Purpose: This is to document the understanding of FX3 DMA Override mode and its notifications in DMA Callback.

Executive Summary:
This explains operating the DMA Channel in Override mode and notifications.

Configuring the DMA Channel in DMA Override mode:

In general, we create a DMA channel between producer and consumer socket and allocate some buffers to it.

```
    dmaCfg.size = 1024;
    dmaCfg.count = CY_FX_BULKLP_DMA_BUF_COUNT; /* Count is 8*/
    dmaCfg.prodSckId = CY_FX_EP_PRODUCER_SOCKET; /* UIB Producer */
    dmaCfg.consSckId = CY_FX_EP_CONSUMER_SOCKET; / UIB Consumer */
    dmaCfg.dmaMode = CY_U3P_DMA_MODE_BYTE;
    dmaCfg.notification =  CY_U3P_DMA_CB_RECV_CPLT | CY_U3P_DMA_CB_SEND_CPLT
| CY_U3P_DMA_CB_PROD_EVENT | CY_U3P_DMA_CB_CONS_EVENT;
    dmaCfg.cb = CyFxApplnDmaCb;
    dmaCfg.prodHeader = 0;
    dmaCfg.prodFooter = 0;
    dmaCfg.consHeader = 0;
    dmaCfg.prodAvailCount = 0;

    apiRetStatus = CyU3PDmaChannelCreate (&glChHandleBulkLp,
            CY_U3P_DMA_TYPE_AUTO, &dmaCfg);
    if (apiRetStatus != CY_U3P_SUCCESS)
    {
        CyU3PDebugPrint (4, "CyU3PDmaChannelCreate failed, Error code =
%d\n", apiRetStatus);
        CyFxAppErrorHandler(apiRetStatus);
    }
```

When we operate the DMA Channel in DMA override mode, there is no necessary to allocate buffers to it. We can allocate 0-byte size buffers to it.

```
    dmaCfg.size = 0;/* No need to be zero */
    dmaCfg.count = CY_FX_BULKLP_DMA_BUF_COUNT; /* Count is 8*/
    dmaCfg.prodSckId = CY_FX_EP_PRODUCER_SOCKET; /* UIB Producer */
    dmaCfg.consSckId = CY_FX_EP_CONSUMER_SOCKET; / UIB Consumer */
    dmaCfg.dmaMode = CY_U3P_DMA_MODE_BYTE;
    dmaCfg.notification =  CY_U3P_DMA_CB_RECV_CPLT | CY_U3P_DMA_CB_SEND_CPLT
| CY_U3P_DMA_CB_PROD_EVENT | CY_U3P_DMA_CB_CONS_EVENT;
    dmaCfg.cb = CyFxApplnDmaCb;
    dmaCfg.prodHeader = 0;
    dmaCfg.prodFooter = 0;
    dmaCfg.consHeader = 0;
    dmaCfg.prodAvailCount = 0;

    apiRetStatus = CyU3PDmaChannelCreate (&glChHandleBulkLp,
            CY_U3P_DMA_TYPE_AUTO, &dmaCfg);
    if (apiRetStatus != CY_U3P_SUCCESS)
    {
```

```
        CyU3PDebugPrint (4, "CyU3PDmaChannelCreate failed, Error code =
%d\n", apiRetStatus);
        CyFxAppErrorHandler(apiRetStatus);
    }
```

In the above two cases, we have created DMA Channel. This is said as DMA is in Configured State. The sockets corresponding to a DMA channel are left disabled when the channel is created, so that transfers are started only when desired by the user.

`CyU3PDmaChannelSetXfer` function enables a DMA channel to transfer a specified amount of data before suspending again. An infinite transfer can be started by specifying a transfer size of 0.

If you want to operate the DMA in override mode, you shouldn't call `CyU3PDmaChannelSetXfer` API. Instead you should call `CyU3PDmaChannelSetupRecvBuffer` or `CyU3PDmaChannelSetupSend` API.

Case A:

When the host sends any data over the USB bulk out endpoint, we need to call `CyU3PDmaChannelSetupRecvBuffer` API to receive it in override mode. The device receives the data in a user provided ***Receive buffer***. We can process it and decides the further action.

If the host wants to send another data packet to the device, we need to call `CyU3PDmaChannelSetupRecvBuffer` API again to operate in override mode. Otherwise, the channel is left disabled.

Case B:

In another case, device wants to send any data packet to the host, we need to call `CyU3PDmaChannelSetupSendBuffer` API to send it in override mode. The host will receive the data from user provided ***Send buffer***.

If we do not want DMA override mode further, we can call `CyU3PDmaChannelSetXfer` API in above two cases (A and B), then the data transfer happens over DMA Mode, if we allocate buffer for the DMA Channel.

Note that we can operate any type of DMA Channel in DMA Override mode by using the `CyU3PDmaChannelSetupRecvBuffer and CyU3PDmaChannelSetupSendBuffer`

Explained by Example:

In case of MSC example, we call the `CyU3PDmaChannelSetupRecvBuffer` API in *`CY_FX_MSC_STATE_CBW state,`* and decodes the CBW and decides whether there is a data phase.

If there is a *`CY_FX_MSC_STATE_DATA`*, we call `CyU3PDmaChannelSetXfer` by passing the DMA Channel handle and transfer size as per Read or Write data requests.

In *CY_FX_MSC_STATE_CSW state,* we call **CyU3PDmaChannelSetupSendBuffer** to send the status to host.

You can see that the DMA is operating in ***Override*** modes in *CY_FX_MSC_STATE_CBW* and *CY_FX_MSC_STATE_CSW* states. It is operating in DMA normal mode in *CY_FX_MSC_STATE_DATA* state.

Notifications of DMA Override mode:

To receive the notifications, we need to register a DMACallback with desired notifications.

We can only receive the following notifications in DMA override mode:

1. *CY_U3P_DMA_CB_RECV_CPLT* – When the user provided buffer receives the data from producer (**CyU3PDmaChannelSetupRecvBuffer**)
2. *CY_U3P_DMA_CB_SEND_CPLT* – when the user provided buffer sends the data to consumer (**CyU3PDmaChannelSetupSendBuffer**)
3. *CY_U3P_DMA_CB_ABORTED* – This event is generated when the Abort API is invoked
4. *CY_U3P_DMA_CB_ERROR* – This event is generated when the hardware detects an error

---------- END OF MEMO ----------