisterTypeEnum.Control
}
```

1.30 Ilist example : sortby DRILL

- note :

1.30.1 code

```
List<RegisterInfo> regs = map.AllRegisters.Where(x => x.Type == RegisterTypeEnum.Contr
```

1.31 list find DRILL

- note :

1.31.1 code

```
public SolutionFileInfo Find(string name)
{
    return All.Find(e => e.Name == name);
}

public SolutionFileInfo FindFileName(string name)
{

    return All.Find(e => (e.LinkFileInfo != null) && (e.LinkFileInfo.Name == name));
}
```

1.32 ctor **DRILL**

- note :

1.32.1 code

```
public StudioSolution(StudioProject project)
{
    Project = project;
    Files = new SolutionFiles();
    FwReqDataChanged = false;
}

public StudioSolution() : this(null)
{
}
```

1.33 copy ctor : copy one instance into this **DRILL**

- note :

1.33.1 code

```
/// <summary>
/// Copy constructor
/// </summary>
/// <param name="src"></param>
public RegisterMap(RegisterMap src)
    : this(src.Name)
{
    Copy(src);
}

private void Copy(RegisterMap src)
{
    foreach (RegisterInfo registerInfo in src.AllRegisters)
    {
        PacketRegisterInfo packet = registerInfo as PacketRegisterInfo;
        if (packet != null)
        {
            AllRegisters.Add(new PacketRegisterInfo(packet));
        }
        else
        {
            AllRegisters.Add(new RegisterInfo(registerInfo));
        }
    }
    InterruptMasks.Clear();
    foreach (KeyValuePair<byte, ulong> interruptMask in src.InterruptMasks)
    {
        InterruptMasks.Add(interruptMask.Key, interruptMask.Value);
    }
    FunctionRevisions.Clear();
    foreach (KeyValuePair<byte, byte> functionRevision in src.FunctionRevisions)
    {
        FunctionRevisions.Add(functionRevision.Key, functionRevision.Value);
    }
    IsFunctionPublicFlags.Clear();
    foreach (KeyValuePair<byte, bool> pubflag in src.IsFunctionPublicFlags)
    {
        IsFunctionPublicFlags.Add(pubflag.Key, pubflag.Value);
    }
}
```

1.34 Clone for RegisterMap DRILL

- note :

1.34.1 code

```
/// <summary>
/// Default constructor
/// </summary>
public RegisterMap()
    : this("Master")
{
}

public RegisterMap Clone()
{
    return new RegisterMap(this);
}
```

1.35 list where and select **DRILL**

- note :

1.35.1 code

```
/// <summary>
/// All packet registers
/// </summary>
[XmlIgnore]
public List<PacketRegisterInfo> PacketRegisters
{
    get { return AllRegisters.Where(x => x is PacketRegisterInfo).Select(x => (PacketR
}
```

1.36 KeyValuePair **DRILL**

- note :

1.36.1 code

```
foreach (KeyValuePair<byte, ulong> inter in interrupts)
{
    InterruptMasks.Add(inter.Key, inter.Value);
}
foreach (KeyValuePair<byte, byte> revision in functionRevision)
{
    FunctionRevisions.Add(revision.Key, revision.Value);
}
foreach (KeyValuePair<byte, bool> isPubFlag in functionPublicFlags)
{
    IsFunctionPublicFlags.Add(isPubFlag.Key, isPubFlag.Value);
}
```

1.37 get Enumerator DRILL

- note :

1.37.1 code

```
var e = registersInfo.Values.GetEnumerator();
e.MoveNext();
return e.Current;
```

1.38 enumerator DRILL

- note :

1.38.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.39 dispose example DRILL

- note :

1.39.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.40 InterruptService : how to get int from dev DRILL

- note :

1.40.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.41 get data from device.instance DRILL

- note :

1.41.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.42 read interrupt by _deviceHelper DRILL

- note :

1.42.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.43 open external log DRILL

- note :

1.43.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.44 instance lock DRILL

- note :
- `https://dotblogs.com.tw/yc421206/2011/01/07/20624`

1.44.1 code

```
var eNumber = registerCollection.Values.GetEnumerator();
eNumber.MoveNext();
var eSubNumber = eNumber.Current.GetEnumerator();
eSubNumber.MoveNext();
return AllRegisters[eSubNumber.Current.Value];
```

1.45 method invoker DRILL

- note :

1.45.1 code

```
internal void ChangeStartStop()
{
    MethodInvoker f;
    if (IsRunning)
    {
        f = Stop;
    }
    else
    {
        PreProcessForStart();
        f = () => Start(false);
    }
    f.BeginInvoke(null, null);
}
```