Instruction Manual

Drag-and-Drop Buttons Demo Files

Development Tools

Last Revised: 6/24/2015

AMX Limited Warranty and Disclaimer

This Limited Warranty and Disclaimer extends only to products purchased directly from AMX or an AMX Authorized Partner which include AMX Dealers, Distributors, VIP's or other AMX authorized entity.

AMX warrants its products to be free of defects in material and workmanship under normal use for three (3) years from the date of purchase, with the following exceptions:

- Electroluminescent and LCD Control Panels are warranted for three (3) years, except for the display and touch overlay components are warranted for a period of one (1) year.
- Disk drive mechanisms, pan/tilt heads, power supplies, and MX Series products are warranted for a period of one (1) year.
- AMX lighting products are guaranteed to switch on and off any load that is properly connected to our lighting products, as long as the AMX lighting products are under warranty. AMX also guarantees the control of dimmable loads that are properly connected to our lighting products. The dimming performance or quality there of is not guaranteed, impart due to the random combinations of dimmers, lamps and ballasts or transformers.
- AMX software is warranted for a period of ninety (90) days.
- Batteries and incandescent lamps are not covered under the warranty.
- AMX AutoPatch Epica, Modula, Modula Series4, Modula CatPro Series and 8Y-3000 product models will be free of defects in materials and manufacture at the time of sale and will remain in good working order for a period of three (3) years following the date of the original sales invoice from AMX. The three-year warranty period will be extended to the life of the product (Limited Lifetime Warranty) if the warranty card is filled out by the dealer and/or end user and returned to AMX so that AMX receives it within thirty (30) days of the installation of equipment but no later than six (6) months from original AMX sales invoice date. The life of the product extends until five (5) years after AMX ceases manufacturing the product model. The Limited Lifetime Warranty applies to products in their original installation only. If a product is moved to a different installation, the Limited Lifetime Warranty will no longer apply, and the product warranty will instead be the three (3) year Limited Warranty.

All products returned to AMX require a Return Material Authorization (RMA) number. The RMA number is obtained from the AMX RMA Department. The RMA number must be clearly marked on the outside of each box. The RMA is valid for a 30-day period. After the 30-day period the RMA will be cancelled. Any shipments received not consistent with the RMA, or after the RMA is cancelled, will be refused. AMX is not responsible for products returned without a valid RMA number.

AMX is not liable for any damages caused by its products or for the failure of its products to perform. This includes any lost profits, lost savings, incidental damages, or consequential damages. AMX is not liable for any claim made by a third party or by an AMX Authorized Partner for a third party.

This Limited Warranty does not apply to (a) any AMX product that has been modified, altered or repaired by an unauthorized agent or improperly transported, stored, installed, used, or maintained; (b) damage caused by acts of nature, including flood, erosion, or earthquake; (c) damage caused by a sustained low or high voltage situation or by a low or high voltage disturbance, including brownouts, sags, spikes, or power outages; or (d) damage caused by war, vandalism, theft, depletion, or obsolescence.

This limitation of liability applies whether damages are sought, or a claim is made, under this warranty or as a tort claim (including negligence and strict product liability), a contract claim, or any other claim. This limitation of liability cannot be waived or amended by any person. This limitation of liability will be effective even if AMX or an authorized representative of AMX has been advised of the possibility of any such damages. This limitation of liability, however, will not apply to claims for personal injury.

Some states do not allow a limitation of how long an implied warranty last. Some states do not allow the limitation or exclusion of incidental or consequential damages for consumer products. In such states, the limitation or exclusion of the Limited Warranty may not apply. This Limited Warranty gives the owner specific legal rights. The owner may also have other rights that vary from state to state. The owner is advised to consult applicable state laws for full determination of rights.

EXCEPT AS EXPRESSLY SET FORTH IN THIS WARRANTY, AMX MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. AMX EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED IN THIS LIMITED WARRANTY. ANY IMPLIED WARRANTIES THAT MAY BE IMPOSED BY LAW ARE LIMITED TO THE TERMS OF THIS LIMITED WARRANTY. EXCEPT AS OTHERWISE LIMITED BY APPLICABLE LAW, AMX RESERVES THE RIGHT TO MODIFY OR DISCONTINUE DESIGNS, SPECIFICATIONS, WARRANTIES, PRICES, AND POLICIES WITHOUT NOTICE.

Table of Contents

Drag-and-Drop Buttons 1
Overview 1
AMX System Requirements for Drag and Drop 1
Draggable Buttons and Drop Target Buttons 1
Using Draggable Buttons (on the Touch Panel) 1
Drag/Drop Type Button (General) Property 2
Drop Groups 2
Example - Grouping By Connection Type 2
Drop Group Button (General) Property 3
Drop Groups - Notes 4
Drag and Drop-Specific Events 4
Events for Draggable Buttons 4
Events for Drop Target Buttons 4
Custom Event Parameters for Drag and Drop Events 4
DragEvent Parameters 5
DropEvent Parameters 5
^BDC (Button Drag and Drop Custom Event Command) 6
Syntax 6
Variables 6
Events 6
DragDrop.axi 9
Basic Demo - No Drop Groups 15
Overview 15
Before You Begin 15
1) Create a TPDesign5 Project/Import Images 16
2) Create & Configure a Drop Target Button 16
Create a Drop Target Button 16
Set Drop Target Button Properties - General 17
Set Drop Target Button Properties - Programming 17
Set Drop Target Button Properties - States 17
3) Create & Configure Draggable Buttons 18
Create Four Draggable Buttons 18
Set Draggable Button Properties - General 19
Set Draggable Button Properties - Programming 19
Set Draggable Button Properties - States 20
4) Create and Configure a "CLEAR VTC SOURCE" Button 21
Create a "CLEAR VTC SOURCE" Button 21
Set "CLEAR VTC SOURCE" Button Properties - General 21
Set "CLEAR VTC SOURCE" Button Properties - Programming 21
5) Write NetLinx Code To Respond To Custom Event 22

6) Use NetLinx Studio 4 to Compile and Transfer the Project Files 24
End Result 25
Advanced Demo - Three Drop Groups 27
Overview 27
Before You Begin 27
1) Create a TPDesign5 Project/Import Images 28
2) Create & Configure Drop Target Buttons 28
Create Three Drop Target Buttons 28
Set Drop Target Button Properties - General 29
Set Drop Target Button Properties - Programming 29
Set Drop Target Button Properties - States 29
Add States to each Drop Target Button 30
Add a "Target-Valid" or "Target-Invalid" Icon to each State of each Drop Target Button 30
Set Drop Target Button Properties - Events 33
Configure the "Drop Enter" Event for All Drop Target Buttons 34
Configure the "Drop Exit" Event for All Drop Target Buttons 34
Configure the "Drop" Event for All Drop Target Buttons 35
Add Each Drop Target Button to a Drop Group 35
Add the LEFT DISPLAY and CENTER DISPLAY Drop Target Buttons To "group_1" 36
Add the CENTER DISPLAY Drop Target Button To "group_2" 36
Add the RIGHT DISPLAY Drop Target Button To "group_3" 36
3) Create Drop Groups 37
4) Create & Configure Draggable Buttons 38
Create Five Draggable Buttons 38
Set Draggable Button Properties - General 39
Associate Draggable Buttons With a Drop Group 39
Set Draggable Button Properties - Programming 40
Set Draggable Button Properties - States 41
Set Draggable Button Properties - Events 42
Configure the "Drag Start" Event for Draggable Buttons 42
Configure the "Drag Cancel" Event for Draggable Buttons 44
5) Add a "SMALL/LARGE ICONS" Button 45
Create a "SMALL/LARGE ICONS" Button 45
Set "SMALL/LARGE ICONS" Button Properties - Programming 46
Set "SMALL/LARGE ICONS" Button Properties - States 46
6) Add a "CLEAR DISPLAY SOURCE" Button 47
Create a "CLEAR DISPLAY SOURCE" Button 47
Set "CLEAR DISPLAY SOURCE" Button Properties - General 47
Set "CLEAR DISPLAY SOURCE" Button Properties - Programming 47
7) Write NetLinx Code To Respond To Custom Event 48
8) Use NetLinx Studio 4 to Compile and Transfer the Project Files 51
End Result 52

Drag-and-Drop Buttons

Overview

G5 Panels and TPDesign5 support "drag-and-drop" functionality for General and Multi-State General buttons. This function allows the end-user to initiate a drag on a button with a "long press", then drag and release (or "drop") the button onto a drop target (FIG. 1):

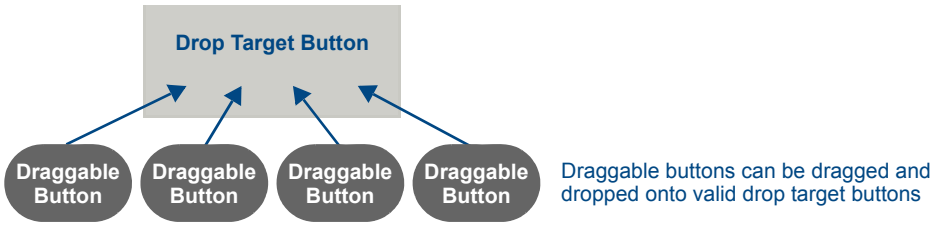


FIG. 1 Draggable buttons and Drop Target button

AMX System Requirements for Drag and Drop

The following software, hardware and firmware requirements must be met to support Drag and Drop functionality:

- TPDesign5 - version 1.3 (or higher)
- X Series G5 Touch Panels - panel firmware v1.3.23 (or higher)
- NetLinx Masters - master firmware v1.3.17 (or higher)

Draggable Buttons and Drop Target Buttons

To use the drag-and-drop function, the TPDesign5 project must include at least one "draggable" button, and at least one "drop target" button. *General* and *Multi-State General* Buttons (only) can be set as either a *Draggable* or as a *Drop Target* button, via the Drag/Drop Type (General) button property.

- "**Draggable**" buttons are buttons that can be long-pressed and dragged onto a drop target button.
- "**Drop Target**" buttons are buttons that serve as potential targets for draggable buttons.

Using Draggable Buttons (on the Touch Panel)

To use draggable buttons on a G5 touch panel (FIG. 2):

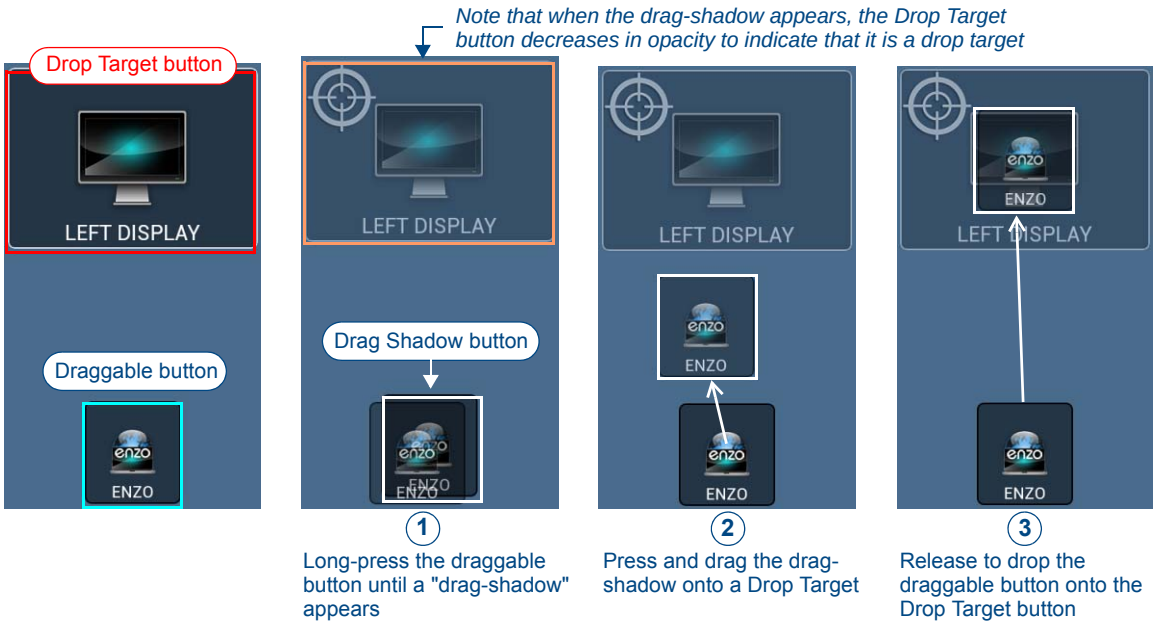


FIG. 2 Using Draggable Buttons on the Touch Panel

1. Long-press (*press and hold for 1 second*) the draggable button (1).
2. In approximately 1 second, a transparent copy of the button appears on the screen (2).
3. Drag the button onto a valid Drop Target button, and release to "drop" the draggable button (3).

NOTE

As shown in FIG. 2, when the drag shadow appears, the target will decrease opacity to indicate it is a drop target. See page 37 for details on details on States properties for Drop Target buttons.

Drag/Drop Type Button (General) Property

A new General property called "*Drag/Drop Type*" is available in TPDesign5 that sets the selected General or Multi-State General button as either "draggable" or as a "drop target" . By default, this property is set to "none" (FIG. 3):

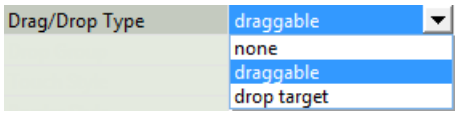


FIG. 3 General Property - Drag/Drop Type

General Property - Drag/Drop Type	
none:	The selected button is neither draggable or a drop target (default setting).
draggable:	With draggable selected, the user can drag the button on the touchpanel.
drop target:	When drop target is selected, the button acts as a target for a draggable button to be dropped on.

Drop Groups

Drop groups provide a means of a validity check for drop targets - they allow you to control which drop target buttons will serve as valid targets for draggable buttons.

Drop Groups are assigned to draggable buttons, and determine which Drop Target buttons are considered to be valid targets for each draggable button. Once a draggable button has a Drop Group assigned to it, only those drop targets that exist within the assigned Drop Group are valid targets. Conversely, draggable buttons are not allowed to be dragged and dropped onto an invalid drop target.

NOTE

While Drop Groups are not a requirement for drag and drop functionality, they provide a powerful method of limiting drag and drop functionality to ensure an optimal user experience.

Example - Grouping By Connection Type

FIG. 4 on page 3 provides an example of three Drop Groups being used to organize the source (input) devices that can be dragged onto each of four Video Output devices:

- Each Video Output device is represented by a *Drop Target* button - each one supports a different set of inputs.
- Each Source (Input) device is represented by a *Draggable* button - each one provides a different type of source.

This example indicates three Drop Groups:

- **Drop Group A**: This group accepts all source inputs that provide HDMI input. Note that Drop Group A includes Video Output devices 1, 2 and 3, as all of these devices support HDMI.
- **Drop Group B**: This group accepts all source inputs that provide Component input. Note that Drop Group B includes Video Output devices 2 and 3, since both devices support Component.
- **Drop Group C**: This group accepts all source inputs that provide 4K input. Note that Drop Group C includes only Video Output device 3, since it is the only one that supports 4K.
- Note that Video Output Device 4 has no Drop Group assignment. Therefore, the only source inputs allowed to be dragged and dropped on this Drop Target are those that also have no Drop Group assignment.

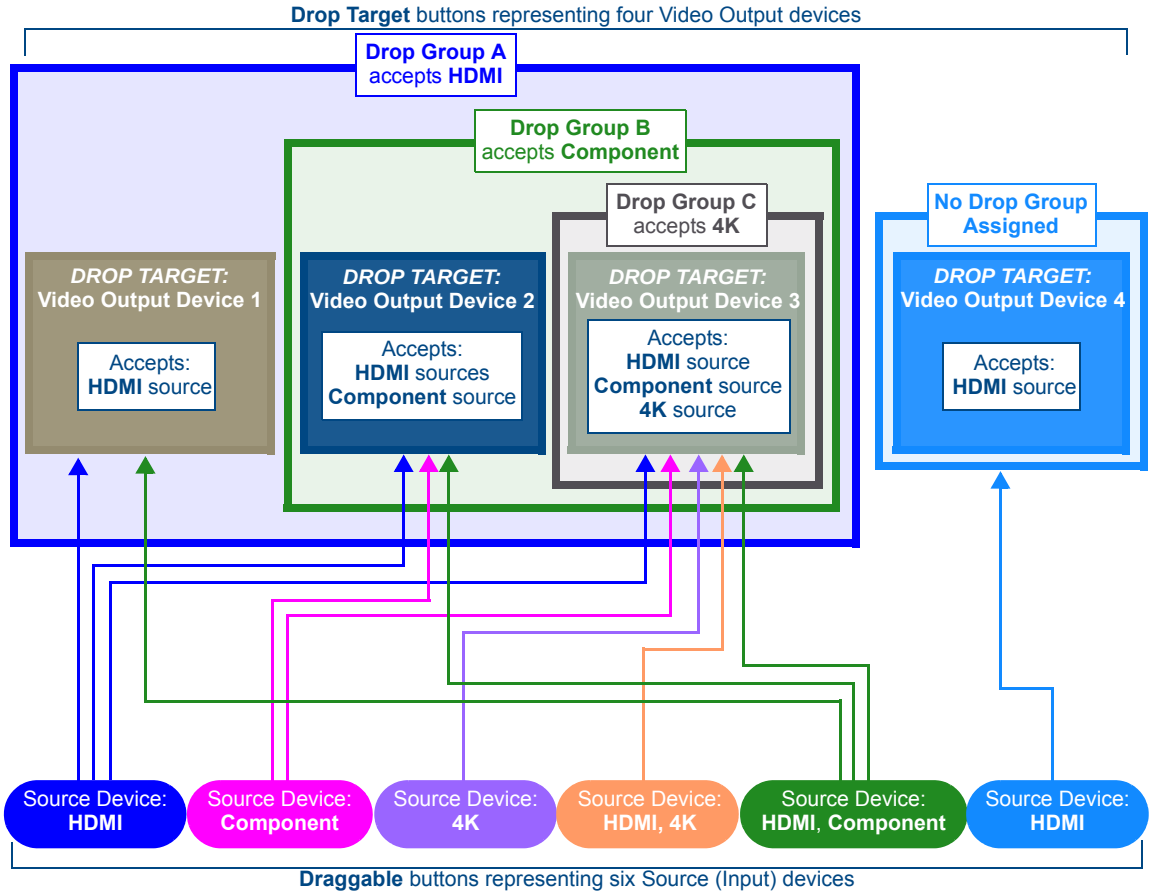


FIG. 4 Using Drop Groups to Control targets for Draggable Buttons

In this example, Drop Groups prevent Input devices from being dragged and dropped onto incompatible video output devices. With the configuration indicated in FIG. 4, HDMI sources are only allowed to be dragged and dropped onto Video Output devices that support HDMI. Likewise, 4K sources are not allowed to be dragged and dropped onto Video Output devices that support do not support 4K input.

NOTE

Use multiple states on drop target buttons to display a specific bitmap on the drop target button based on Drop Group assignments. For example, when a draggable button is dragged onto a valid drop target, a bitmap can be displayed on the drop target button to indicate that it is a valid target for the selected button. Conversely, a different image can be used to indicate that the drop target is invalid for the selected button.

Drop Group Button (General) Property

A new General property called "*Drop Group*" is available in TPDesign5 (v1.3 or higher) that associates the selected Draggable button with a specific Drop Group (FIG. 5):

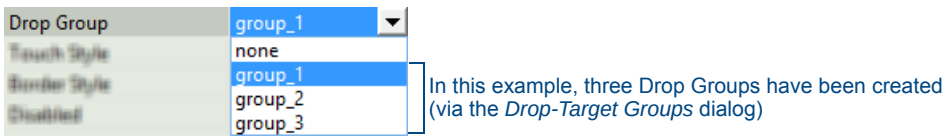


FIG. 5 General Property - Drop Group

General Property - Drop Group	
none:	The selected button is not associated with a Drop Group
Drop Groups:	Select the Drop Group to which the selected draggable button will be associated. Drop Groups are created in the *Drop-Target Groups* dialog (see page 37).

Note that this property is only available for General and Multi-State General buttons that have been set as *Draggable* via the *Drag/Drop Type* (General) property (see page 2).

Drop Groups - Notes

- Drop Group names are case-insensitive.
- Only drop targets can be grouped.
- Drop targets can exist in multiple Drop Groups.
- A draggable button can only have 1 Drop Group assigned to it.
- If no group is assigned to a draggable button, then only drop targets that are not assigned to Drop Groups are valid targets. Note that this is the default (and simplest) use case: it allows designers to quickly create a page with draggables that can be dropped on any drop target with no additional configuration required.

Drag and Drop-Specific Events

TPDesign5 (1.3.23 or higher) supports a set of new Events for Draggable and Drop Target buttons:

Events for Draggable Buttons

Draggable buttons support two drag-specific Events (FIG. 6):

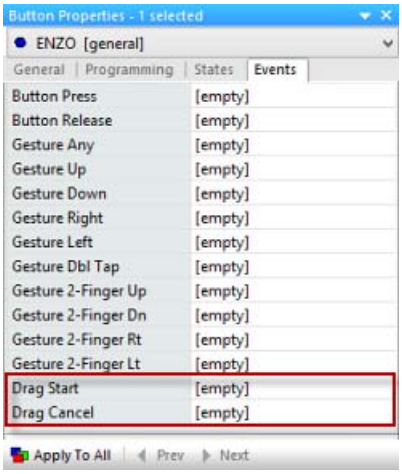


FIG. 6 Drag-Specific Events

- **Drag Start**: The event will occur when the specified draggable button has initiated a drag. Drag starts are initiated by a long press on a draggable button.
- **Drag Cancel**: The event will occur when the specified draggable button has been dropped outside of a valid drop target.

Events for Drop Target Buttons

Drop Target buttons support three drop-specific Events (FIG. 7):

- **Drop Enter**: The event will occur when a draggable button has entered a valid drop target.
- **Drop Exit**: The event will occur when a draggable button has exited a valid drop target.
- **Drop**: The event will occur when a draggable button has been dropped onto a valid drop target.

Custom Event Parameters for Drag and Drop Events

The events are:

- ActionDragStarted - a draggable button has initiated a drag
- ActionDragCancel - a draggable button has been dropped outside of a valid target
- ActionDropEntered - a draggable button has entered a valid target
- ActionDrop Exited - a draggable button has exited a valid target
- ActionDrop - a draggable button has been dropped on a valid target

Also, the Drag/Drop events provide predefined variables that are populated when action event occurs. These values can be used in the custom event definition:

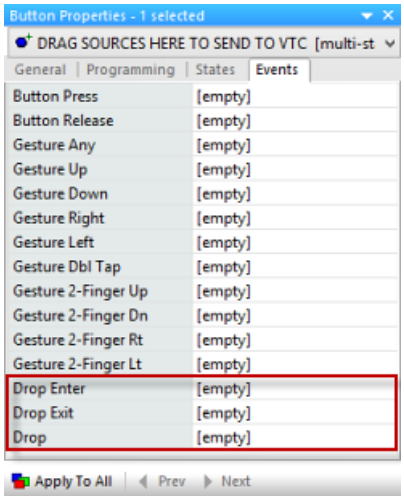


FIG. 7 Drop-Specific Events

DragEvent Parameters

- ${dragChannelPort}
- ${dragChannelCode}
- ${dragAddressPort}
- ${dragAddressCode}
- ${dragGroupName}
- ${dragButtonName}
- ${dragPageName}
- ${dragInfo}
- ${dropTargetsValid}
- ${dropTargetsInvalid}

DropEvent Parameters

- ${dropChannelPort}
- ${dropChannelCode}
- ${dropAddressPort}
- ${dropAddressCode}
- ${dragChannelPort}
- ${dragChannelCode}
- ${dragAddressPort}
- ${dragAddressCode}
- ${dragGroupName}
- ${dragButtonName}
- ${dragPageName}
- ${dragInfo}
- ${dropTargetsValid}
- ${dropTargetsInvalid}

^BDC (Button Drag and Drop Custom Event Command)

This command configures Drag and Drop custom events. This command can be used to enable or disable the transmission of custom events to the master whenever certain operations occur. For example, the system programmer may want to be notified whenever a drag button enters an acceptable target.

When using the ^BDC command, it is not necessary to assign button specific event actions. These can be empty if the ^BDC command is used.
If these are defined, the action generated does not have to conform to the custom event definition as set in the ^BDC command.
If the ^BDC command events are enabled, and button specific actions (i.e. custom event action) as both defined, then both will be sent when an event occurs.

The notification mechanism is a custom event. The ^BDC command takes the form of a comma separated list of custom event numbers. If the number is 0 or blank for a given event type then no custom event will be transmitted when that event occurs. If a number is specified, then it is used as the EVENTID value for the custom event.

The range of 32001 to 65535 has been reserved in the panel for user custom event numbers. A different value could be used but might collide with other AMX event numbers. Event configuration is not permanent and all event numbers revert to the defaults when the panel restarts.

By default the ^BDC command is *enabled*, and the default values are:

- DragStartedEvent = 1410
- DropEnterEvent = 1411
- DropExitEvent = 1412
- DropEvent = 1413
- DragCancelEvent = 1414

To *disable* the ^BDC command send: ^BDC-0,0,0,0,0

Syntax

```
"'^BDC-[optional DragStarted event num], [optional DropEntered event num],
  [optional DropExited event num], [optional Drop event num], [optional DragCancel event num]'"
```

Variables

- DragStarted Event Number = 0 for no event or a value from 32001 to 65535.
- DropEntered Event Number = 0 for no event or a value from 32001 to 65535.
- DropEntered Event Number = 0 for no event or a value from 32001 to 65535.
- Drop Event Number = 0 for no event or a value from 32001 to 65535.
- DragCancel Event Number = 0 for no event or a value from 32001 to 65535.

Events

- DragStarted - a draggable button has initiated a drag
- DropEntered - a draggable button has entered a valid target
- DropExited - a draggable button has exited a valid target
- Drop - a draggable button has been dropped on a valid target
- DragCancel - a draggable button has been dropped outside of a valid target

In response to any or all of the above events, the panel will create a custom event which is then sent to the master.

The format of **START** custom events transmitted to the master are as follows:

Format - START custom events	
CUSTOM.TYPE	the specified drag event custom event type (started)
CUSTOM.ID	the address of the viewer button which generated the event
CUSTOM.FLAG	0
CUSTOM.VALUE1	the button address of the draggable
CUSTOM.VALUE2	0
CUSTOM.VALUE3	0
CUSTOM.TEXT	'dr{ch=<channelPort>,<channel>:ad=<addressPort>,<address>:gp=<groupName>:nm=<buttonName>} dt{vl=<dropTargetValid 1=valid,0=invalid>:ch=<channelPort>,<channel>:ad=<addressPort>,<address>:nm=<buttonName>}... dt{vl=<dropTargetValid 1=valid,0=invalid>:ch=<channelPort>,<channel>:ad=<addressPort>,<address>:nm=<buttonName>}'

The CUSTOM.TEXT provides data sets that represent the draggable's info (dr). The draggable's info included is the drag channel port, the drag channel code, the drag address port, the drag address code, the drag group name, and the drag button name. Drag target info is also presented, with a data set for each drag target visible at that time. The drag targets info (dt) includes the target validity to accept the drop, the drop target channel port, the drop target channel code, the drop target address port, the drop target address code, and the drop target button name.

- Buttons are identified as dr (draggable) or dt (drop target)
- Button properties are contained between open brace ({) and close brace (})
- Button properties are represented by key=value pairs (KVP).
- Keys are two letters followed by equal (=) by convention but the two letter keys are not a requirement.
- Property KVPs are separated by colon (:).
- **Each Button's data sets are on a separate line (i.e. the close brace is followed by a \n).**

Key values	
• dr	draggable
• ch	channel (port,channel)
• ad	address (port,address)
• gp	group name
• nm	button name
• dt	drop target
• vl	validity of drop target (valid=1, invalid=0)
• ch	channel (port,channel)
• ad	address (port,address)
• nm	button name

Example texts:

```
• dr{ch=1,31:ad=1,31:gp=:nm=Drag1}
• dt{vl=1:ch=1,101:ad=1,101:nm=Tgt1}
• dt{vl=1:ch=3,103:ad=3,103:nm=Tgt3}
• dt{vl=1:ch=3,103:ad=3,103:nm=Tgt3}
• dt{vl=0:ch=1,11:ad=1,11:nm=Grp1 Tgt1}
• dt{vl=0:ch=1,12:ad=1,12:nm=Grp1 Tgt2}
• dt{vl=0:ch=2,11:ad=2,11:nm=Grp2 Tgt1}
• dt{vl=0:ch=1,15:ad=1,15:nm=Grp1 Tgt5}
• dt{vl=0:ch=1,16:ad=1,16:nm=Grp1 Tgt6}
• dt{vl=0:ch=2,13:ad=2,13:nm=Grp2 Tgt3}
• dt{vl=0:ch=1,15:ad=1,15:nm=Grp1 Tgt5}
• dt{vl=0:ch=1,16:ad=1,16:nm=Grp1 Tgt6}
• dt{vl=0:ch=2,13:ad=2,13:nm=Grp2 Tgt3}
• dr{ch=2,4:ad=2,4:gp=Group1+2:nm=Drag2_4}
• dt{vl=1:ch=1,11:ad=1,11:nm=Grp1 Tgt1}
• dt{vl=1:ch=1,12:ad=1,12:nm=Grp1 Tgt2}
• dt{vl=1:ch=2,11:ad=2,11:nm=Grp2 Tgt1}
• dt{vl=1:ch=1,15:ad=1,15:nm=Grp1 Tgt5}
• dt{vl=1:ch=1,16:ad=1,16:nm=Grp1 Tgt6}
• dt{vl=1:ch=2,13:ad=2,13:nm=Grp2 Tgt3}
• dt{vl=1:ch=1,15:ad=1,15:nm=Grp1 Tgt5}
• dt{vl=1:ch=1,16:ad=1,16:nm=Grp1 Tgt6}
• dt{vl=1:ch=2,13:ad=2,13:nm=Grp2 Tgt3}
• dt{vl=0:ch=1,101:ad=1,101:nm=Tgt1}
• dt{vl=0:ch=3,103:ad=3,103:nm=Tgt3}
• dt{vl=0:ch=3,103:ad=3,103:nm=Tgt3}
```

A NetLinx .AXI file that can provide routines to parse the drag and drop info strings can be found on page 9.

The format of **ENTER/EXIT/CANCEL** custom events transmitted to the master are as follows:

Format - ENTER/EXIT/CANCEL custom events	
CUSTOM.TYPE	the specified drag event (started/entered/exited/drop/cancel) the address of the viewer button which generated the event
CUSTOM.ID	the address of the viewer button which generated the event
CUSTOM.FLAG	0
CUSTOM.VALUE1	the button address of the draggable
CUSTOM.VALUE2	0
CUSTOM.VALUE3	0
CUSTOM.TEXT	""

The format of the DROP custom event transmitted to the master is as follows:

Format - DROP custom event	
CUSTOM.TYPE	the specified drag event (started/entered/exited/drop/cancel) the address of the viewer button which generated the event
CUSTOM.ID	the address of the viewer button which generated the event
CUSTOM.FLAG	0
CUSTOM.VALUE1	the button address of the draggable
CUSTOM.VALUE2	the button address of the dropTarget
CUSTOM.VALUE3	0
CUSTOM.TEXT	group name to which the dropTarget belongs

Example:

```
SEND_COMMAND panel,"'^BDC-32001,32002,32003,32004,32005'"
```

After the users sends this command to the panel, if the user then drags a button addressed 9 and then proceeds to drop that draggable button on a dropTarget button addressed 10, the following event would be transmitted to the master.

```
CUSTOM.TYPE = 32004 (the drop event)
CUSTOM.ID = 10 (the drop target creating the event)
CUSTOM.FLAG = 0
CUSTOM.VALUE1 = 9 (the button we dragged over the target & dropped)
CUSTOM.VALUE2 = 10 (the dropTarget that the draggable was dropped on)
CUSTOM.VALUE3 = 0
CUSTOM.TEXT = "" (a name we had given to the group the target was assigned,
                  since the target was not assigned to a group we'll receive an
                  empty string)
```

DragDrop.axi

The NetLinx .AXI file below provides routines to parse the drag and drop info strings:

```
PROGRAM_NAME='DragDrop'

(***********************************************************)
(*          DEVICE NUMBER DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_DEVICE

(***********************************************************)
(*               CONSTANT DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_CONSTANT

#IF_NOT_DEFINED __DRAG_DROP_MAX_TARGETS__
#DEFINE __DRAG_DROP_MAX_TARGETS__ '__DRAG_DROP_MAX_TARGETS=100'
INTEGER __DRAG_DROP_MAX_TARGETS = 100;
#END_IF

#IF_NOT_DEFINED __DRAG_DROP_NUM_PANELS__
#DEFINE __DRAG_DROP_NUM_PANELS__ '__DRAG_DROP_NUM_PANELS=1'
INTEGER __DRAG_DROP_NUM_PANELS = 1;
#END_IF
(***********************************************************)
(*              DATA TYPE DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_TYPE
STRUCTURE __DRAG_DROP_sDragObject
{
   INTEGER chanPort;
   INTEGER chan;
   INTEGER addrPort;
   INTEGER addr;
   char    groupName[100];
   char    buttonName[100];
}

STRUCTURE __DRAG_DROP_sDropTargetObject
{
    INTEGER valid;
    INTEGER chanPort;
    INTEGER chan;
    INTEGER addrPort;
    INTEGER addr;
    char    buttonName[100];
}

(***********************************************************)
(*               VARIABLE DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_VARIABLE

VOLATILE __DRAG_DROP_sDragObject __DRAG_DROP_current_drag[__DRAG_DROP_NUM_PANELS];
VOLATILE __DRAG_DROP_sDropTargetObject __DRAG_DROP_current_targets[__DRAG_DROP_NUM_PANELS]
                                                                  [__DRAG_DROP_MAX_TARGETS];
VOLATILE INTEGER __DRAG_DROP_target_count[__DRAG_DROP_NUM_PANELS];
VOLATILE INTEGER __DRAG_DROP_panel_devices[__DRAG_DROP_NUM_PANELS]


(***********************************************************)
(*        SUBROUTINE/FUNCTION DEFINITIONS GO BELOW         *)
(***********************************************************)
(* EXAMPLE: DEFINE_FUNCTION <RETURN_TYPE> <NAME> (<PARAMETERS>) *)
(* EXAMPLE: DEFINE_CALL '<NAME>' (<PARAMETERS>) *)

DEFINE_FUNCTION __DRAG_DROP_SET_PANELS(INTEGER panels[])
{
   if(LENGTH_ARRAY(panels) <= __DRAG_DROP_NUM_PANELS)
   {
      __DRAG_DROP_panel_devices = panels;
   }
   else
   {
      STACK_VAR INTEGER count;

      for(count = 1 ; count <= __DRAG_DROP_NUM_PANELS; count++)
      {
         __DRAG_DROP_panel_devices[count] = panels[count];
      }
```

```
        SET_LENGTH_ARRAY(__DRAG_DROP_panel_devices,count);
    }
}

DEFINE_FUNCTION __DRAG_DROP_CLEAR_DATA(INTEGER panel)
{
    STACK_VAR INTEGER count;
    __DRAG_DROP_current_drag[panel].chanPort = 0;
    __DRAG_DROP_current_drag[panel].chan     = 0;
    __DRAG_DROP_current_drag[panel].addrPort = 0;
    __DRAG_DROP_current_drag[panel].addr     = 0;
    __DRAG_DROP_current_drag[panel].buttonName = '';
    __DRAG_DROP_current_drag[panel].groupName  = '';

    count = LENGTH_ARRAY(__DRAG_DROP_current_targets[panel]);
    if(count > 0)
    {
        STACK_VAR INTEGER x;
        for(x = 1; x <= count; x++)
        {
            __DRAG_DROP_current_targets[panel][x].chanPort = 0;
            __DRAG_DROP_current_targets[panel][x].chan = 0;
            __DRAG_DROP_current_targets[panel][x].addrPort = 0;
            __DRAG_DROP_current_targets[panel][x].addr = 0;
            __DRAG_DROP_current_targets[panel][x].buttonName = '';
            __DRAG_DROP_current_targets[panel][x].valid = 0;
        }
    }
    __DRAG_DROP_target_count[panel] = 0;
}

DEFINE_FUNCTION INTEGER __DRAG_DROP_PARSE_PORT_VALUE(
                        CHAR line[],
                        INTEGER start,
                        INTEGER port,
                        INTEGER value)
{
    STACK_VAR INTEGER x, run, state;
    STACK_VAR char ch;

    x = start;
    run = 1;
    state = 0;
    ch = 0;

    port = 0;
    value = 0;
    while(run)
    {
        ch = line[x];
        switch(state)
        {
            case 0:
            {
                if(ch >= '0' && ch <= '9')
                {
                    port = port * 10 + (ch-'0');
                }
                else if(ch == ',')
                {
                    state = 1;
                }
            }
            case 1:
            {
                if(ch >= '0' && ch <= '9')
                {
                    value = value * 10 + (ch-'0')
                }
                else if(ch == ':')
                {
                    run = 0;
                }
                else if(ch == '}')
                {
                    run = 0;
                }
            }
        }
        x++;
    }
```

```
   return x;
}

DEFINE_FUNCTION INTEGER __DRAG_DROP_PARSE_NAME(CHAR line[],
                                               INTEGER start,
                                               CHAR value[])
{
   STACK_VAR INTEGER end;

   value = '';
   end = FIND_STRING(line,':',start);
   if(end <= 0)
   {
      end = FIND_STRING(line,'}',start);
   }
   if(end > start)
   {
      value = MID_STRING(line,start,end-start);
      return end+1;
   }
   return start+1;

}

DEFINE_FUNCTION INTEGER __DRAG_DROP_PARSE_VALUE(CHAR line[],
                                                INTEGER start,
                                                INTEGER value)
{
   STACK_VAR INTEGER x, run;
   STACK_VAR INTEGER ch;

   x = start;
   run = 1;
   ch = 0;
   value = 0;
   while(run)
   {
      ch = line[x]
      if(ch >= '0' && ch <= '9')
      {
         value = value * 10 + (ch-'0');
      }
      else if(ch == ':')
      {
         run = 0;
      }
      else if(ch == '}')
      {
         run = 0;
      }
      x++;
   }
   return x;
}

DEFINE_FUNCTION __DRAG_DROP_PARSE_DRAG_START(INTEGER panel, TCUSTOM s)
{
   STACK_VAR char line[200],text[2000];
   STACK_VAR INTEGER length,index;

   length = 0;
   __DRAG_DROP_CLEAR_DATA(panel);

   text = s.text;
   line = REMOVE_STRING(text,"10",1);
   length = LENGTH_STRING(line);
   while( length > 0)
   {
      if(FIND_STRING(line,'dr{',1) == 1)
      {
         index = 4;
         while (index < length)
         {
            SELECT
            {
               ACTIVE(FIND_STRING(line,'ch=',index) == index) :
               {
                  index = __DRAG_DROP_PARSE_PORT_VALUE(line,index+3,
                     __DRAG_DROP_current_drag[panel].chanPort,
                     __DRAG_DROP_current_drag[panel].chan);
```

```
                    }
                    ACTIVE(FIND_STRING(line,'ad=',index) == index) :
                    {
                        index = __DRAG_DROP_PARSE_PORT_VALUE(line,index+3,
                            __DRAG_DROP_current_drag[panel].addrPort,
                            __DRAG_DROP_current_drag[panel].addr);
                    }
                    ACTIVE(FIND_STRING(line,'gp=',index) == index) :
                    {
                        index = __DRAG_DROP_PARSE_NAME(line,index+3,
                            __DRAG_DROP_current_drag[panel].groupName);
                    }
                    ACTIVE(FIND_STRING(line,'nm=',index )== index) :
                    {
                        index = __DRAG_DROP_PARSE_NAME(line,index+3,
                            __DRAG_DROP_current_drag[panel].buttonName);
                    }
                    ACTIVE(1) :
                    {
                        index = length;
                    }
                }
            }
        }
        else if(FIND_STRING(line,'dt{',1) == 1)
        {
            index = 4;
            __DRAG_DROP_target_count[panel]++;

            while (index < length)
            {
                SELECT
                {
                    ACTIVE(FIND_STRING(line,'vl=',index) == index) :
                    {
                        if(line[index+3] == '1')
                        {
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].valid = 1;
                        }
                        else
                        {
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].valid = 0;
                        }
                        index = index+5;
                    }
                    ACTIVE(FIND_STRING(line,'ch=',index) == index) :
                    {
                        index = __DRAG_DROP_PARSE_PORT_VALUE(line,index+3,
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].chanPort,
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].chan);
                    }
                    ACTIVE(FIND_STRING(line,'ad=',index) == index) :
                    {
                        index = __DRAG_DROP_PARSE_PORT_VALUE(line,index+3,
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].addrPort,
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].addr);
                    }
                    ACTIVE(FIND_STRING(line,'nm=',index) == index) :
                    {
                        index = __DRAG_DROP_PARSE_NAME(line,index+3,
                            __DRAG_DROP_current_targets[panel][__DRAG_DROP_target_count[panel]].buttonName);
                    }
                    ACTIVE(1) :
                    {
                        index = length;
                    }
                }
            }
        }
        line = REMOVE_STRING(text,"10",1);
        length = LENGTH_STRING(line);
    }
    SET_LENGTH_ARRAY(__DRAG_DROP_current_targets[panel],__DRAG_DROP_target_count[panel]);

}

DEFINE_FUNCTION __DRAG_DROP_PRINT_DATA(INTEGER panel)
{
    STACK_VAR INTEGER x;

    SEND_STRING 0,"FORMAT('drag ch=%d',__DRAG_DROP_current_drag[panel].chanPort),
```

```
                    FORMAT(',%-5d',__DRAG_DROP_current_drag[panel].chan),
                    FORMAT('ad=%d',__DRAG_DROP_current_drag[panel].addrPort),
                    FORMAT(',%-5d',__DRAG_DROP_current_drag[panel].addr),
                    'gp=''',__DRAG_DROP_current_drag[panel].groupName,''' bn=''',
                            __DRAG_DROP_current_drag.buttonName,'''';
    for(x = 1; x <= __DRAG_DROP_target_count[panel]; x++)
    {
       SEND_STRING 0,"FORMAT('target ch=%d',__DRAG_DROP_current_targets[panel][x].chanPort),
                     FORMAT(',%-5d',__DRAG_DROP_current_targets[panel][x].chan),
                     FORMAT('ad=%d',__DRAG_DROP_current_targets[panel][x].addrPort),
                     FORMAT(',%-5d',__DRAG_DROP_current_targets[panel][x].addr),
                     'bn=''',__DRAG_DROP_current_targets[panel][x].buttonName,
                     FORMAT(''' valid=%d',__DRAG_DROP_current_targets[panel][x].valid)";
    }
}
```

Basic Demo - No Drop Groups

Overview

The following instructions illustrate creating a set of *draggable* buttons that represent input devices, and a *drop target* button that represents an output device (a VTC).

NOTE

This set of instructions uses files that are included in the "DragAndDropNoGroups" demo file which is available to download from the UI RESOURCE CENTER at www.amx.com.

The resulting demo page will provide four draggable buttons that represent source (input) devices, and one drop target button representing an output (VTC) device. End users will be able to switch sources on the VTC by dragging and dropping a draggable button within the bounds of a the drop target button (FIG. 8):

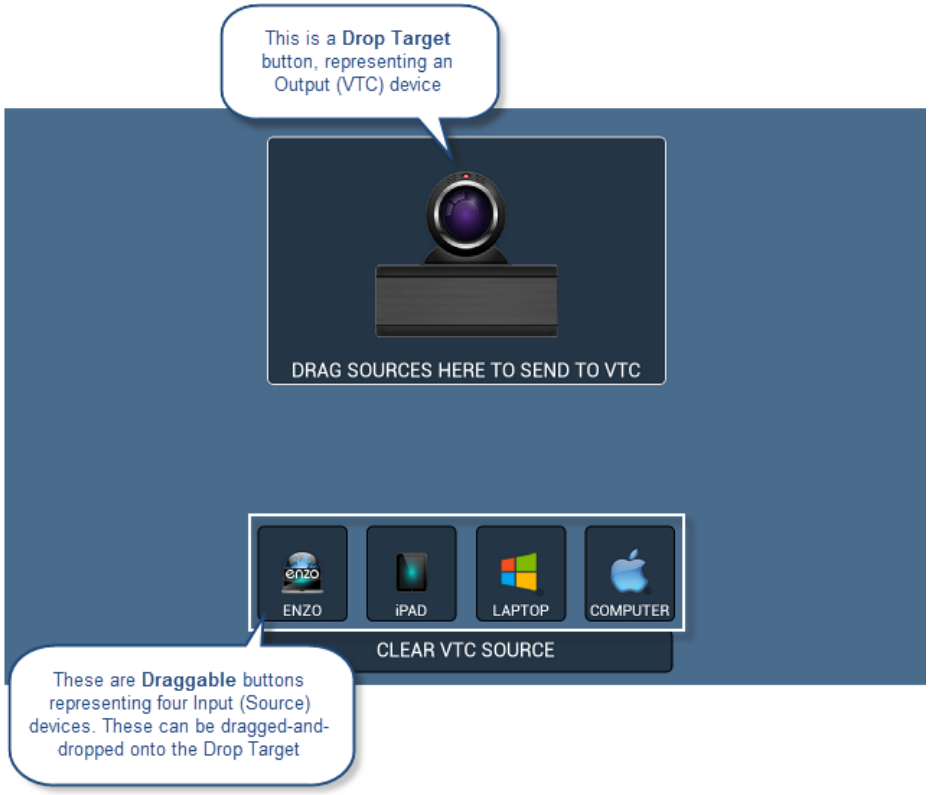


FIG. 8 Drag and Drop Demo - Four (draggable) Input buttons and one (drop target) Output button

Before You Begin

Download the *DragAndDropNoGroups.zip* file from www.amx.com and extract its contents to a known location. This ZIP file contains the following files, all of which are required for the demo described in this manual:

- **DragAndDropNoGroups.TP5** - A TPDesign5 project file, as well as all of the image files used by the Page and Buttons in this project:
  - icon-apple.png
  - icon-enzo.png
  - icon-iPad.png
  - icon-windows8.png
  - vtc.png
- **DragAndDropNoGroups.apw** - A NetLinx Studio workspace file that contains the NetLinx code:
  - DragAndDropNoGroups.axs
  - DragAndDropNoGroups.src
  - DragAndDropNoGroups.tkn
  - DragAndDropNoGroups.tko

1) Create a TPDesign5 Project/Import Images

In order to display the images on the page and buttons shown in this demo, the image files must be added to the project, via the Resource Manager - *Images* tab:

1. Open TPDesign5 (v1.3 or higher) and start a new Project (**File > New**).
2. Open the Resource Manager to the *Images* tab.
3. Click **Import** to locate and select all of the image files that were included in the *DragAndDropNoGroups.ZIP* file.
4. Click **OK** to import the selected files and return to the Resource Manager (FIG. 9):

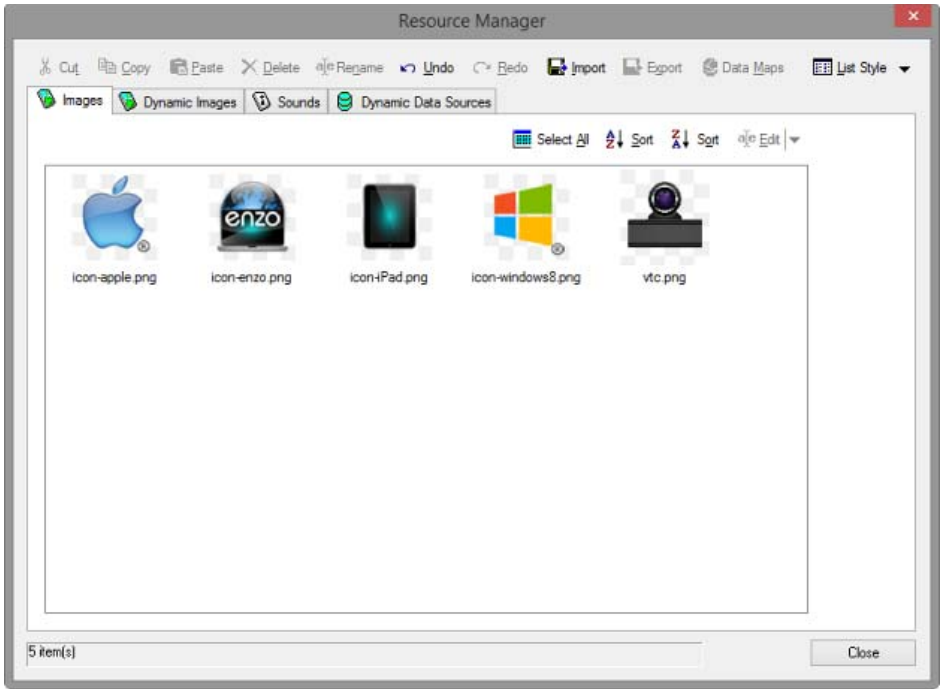


FIG. 9 Resource Manager Images tab - DragAndDropNoGroups Demo images imported

5. Click **Close** to close the Resource Manager.

NOTE

The DragAndDropNoGroups.TP5 file in the Drag and Drop demo has the images shown above already imported into the project.

2) Create & Configure a Drop Target Button

In this example, there is only a single Drop Target button that will represent the Output Device (VTC) that can accept input from the source devices represented by the draggable buttons.

Create a Drop Target Button

1. Use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **multi-state general** (FIG. 10):

FIG. 10 TPDesign5 General Properties - Type set to "general"

3. Set the button's *Drag/Drop Type* (General) property to **drop target** (FIG. 11):

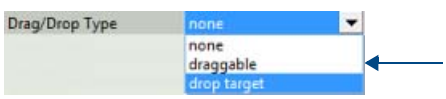


FIG. 11 TPDesign5 General Properties - Drag/Drop Type set to "drop target"

Set Drop Target Button Properties - General

Set the remaining *General* properties for the Drop Target button as shown in FIG. 12:

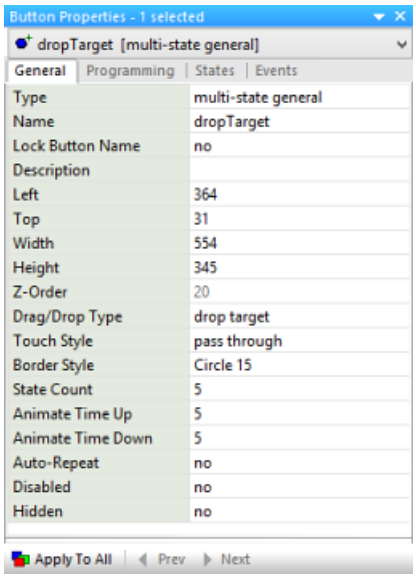


FIG. 12 Drop Target Button - General Properties

Set Drop Target Button Properties - Programming

Set the *Programming* properties for the Drop Target button as shown in FIG. 13:

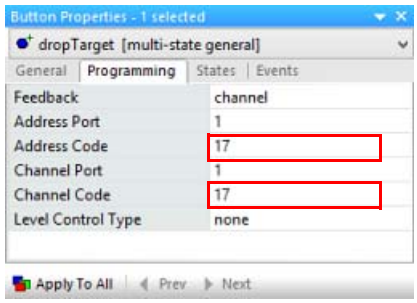


FIG. 13 Drop Target Button - Programming Properties

On the Drop Target button, set the *Address Code* to **17** and set the *Channel Code* to **17**.

Set Drop Target Button Properties - States

In this example, this button will represent the output (VTC) device. Use the *Text* (States) property to add the following labels to the button (*All States)*: **DRAG SOURCES HERE TO SEND TO VTC**.

Use the *Bitmaps* (States) property to add the VTC bitmap to the button (*All States)*:

- Note that all images must first be imported in to the project via the Resource Manager in order to be available to apply to buttons or pages in the project. The images used in this demo are pre-loaded in the TP5 project file.
- Select the drop target button and under *All States*, apply the bitmap: **VTC.png**.
- In this example, the *Bitmap Justification* is set to **center-middle**.

Drop target buttons can use states to provide a visual indication of target validity for draggable buttons. In this example, if a drag is started on a draggable button, the opacity of the drop target button is reduced to indicate that it is a drop target.

1. Select the Drop Target button and open the State Manager window.
2. In the State Manager window, select *State 1*, and set the State properties as shown in FIG. 14:
3. In the State Manager window, select *State 2*, and set the State properties as shown in FIG. 14:

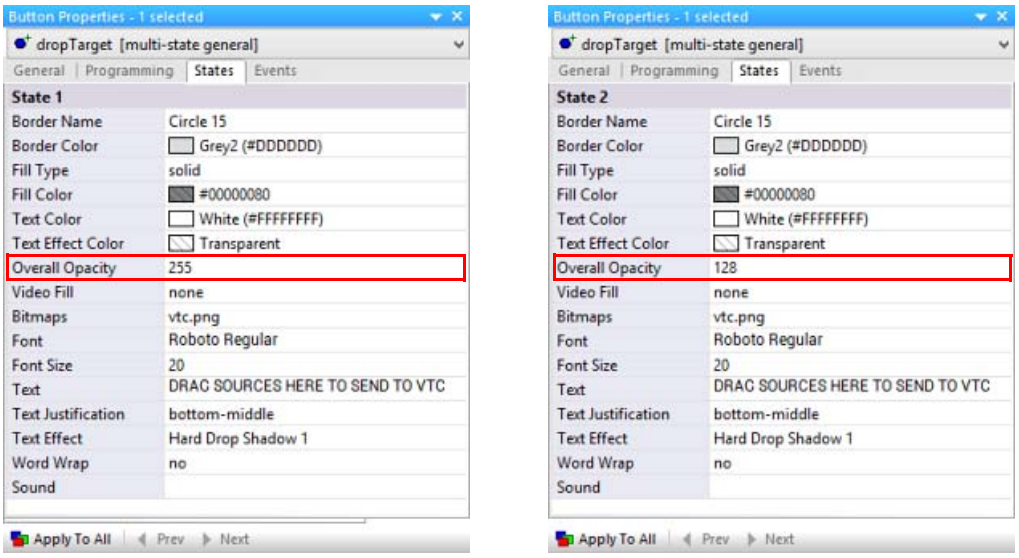


FIG. 14 State Manager context menu - Insert States

NOTE

*In this example the only difference between the two states is the **Overall Opacity** property setting: State one uses "**255**" (totally opaque), and State 2 uses "**128**" (half-opacity). This provides a visual indication that this button is a drop target.*

4. The two states are indicated in the State Manager window (FIG. 15):

FIG. 15 State Manager window indicating five states

3) Create & Configure Draggable Buttons

In this example, four draggable buttons represent four source (input) devices that are used as the input for the VTC Output device represented by the Drop Target button.

Create Four Draggable Buttons

1. In TPDesign5, open a Page and use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **general** (FIG. 16):

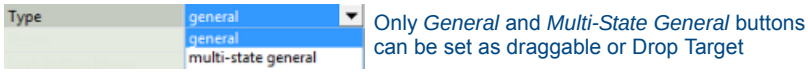


FIG. 16 TPDesign5 General Properties - Type set to "general"

3. Set the button's *Drag/Drop Type* (General) property to **draggable** (FIG. 17):

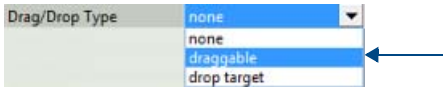


FIG. 17 TPDesign5 General Properties - Drag/Drop Type set to "draggable"

4. Repeat these steps to create a total of four draggable buttons. Alternatively, copy and paste the new button three times (FIG. 18):

FIG. 18 Draggable Buttons

Set Draggable Button Properties - General

Set the remaining *General* properties for the draggable buttons as shown in FIG. 19:

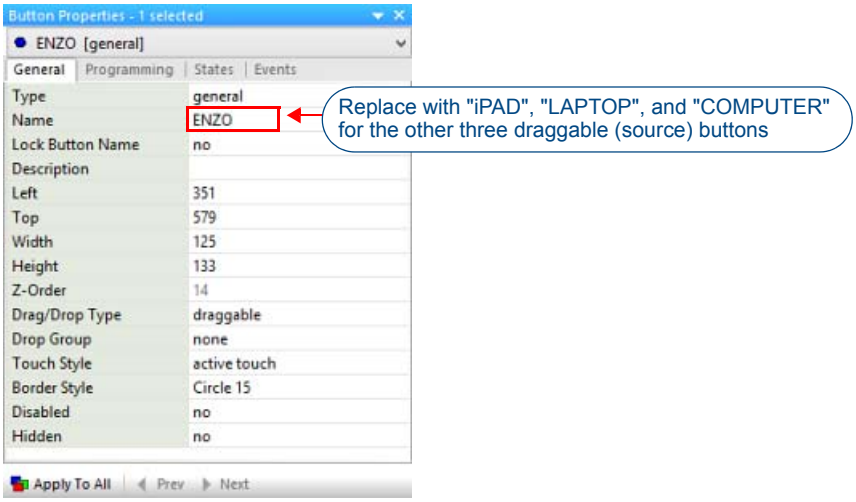


FIG. 19 Draggable Buttons - General Properties

Set Draggable Button Properties - Programming

Each of the draggable buttons needs to be configured with unique Address and Channel Codes. For this example, set the Address/Channel Codes as shown below (FIG. 20):

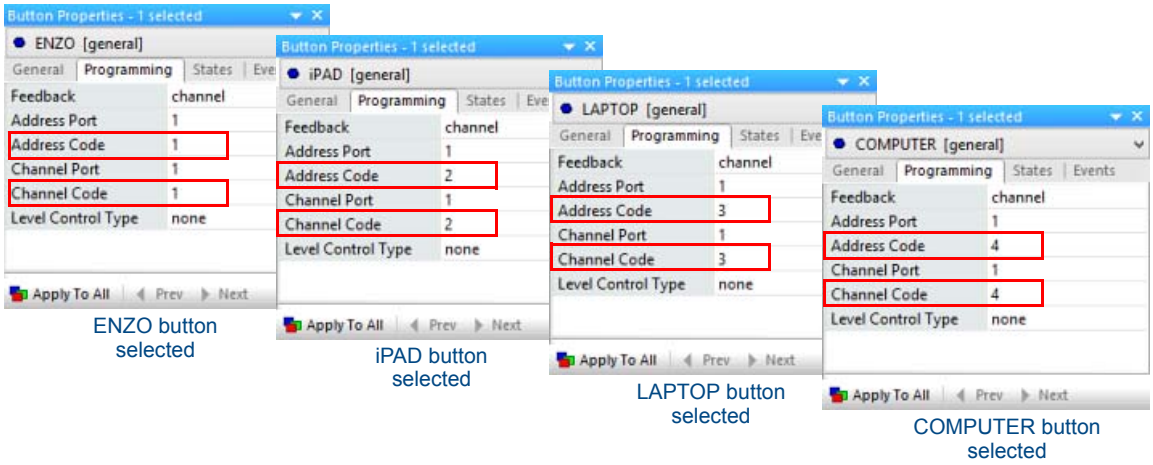


FIG. 20 Draggable Buttons - Programming Properties

- On the ENZO draggable button, set the *Address Code* to **1** and set the *Channel Code* to **1**.
- On the iPAD draggable button, set the *Address Code* to **2** and set the *Channel Code* to **2**.
- On the LAPTOP draggable button, set the *Address Code* to **3** and set the *Channel Code* to **3**.
- On the COMPUTER draggable button, set the *Address Code* to **4** and set the *Channel Code* to **4**.

Set Draggable Button Properties - States

In this example, each of these buttons will represent a different type of input (source) device. Edit the buttons to add text and icons to indicate the specific device represented by each button:

1. Use the *Text* (States) property to add labels to each of the buttons. Select each button and under *All States*, enter the following labels: **ENZO**, **iPAD**, **LAPTOP** and **COMPUTER**.
2. Use the *Bitmaps* (States) property to apply an appropriate icon to each of the buttons.

NOTE

All images must first be imported in to the project via the Resource Manager in order to be available to apply to buttons or pages in the project. The images used in this demo are pre-loaded in the TP5 project file.

- Select each button and under *All States*, apply the following bitmaps: **icon-enzo.png**, **icon-iPad.png**, **icon-windows8.png** and **icon-apple.png**.
- In this example, the *Bitmap Justification* is set to **top-middle** for all four draggable buttons.

Set the remaining *States* properties for the draggable buttons as shown in FIG. 21:

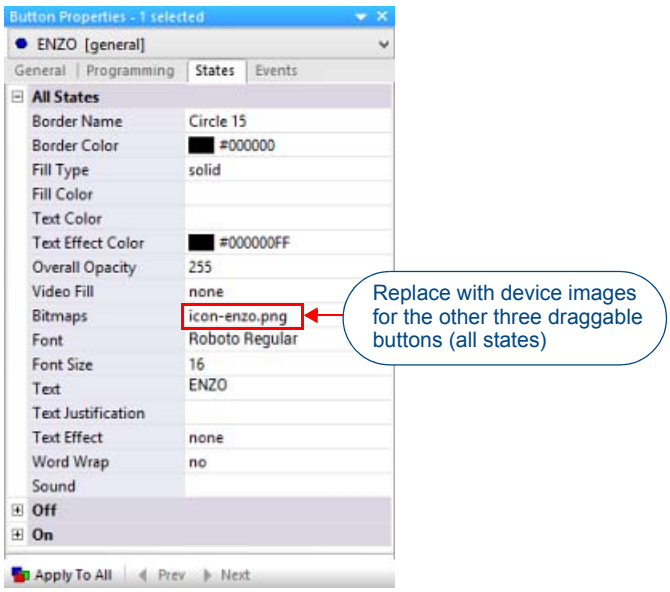


FIG. 21 Draggable Buttons - States Properties (All States shown)

For this example, the draggable buttons should look similar to the buttons shown below (FIG. 22):

FIG. 22 Draggable Buttons (Representing four Input Devices)

4) Create and Configure a "CLEAR VTC SOURCE" Button

This example includes the option for the user to "clear" the current input (Source) device setting on the VTC (FIG. 23):

FIG. 23 CLEAR VTC SOURCE button

To add a button that supports this option:

Create a "CLEAR VTC SOURCE" Button

1. Use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **general**.

Set "CLEAR VTC SOURCE" Button Properties - General

Set the remaining *General* properties for the "CLEAR VTC SOURCE" buttons as shown in FIG. 24:

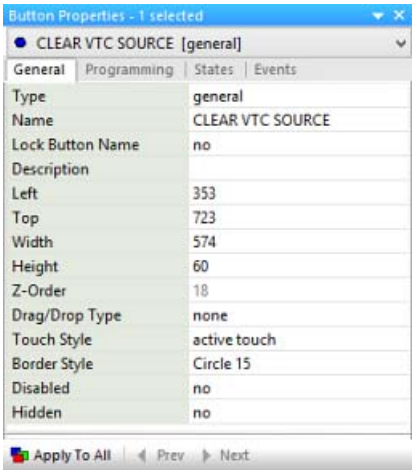


FIG. 24 "CLEAR VTC SOURCE" Button - General Properties

Set "CLEAR VTC SOURCE" Button Properties - Programming

Set the *Programming* properties for the "CLEAR VTC SOURCE" button as shown in FIG. 25:

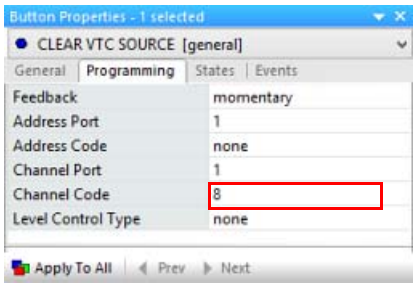


FIG. 25 "CLEAR VTC SOURCE" Button - Programming Properties

On the "CLEAR VTC SOURCE" button, set the *Channel Code* to **8**.

5) Write NetLinx Code To Respond To Custom Event

The NetLinx Code below utilizes the custom events that were configured in the TP file for "behavior" changes on the drop target buttons via the states configured earlier in this section.

1. Use NetLinx Studio 4 to add the following code to the NetLinx program loaded on the Master:

```
PROGRAM_NAME='MASTER'

DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT
//dropTargets
INTEGER btnDT  = 17
//draggables
INTEGER btnDG1 = 1
INTEGER btnDG2 = 2
INTEGER btnDG3 = 3
INTEGER btnDG4 = 4

DEFINE_VARIABLE
//an array to store our draggable buttons
INTEGER dgBTNS[] = {btnDG1 ,btnDG2 ,btnDG3 ,btnDG4}
//to store draggable address from start event
INTEGER nDragAddress = 0

DEFINE_MUTUALLY_EXCLUSIVE
([dvTP,1]..[dvTP,4])

//In this example the groups are defined as follows
//        - buttonAddresses 1,2, are assigned: group_1
//        - buttonAddresses 3,4 are assigned: group_2
//        - btnDT [17] will accept draggables from: group_2

DEFINE_EVENT

DATA_EVENT[dvTP]
{
   ONLINE:
   {
   //Let's make sure we are starting in state 1
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',1,1,0'"
   }
}

//Custom event for START [1410]
//Any time a draggable is initiated (long press, dragShadow appears)
//a START event is sent.
//CUSTOM_EVENT[dvTP,ID,Type]
CUSTOM_EVENT[dvTP,dgBTNS,1410]
{
   //Get the dragButtonAddress from the customEvent
   nDragAddress = custom.value1
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',2,2,0'"
}
//Custom event for ENTER [1411]
//Once the dragShadow enters the boundaries of a valid dropTarget
//a ENTER event is sent
CUSTOM_EVENT[dvTP,btnDT,1411]
{
    SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',2,2,0'"
}
//Custom event for EXIT [1412]
//Once the dragShadow leaves the boundaries of a valid dropTarget
//a EXIT event is sent

CUSTOM_EVENT[dvTP,btnDT,1412]
{
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',1,1,0'"
}
```

```
//Custom event for DROP [1413]
//A DROP event occurs when a draggable has been released within the boundaries
//of a valid dropTarget. A valid dropTarget is a dropTarget that has a group
//which the draggable is assigned to.
CUSTOM_EVENT[dvTP,btnDT,1413]
{
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',1,1,0'"
   //turn on the source(draggable)
   ON[dvTP,nDragAddress]
}
//Custom event for CANCEL [1414]
//A CANCEL event occurs when a draggable has been released over anything that
//is not a VALID dropTarget.
CUSTOM_EVENT[dvTP,dgBTNS,1414]
{
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',1,1,0'"
}

BUTTON_EVENT[dvTP,8] //CLEAR VTC SOURCES
{
   PUSH:
   {
   OFF[dvTP,1]
   OFF[dvTP,2]
   OFF[dvTP,3]
   OFF[dvTP,4]
   SEND_COMMAND dvTP,"'^ANI-',ITOA(btnDT),',1,1,0'"
   }
}
```

2. Save changes.

The NetLinx code shown above is included in the NetLinx Studio Workspace file (DragAndDropNoGroups.apw) that is in the DragAndDropNoGroups.ZIP file.

6) Use NetLinx Studio 4 to Compile and Transfer the Project Files

Use NetLinx Studio 4 to compile the code and transfer the project files to the Master:

1. At the top of the *DragAndDropNoGroups.axs* source code file, change the ***dvTP*** value to match the device number of your touch panel (FIG. 26):

```
PROGRAM_NAME='MASTER'

DEFINE_DEVICE
dvTP = 10001:1:0
```

FIG. 26 dvTP Device Number value - Change to match the device number of your Touch Panel

2. Compile the code (select **Build > Build Active System**).
3. Transfer the *DragAndDropNoGroups.apw* workspace file to the NetLinx Master:
   a. Select **Tools > File Transfer** to open the *File Transfer* dialog.
   b. Open the *Send* tab and clear any files that are listed by clicking **Remove All**.
   c. Click **Add** to open the *Select Files for File Transfer* dialog.
   d. Select the top-level **Projects** folder to select all files in the workspace for transfer (FIG. 27):

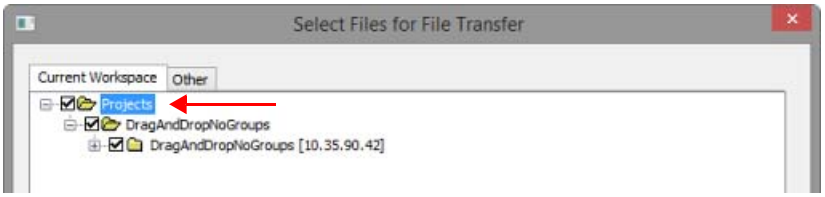


FIG. 27 Select Files for File Transfer dialog

   e. Select **OK** to return to the *File Transfer* dialog (FIG. 28):

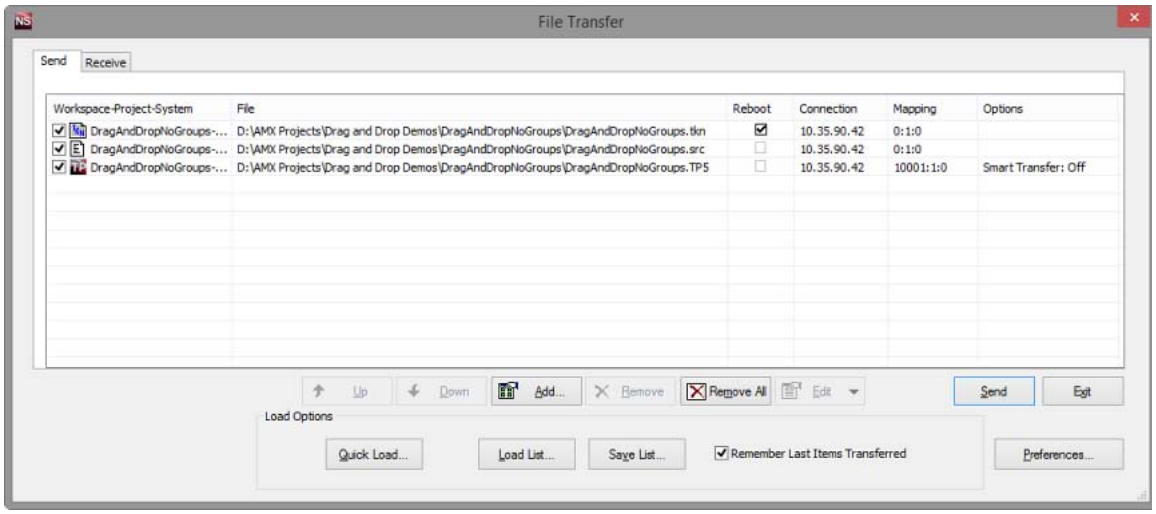


FIG. 28 File Transfer dialog - indicating files in the DragAndDropNoGroups.apw workspace queued for transfer

   f. Click **Send** to initiate the file transfer.
   g. The progress of the transfer is indicated in the Output Bar.

End Result

The result of this demo is a touch panel page with four draggable buttons representing source (input) devices and one drop target button representing an output device (VTC):

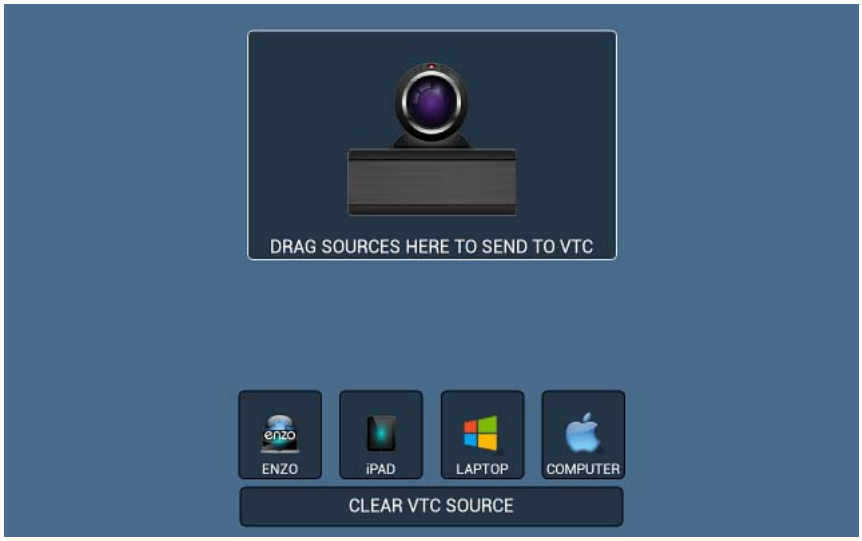


FIG. 29 Drag and Drop Demo - Page Layout

- The VTC button ("DRAG SOURCES HERE TO SEND TO VTC") is a drop target button representing an output device (VTC) that can accept any of the sources represented by the four draggable buttons. Note that in this example the VTC is a valid target for all sources, since no Drop Groups have been defined.
- Source buttons can each be dragged onto the VTC button individually. When one of the draggable buttons is released within the bounds of the VTC button, NetLinx code receives the custom events and turns on the source represented by the draggable button that was dropped.
- Press the CLEAR VTC SOURCE button to clear the current input setting.

Advanced Demo - Three Drop Groups

Overview

The following instructions illustrate creating a set of *draggable* buttons that represent input devices, and a set of *drop target* buttons that represent three output device (Displays).

NOTE

This set of instructions uses files that are included in the "Drag-and-Drop " demo file which is available to download from the UI RESOURCE CENTER at www.amx.com.

The resulting demo page will provide five draggable buttons that represent source (input) devices, and three drop target buttons representing three output (Display) devices. End users will be able to switch sources on the Displays by dragging and dropping a draggable button within the bounds of the drop target buttons (FIG. 30):

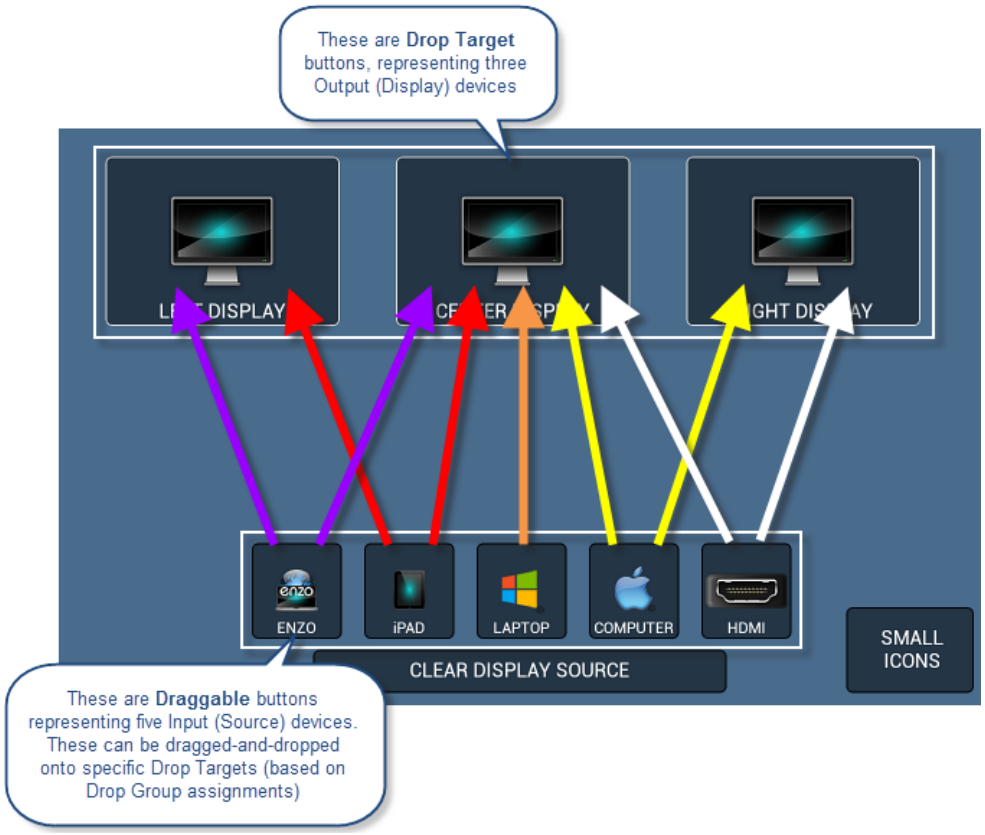


FIG. 30 Drag and Drop Demo - Five Draggable (Inputs) buttons and three Drop Target (Outputs) button

Before You Begin

Download the *AdvancedDragAndDropExample.ZIP* file from www.amx.com and extract its contents to a known location. This ZIP file contains the following files, all of which are required for the demo described in this manual:

- **AdvancedDragAndDropExample.TP5** - A TPDesign5 project file, as well as all of the image files used by the Page and Buttons in this project:
  - amxicons_target-invalid.png
  - amxicons_target-invalid-small.png
  - amxicons_target-valid.png
  - amxicons_target-valid-small.png
  - icon-apple.png
  - icon-enzo.png
  - icon-iPad.png
  - icon-windows8.png
  - SZ9_icon display.png

- **AdvancedDragAndDropExample.APW** - A NetLinx Studio workspace file that contains the NetLinx code:
  - AdvancedDragAndDropExample.axs
  - AdvancedDragAndDropExample.src
  - AdvancedDragAndDropExample.tkn
  - AdvancedDragAndDropExample.tko

1) Create a TPDesign5 Project/Import Images

In order to display the images on the page and buttons shown in this demo, the image files must be added to the project, via the Resource Manager - *Images* tab:

1. Open TPDesign5 (v1.3 or higher) and start a new Project (**File > New**).
2. Open the Resource Manager to the *Images* tab.
3. Click **Import** to locate and select all of the image files that were included in the *Drag and Drop Demo.ZIP* file.
4. Click **OK** to import the selected files and return to the Resource Manager (FIG. 31):

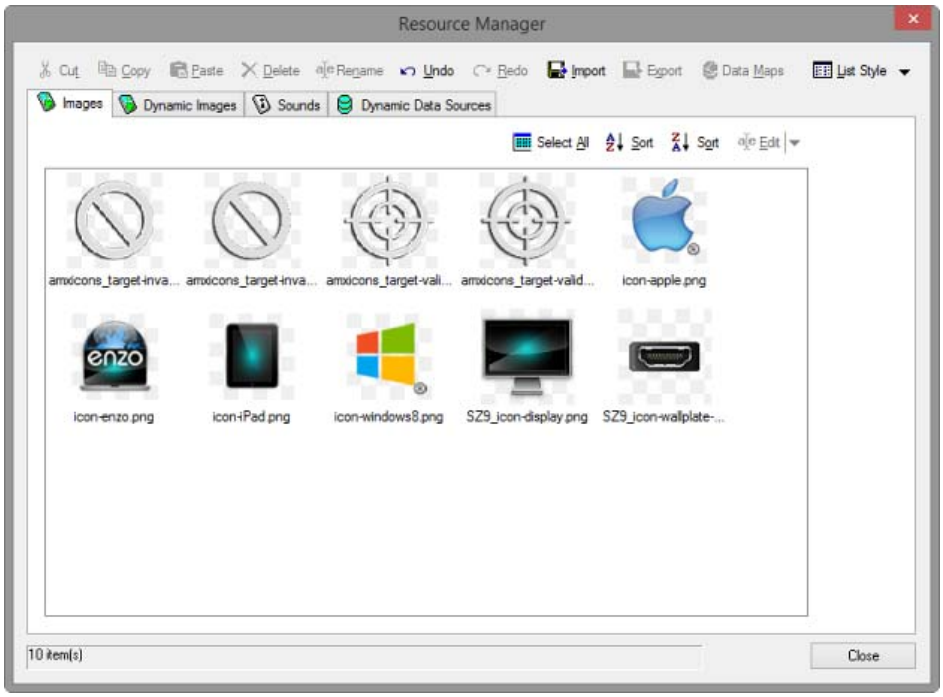


FIG. 31 Resource Manager Images tab - AdvancedDragAndDropExample Demo images imported

5. Click **Close** to close the Resource Manager.

NOTE

The AdvancedDragAndDropExample.TP5 file in the Drag and Drop demo has the images shown above already imported into the project.

2) Create & Configure Drop Target Buttons

In this example, there are three Drop Target buttons that will represent the Output devices (Displays) that can accept input from the source devices represented by the draggable buttons.

Create Three Drop Target Buttons

1. Use the Button Draw tool to create three new buttons.
2. Arrange them horizontally on the top half of the page, and enter the following names for each button, in the *Name* (General) property:
   - LEFT DISPLAY
   - CENTER DISPLAY
   - RIGHT DISPLAY
3. Set each button's *Type* (General) property to **multi-state general** (FIG. 32):

FIG. 32 TPDesign5 General Properties - Type set to "general"

4. Set each button's *Drag/Drop Type* (General) property to **drop target** (FIG. 33):

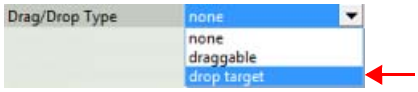


FIG. 33 TPDesign5 General Properties - Drag/Drop Type set to "drop target"

Set Drop Target Button Properties - General

Set the remaining *General* properties for the Drop Target buttons as shown in FIG. 34:

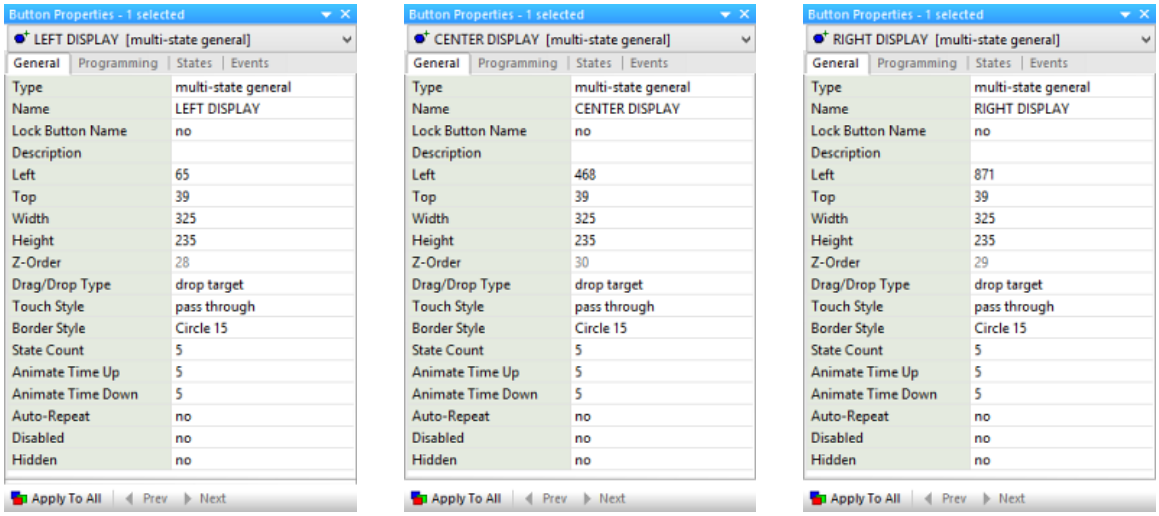


FIG. 34 Drop Target Buttons - General Properties

Set Drop Target Button Properties - Programming

Set the *Programming* properties for each of the Drop Target buttons as shown in FIG. 35:

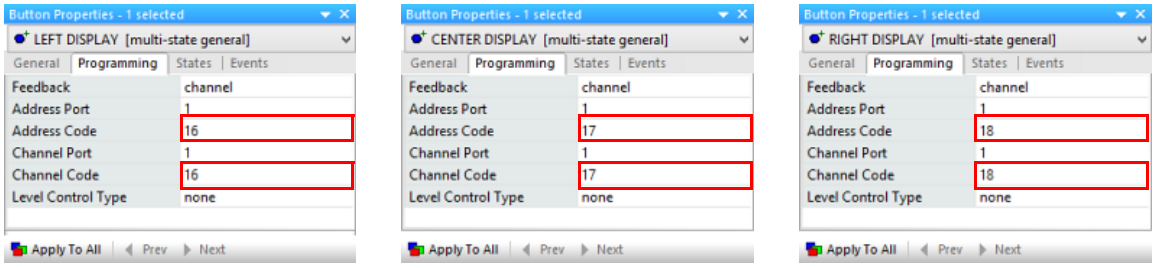


FIG. 35 Drop Target Button - Programming Properties

- On the LEFT DISPLAY button, set the *Address Code* to **16** and set the *Channel Code* to **16**.
- On the CENTER DISPLAY button, set the *Address Code* to **17** and set the *Channel Code* to **17**.
- On the RIGHT DISPLAY button, set the *Address Code* to **18** and set the *Channel Code* to **18**.

Set Drop Target Button Properties - States

In this example, these button will represent output (Display) devices. Use the *Text* (States) property to add the following labels to each button (*All States)*:

- On the LEFT DISPLAY button, set the *Text* to LEFT DISPLAY.
- On the CENTER DISPLAY button, set the *Address Code* to **17** and set the *Channel Code* to **17**.
- On the RIGHT DISPLAY button, set the *Address Code* to **18** and set the *Channel Code* to **18**.

NOTE

As explained below, the drop-targets will use multiple states to provide visual feedback to the user in terms of whether each drop target is valid or invalid for a selected draggable button. While each of these states includes a bitmap that specifically indicates whether the target is valid or not, the button text remains the same across all States. Add and configure each button's text before duplicating states to avoid having to add the text to each individual state.

Use the *Bitmaps* (States) property to add the Display bitmap to each button *(All States)*:

- Note that all images must first be imported in to the project via the Resource Manager in order to be available to apply to buttons or pages in the project. The images used in this demo are pre-loaded in the TP5 project file.
- Select each drop target button and under *All States*, apply the bitmap: **SZ9_icon-display.png**.
- In this example, the *Bitmap Justification* is set to **center-middle**.

Add States to each Drop Target Button

In this example, the drop target buttons will use multiple states to provide a visual indication of whether it is a valid target for draggable buttons.

For example, if a drag is started on a draggable button, a bitmap featuring either a "Target-Valid" or "Target-Invalid" icon is displayed on Drop Target buttons, depending on whether each Drop Target button is a valid or invalid target for the selected draggable button.

NOTE

Use Drop Groups to configure valid/invalid drop targets for draggable buttons. See the Drop Groups section on page 2 *for details.*

Additionally, this example includes the option to use either "small icons" or large icons", so there are large versions of the "Target-Valid" and "Target-Invalid" icons as well. Therefore, there are five potential bitmap arrangements that need to be available to the drop target button - these are all configured as separate *States* for these buttons.

By default, multi-state general buttons have two states. To add three states:

1. Select a Drop Target button and open the State Manager window.
2. In the General tab of the Properties window, click in the State Count property and change the value to **5** (FIG. 36):

FIG. 36 State Count (General) Property - set to "5"

3. Press **Enter** to save changes. The new states are indicated in the State Manager window (FIG. 37):

FIG. 37 State Manager window indicating five states

4. Repeat these steps for the CENTER DISPLAY and RIGHT DISPLAY Drop Target buttons.

Add a "Target-Valid" or "Target-Invalid" Icon to each State of each Drop Target Button

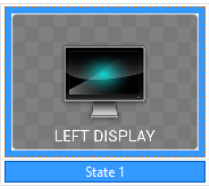
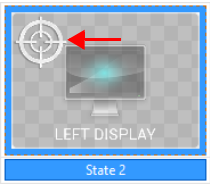
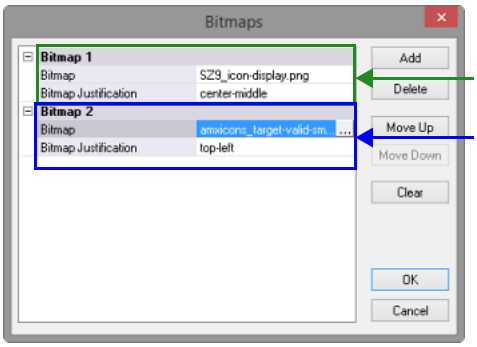
As described on page 30, each of the five States for the drop target buttons are used to indicate whether each drop target is a valid target for a selected draggable button:

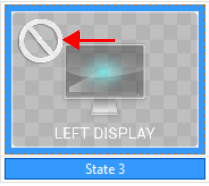
Additionally, this example includes the option to use either small or large icons, so there are large versions of the "Target-Valid" and "Target-Invalid" icons for each drop target button as well. Therefore, there are five potential bitmap arrangements that need to be available to the drop target button - these are all configured as separate *States* for each button.

NOTE

This example employs specific "Target-Valid" and "Target-Invalid" icons, as well as a change in opacity to indicate the fact that there are drop-targets, and whether they are valid targets for each draggable button. However, the icons and opacity settings are optional. Any bitmaps (or no bitmaps) can be used; any change in opacity (or no change) can be used. There are many possible ways to indicate the presence of drop target buttons as well as the validity of each drop target relative to a selected draggable

State 1 is used to when the drop target is being displayed (with no drag and drop action). Therefore, bitmaps must be added to states 2-5 to indicate a valid target or invalid target:

<table>
<tr><th colspan="2">Drop Target Button - States 1-5</th></tr>
<tr><td>State 1
 
</td><td>State 1 (no feedback)
State 1 doesn't display either the "Target-Valid" or "Target-Invalid" icon, so no bitmap is added.
This state includes the Display icon (only), and is used when the drop target is simply being displayed (without any drag and drop action).</td></tr>
<tr><td>State 2
 
</td><td>State 2 (small "Target-Valid" icon)
State 2 displays the small "Target-Valid" icon in the upper-left corner of the button. This state is used to indicate that the drop target is valid for the selected button.
1. In the State Manager window, select State 2.
2. Set the Overall Opacity (States) property to 128.
3. Click the browse (...) button in the Bitmaps (States) property to open the Bitmaps dialog. Note that Bitmap 1 is already set to SZ9_icon-display.png:
4. Click Add to add the Bitmap 2 field, and select the appropriate bitmap in the Select Resource dialog:
 

In this example, all states use "SZ9-icon-display.png" as Bitmap 1 (Bitmap Justification = center-middle)
In each State (2-5) use Bitmap 2 to show the appropriate feedback icon: "Target Valid" - small or large, or "Target Invalid" - small or large
•Set Bitmap 2 to "amxicons_target-valid-small.png".
•Set Bitmap Justification for Bitmap 2 to top-left.
5. Click OK to close the Bitmaps dialog. The image for State 2 is updated in the State Manager.</td></tr>
<tr><td>State 3
 
</td><td>State 3 (small "Target-Invalid" icon)
State 3 displays the small "Target Invalid" icon in the upper-left corner of the button. This state is used to indicate that the drop target is not valid for the selected button.
1. In the State Manager window, select State 3.
2. Set the Overall Opacity (States) property to 128.
3. Click the browse (...) button in the Bitmaps (States) property to open the Bitmaps dialog. Note that Bitmap 1 is already set to SZ9_icon-display.png.
4. Click Add to add the Bitmap 2 field, and select the appropriate bitmap in the Select Resource dialog.
•Set Bitmap 2 to "amxicons_target-invalid-small.png".
•Set Bitmap Justification for Bitmap 2 to top-left.
5. Click OK to close the Bitmaps dialog. The image for State 3 is updated in the State Manager.</td></tr>
</table>

Drop Target Button - States 1-5 (Cont.)	
State 4  	**State 4 (large "Target Valid" icon)** State 4 includes the VTC icon and a (large) "Target-Valid" icon. This state is used to indicate that the drop target is valid for the selected button (and that the LARGE ICONS option has been selected on the touch panel). **1.** In the State Manager window, select **State 4**. **2.** Set the *Overall Opacity* (States) property to **128**. **3.** Click the browse (...) button in the *Bitmaps* (States) property to open the *Bitmaps* dialog. Note that *Bitmap 1* is already set to **SZ9_icon-display.png**. **4.** Click **Add** to add the *Bitmap 2* field, and select the appropriate bitmap in the *Select Resource* dialog. •Set *Bitmap 2* to "**amxicons_target-valid.png**". •Set *Bitmap Justification* for Bitmap 2 to **scale to fit**. **5.** Click **OK** to close the *Bitmaps* dialog. The image for State 4 is updated in the State Manager.
State 5  	**State 5 (large "Target Invalid" icon)** State 5 includes the VTC icon and a (large) "Target-Invalid" icon in the middle of the button. This state is used to indicate that the drop target is *not* valid for the selected button (and that the LARGE ICONS option has been selected on the touch panel). **1.** In the State Manager window, select **State 5**. **2.** Set the *Overall Opacity* (States) property to **128**. **3.** Click the browse (...) button in the *Bitmaps* (States) property to open the *Bitmaps* dialog. Note that *Bitmap 1* is already set to **SZ9_icon-display.png**. **4.** Click **Add** to add the *Bitmap 2* field, and select the appropriate bitmap in the *Select Resource* dialog. •Set *Bitmap 2* to "**amxicons_target-valid.png**". •Set *Bitmap Justification* for Bitmap 2 to **scale to fit**. **5.** Click **OK** to close the *Bitmaps* dialog. The image for State 2 is updated in the State Manager.

The result of these updates, as indicted in the State Manager window are shown in FIG. 38:

FIG. 38 State Manager - Drop Target button states with icons placed

Set Drop Target Button Properties - Events

TPDesign5 (1.3.23 or higher) supports three new Events for drop target buttons: **Drop Enter, Drop Exit** and **Drop** (see the *Drag and Drop-Specific Events* section on page 4 for details):

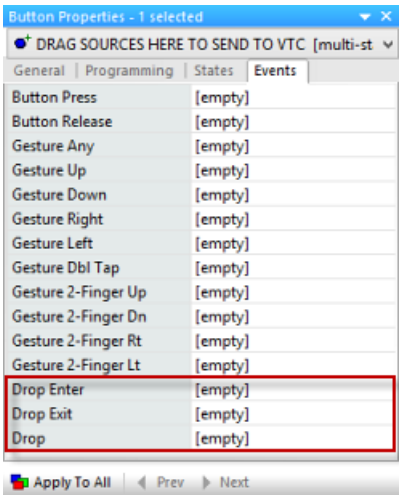


FIG. 39 Events for Drop Target Buttons

Button Event Actions are listed in the *Edit Event Actions* dialog. Use the **Add Action** option in this dialog to create new custom (event) actions via the *Edit Custom Action* dialog (FIG. 40):

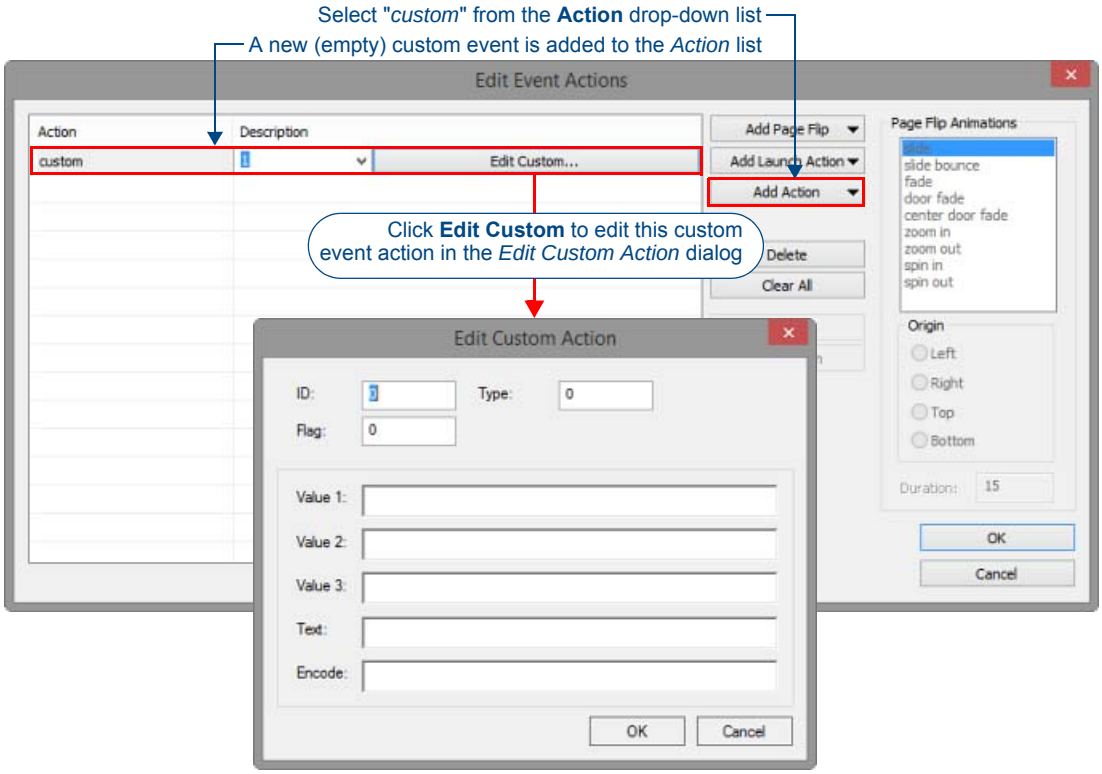


FIG. 40 Edit Event Actions dialog indicating a new (empty) custom event

Configure the "Drop Enter" Event for All Drop Target Buttons

The following steps describe how to configure custom events that will be sent to the Master, and picked up with our NetLinx code (which then handles the visual changes). Refer to page 48 to view the accompanying code.

1. Select a drop target button (in this example, start with the "LEFT DISPLAY" button).
2. In the *Events* tab of the Properties window, select the **Drop Enter** event and click the browse (...) button to open the *Edit Event Actions* dialog.
3. Click **Add Action**, and select "*custom*" from the drop-down list - this adds a new (empty) custom event action to the *Action* list.
4. Click **Edit Custom** to open the *Edit Custom Action* dialog. Use the fields in this dialog to define the event action for the selected drop target button event.

To configure the *Drop Enter* event for the drop target buttons, enter the **ID**, **Type**, **Flag** and **Value 1** fields according the table below. These fields must be configured for each drop target button in the demo.

Note that in this case, the Custom Action Settings values are the same for all three drop target buttons:

Custom Action Settings for "Drop Enter" Event	
"LEFT DISPLAY", CENTER DISPLAY," and "RIGHT DISPLAY" drop target buttons	• ID: **17** • Type: **2** • Flag: **0** • Value 1: **${dropChannelCode}**

Once they have been configured, the custom event properties are displayed in the *Drop Enter* property for each drop target button (FIG. 41):

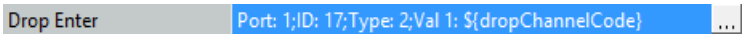


FIG. 41 Drop Enter Event indicating sample data

Configure the "Drop Exit" Event for All Drop Target Buttons

The following steps describe how to configure custom events that will be sent to the Master, and picked up with our NetLinx code (which then handles the visual changes). Refer to page 48 to view the accompanying code.

1. Select a drop target button (in this example, start with the "LEFT DISPLAY" button).
2. In the *Events* tab of the Properties window, select the **Drop Exit** event and click the browse (...) button to open the *Edit Event Actions* dialog.
3. Click **Add Action**, and select "*custom*" from the drop-down list.
4. This adds a new (empty) custom event action to the *Action* list.
5. Click **Edit Custom** to open the *Edit Custom Action* dialog. Use the fields in this dialog to define the event action for the selected drop target button event.

To configure the *Drop Exit* event for the drop target buttons, enter the **ID**, **Type**, **Flag** and **Value 1** fields according the table below. These fields must be configured for each drop target button in the demo.

Note that in this case, the Custom Action Settings values are the same for all three drop target buttons:

Custom Action Settings for "Drop Exit" Event	
"LEFT DISPLAY", CENTER DISPLAY," and "RIGHT DISPLAY" drop target buttons	• ID: **17** • Type: **3** • Flag: **0** • Value 1: **${dropChannelCode}**

Once they have been configured, the custom event properties are displayed in the *Drop Exit* property for each drop target button (FIG. 42):

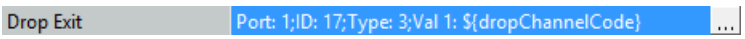


FIG. 42 Drop Exit Event indicating sample data

Configure the "Drop" Event for All Drop Target Buttons

The following steps describe how to configure custom events that will be sent to the Master, and picked up with our NetLinx code (which then handles the visual changes). Refer to page 48 to view the accompanying code.

1. Select a drop target button (in this example, start with the "LEFT DISPLAY" button).
2. In the *Events* tab of the Properties window, select the **Drop Exit** event and click the browse (...) button to open the *Edit Event Actions* dialog.
3. Click **Add Action**, and select "*custom*" from the drop-down list.
4. This adds a new (empty) custom event action to the *Action* list.
5. Click **Edit Custom** to open the *Edit Custom Action* dialog. Use the fields in this dialog to define the event action for the selected drop target button event.

To configure the *Drop* event for the drop target buttons, enter the **ID**, **Type**, **Flag** and **Value 1** fields according the table below. These fields must be configured for each drop target button in the demo.

Note that in this case, the Custom Action Settings values are the same for all three drop target buttons:

Custom Action Settings for "Drop" Event	
"LEFT DISPLAY", CENTER DISPLAY," and "RIGHT DISPLAY" drop target buttons	• ID: **17** • Type: **4** • Flag: **0** • Value 1: **${dropChannelCode}**

Once they have been configured, the custom event properties are displayed in the *Drop* property for the selected button (FIG. 43):

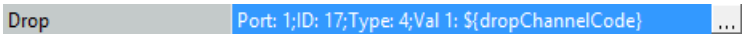


FIG. 43 Drop Event indicating sample data

When all of the Events have been configured for the drop target buttons, save your project file.

Add Each Drop Target Button to a Drop Group

Each Drop Target button will be added a Drop Group. The Drop Groups that have been created in this demo are named "*group_1*", "*group_2*" and "*group_3*" (see page 37 for details on creating these Drop Groups).

This will allow draggable buttons associated each group to use specific drop target buttons as a "valid" targets.

For example, only draggable buttons that are associated with "group_1" will be able to use the *group 1* drop target button as a valid target. Only draggable buttons that are associated with "group_2" will be able to use the *group 2* drop target button as a valid target. Only draggable buttons that are associated with "group_3" will be able to use the *group 3* drop target button as a valid target.

Drop Group assignments are managed in the *Drop-Target Groups* dialog (FIG. 44):

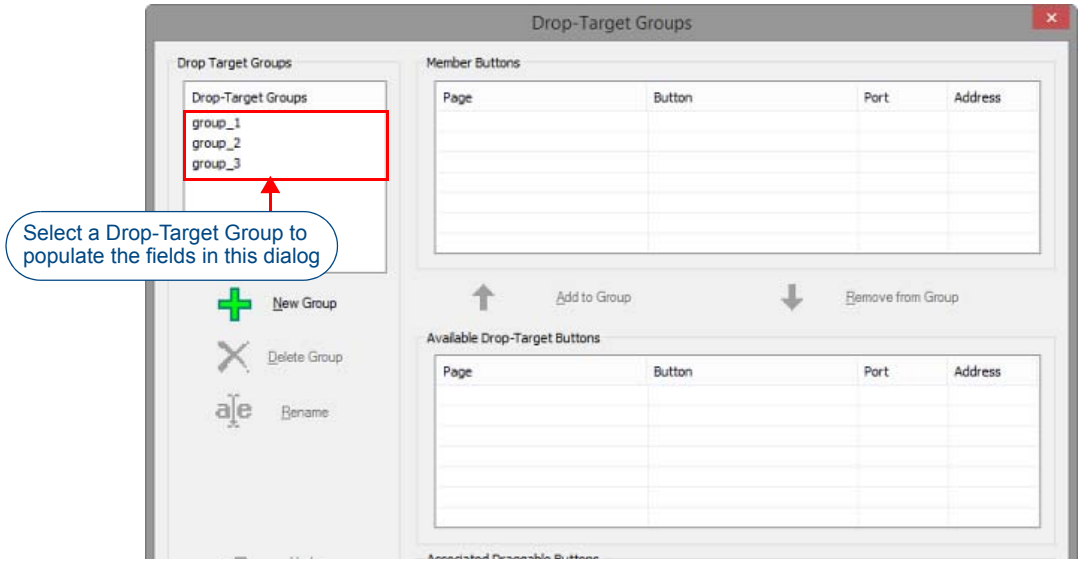


FIG. 44 Drop-Target Groups dialog (detail)

Add the LEFT DISPLAY and CENTER DISPLAY Drop Target Buttons To "group_1"

1. Select **Panel > Edit Drop Target Groups** to open the *Drop-Target Groups* dialog.
2. In the *Drop Target Groups* window, select "**group_1**".
3. The Drop Target buttons created in Step 3 (see page 37) are indicated in the *Available Drop Target Buttons* window. Select LEFT DISPLAY and click **Add to Group** to move it into the *Member Buttons* window.
4. Select CENTER DISPLAY and click **Add to Group** to move it into the *Member Buttons* window.
5. The *Member Buttons* window indicates that the LEFT DISPLAY and CENTER DISPLAY Drop Target buttons are "members" of the "*group_1*" Drop-Target Group (FIG. 45):

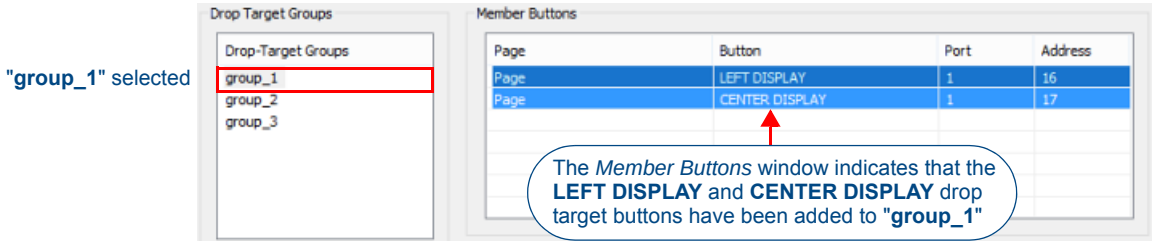


FIG. 45 Drop-Target Groups dialog - the LEFT DISPLAY dropTarget button is a member of "group_1"

Add the CENTER DISPLAY Drop Target Button To "group_2"

1. Select **Panel > Edit Drop Target Groups** to open the *Drop-Target Groups* dialog.
2. In the *Drop Target Groups* window, select "**group_2**".
3. Select CENTER DISPLAY and click **Add to Group** to move it into the *Member Buttons* window. This indicates that the CENTER DISPLAY Drop Target button is now a "member" of the "*group_2*" Drop-Target Group (FIG. 46):

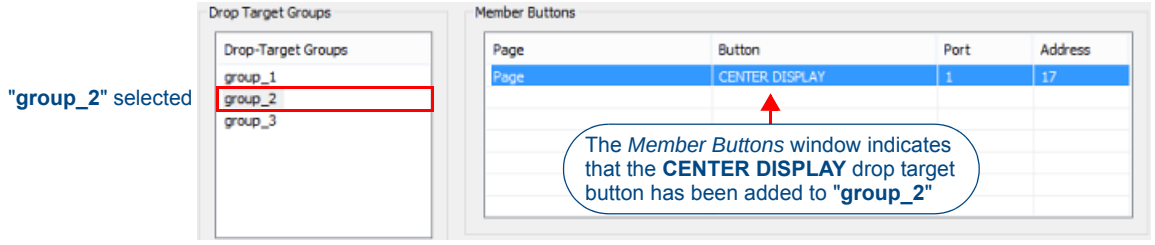


FIG. 46 Drop-Target Groups dialog - the CENTER DISPLAY dropTarget button is a member of "group_2"

Add the RIGHT DISPLAY Drop Target Button To "group_3"

1. Select **Panel > Edit Drop Target Groups** to open the *Drop-Target Groups* dialog.
2. In the *Drop Target Groups* window, select "**group_3**".
3. Select RIGHT DISPLAY and click **Add to Group** to move it into the *Member Buttons* window. This indicates that the RIGHT DISPLAY Drop Target button is now a "member" of the "*group_3*" Drop-Target Group (FIG. 47):

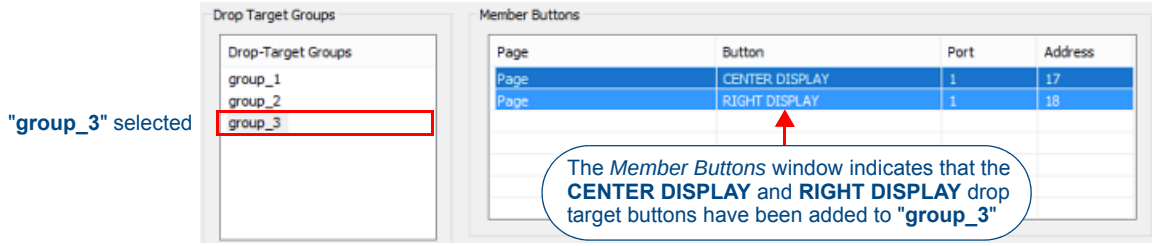


FIG. 47 Drop-Target Groups dialog - the RIGHT DISPLAY dropTarget button is a member of "group_3"

After all three drop target buttons have been assigned to their Drop Groups, click **Close** to save changes and close the *Drop-Target Groups* dialog.

3) Create Drop Groups

Drop Groups are created via the *Drop-Target Groups* dialog. After creating a drop group, drop target buttons will then be available for assignment.

- Drop Target buttons are added as "Members" of a specific Drop Group.
- Draggable buttons are associated with a specific Drop Group via the Drop Group (General) button property.

Once a Drop Target button has been added to a Drop Group (as a "member"), only draggable buttons that are associated with that Drop Group can be dragged and dropped onto the Drop Target button. Refer to the *Drop Groups* section on page 2 for a more detailed explanation of Drop Groups. In this example there are three Drop Target buttons, each of which represents a Display as an output device (see FIG. 30 on page 27):

- LEFT DISPLAY - This will be a valid drop target for the *ENZO* and *iPAD* draggable buttons.
- CENTER DISPLAY - This will be a valid drop target for all of the draggable buttons (*ENZO*, *iPAD*, *LAPTOP*, *COMPUTER* and *HDMI*).
- RIGHT DISPLAY - This will be a valid drop target for the *COMPUTER* and *HDMI* draggable buttons .

To create new Drop Groups:

1. Select **Panel > Edit Drop-Target Groups** to open the *Drop-Target Groups* dialog (FIG. 48):

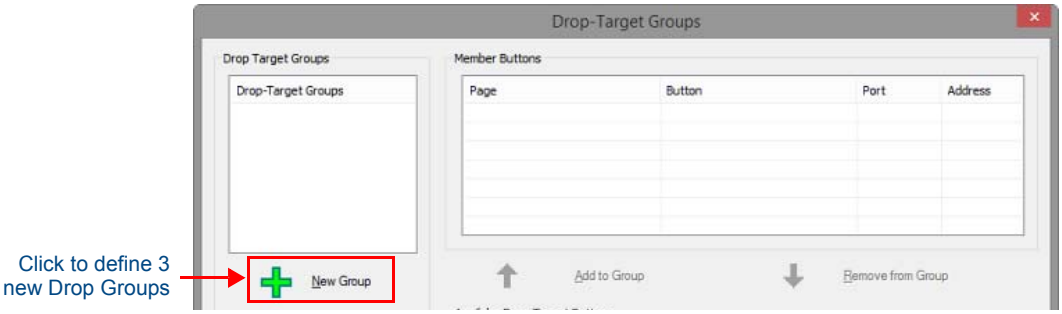


FIG. 48 Drop-Target Groups dialog

2. Click **New Group** to add a new Drop Group, via the *Create Drop-Target Group* dialog.
   a. Create a new group named "**group_1**", (FIG. 49):

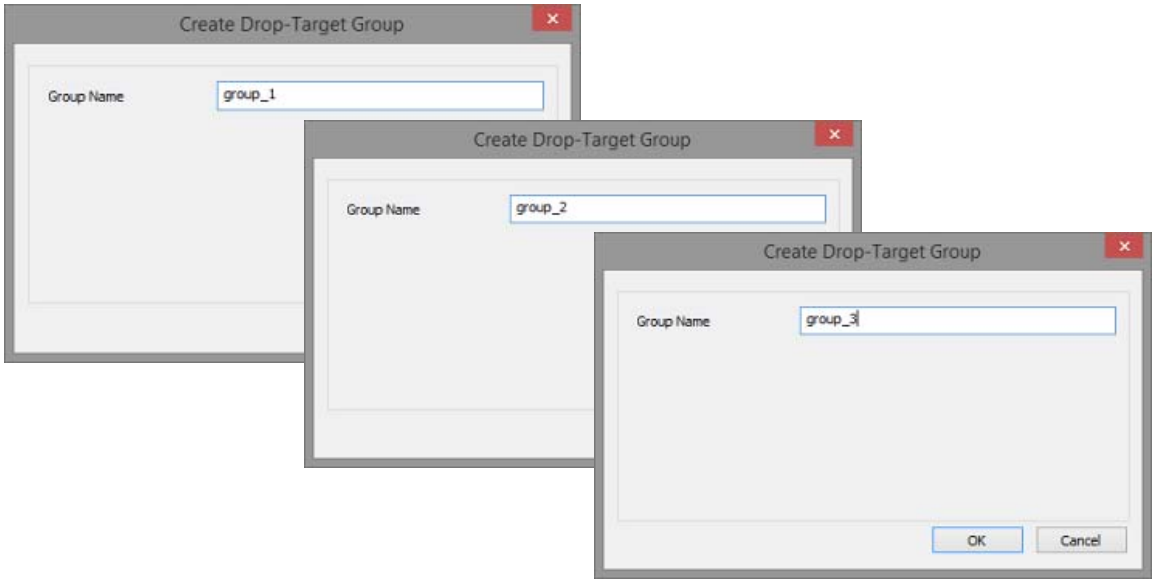


FIG. 49 Create Drop-Target Group dialog

   b. Click **OK** to save changes and close the *Create Drop Group Target* dialog.
   c. Create two additional groups named "**group_2**" and "**group_3**".
   d. The new Groups are indicated in the *Drop Target Groups* window (FIG. 50):
3. Click **Close** to save changes and close the *Drop-Target Groups* dialog.

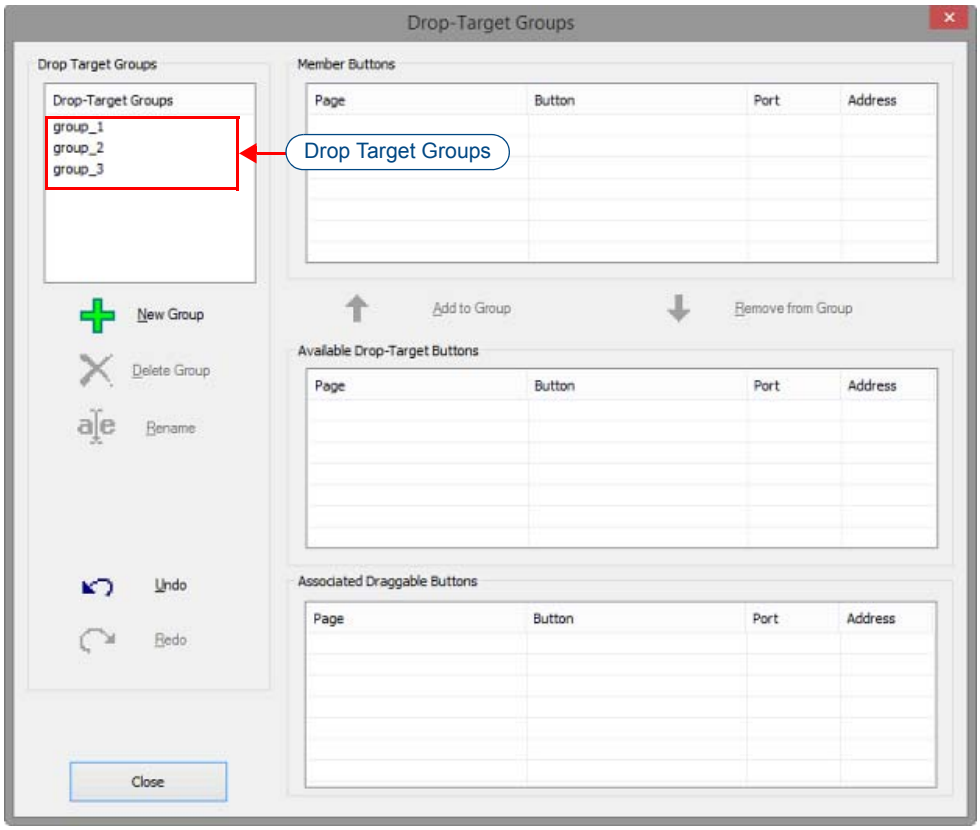


FIG. 50 Drop-Target Groups dialog - indicating "group_1", "group_2" and "group_3"

4) Create & Configure Draggable Buttons

In this example, four draggable buttons represent four source (input) devices that are used as inputs for the Output (Display) devices represented by the three Drop Target buttons.

Create Five Draggable Buttons

1. Use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **general** (FIG. 51):

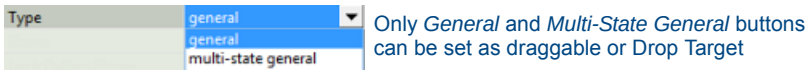


FIG. 51 TPDesign5 General Properties - Type set to "general"

3. Set the button's *Drag/Drop Type* (General) property to **draggable** (FIG. 52):

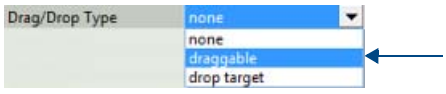


FIG. 52 TPDesign5 General Properties - Drag/Drop Type set to "draggable"

4. Repeat these steps to create a total of five draggable buttons. Alternatively, copy and paste the new button four times (FIG. 53):

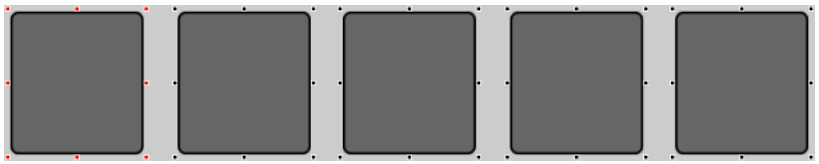


FIG. 53 Draggable Buttons

Set Draggable Button Properties - General

Set the remaining *General* properties for the draggable buttons as shown in FIG. 54:

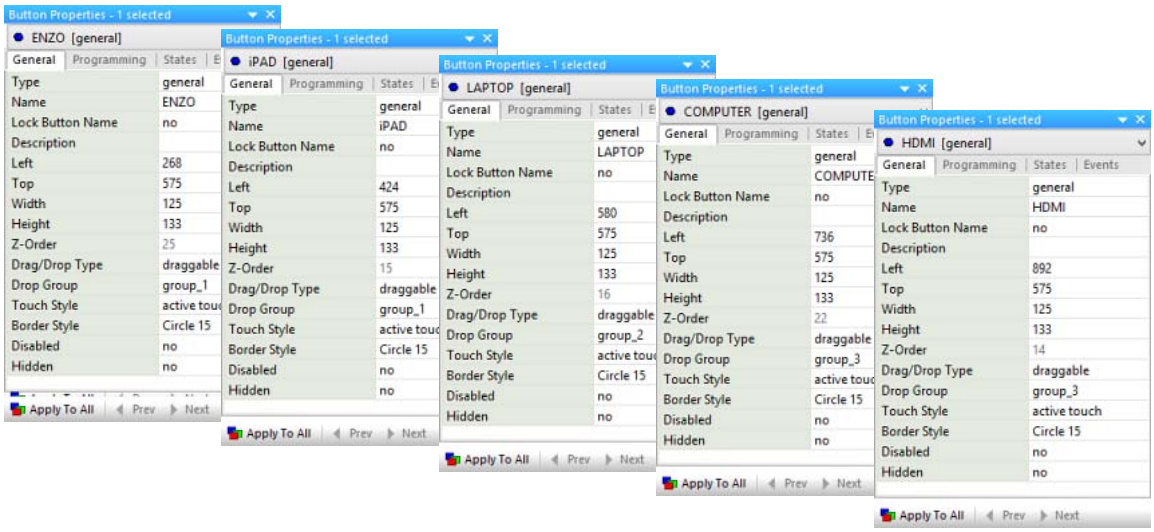


FIG. 54 Draggable Buttons - General Properties

Associate Draggable Buttons With a Drop Group

In this example, three Drop Groups have been created. Draggable buttons are associated with a specific Drop Group via the **Drop Group** (General) property (FIG. 55):

FIG. 55 General Properties - Drop Group

FIG. 55 indicates that three Drop Groups have been created: "group_1", "group_2" and "group_3". In this example, the ENZO and iPAD draggable buttons will be assigned to "*group_1*". The LAPTOP draggable button will be assigned to "*group_2*". The COMPUTER and HDMI draggable buttons will be assigned to "*group_3*":

1. Select the ENZO draggable button.
   a. In the *General* tab of the Properties window, click on **Drop Group** to access the drop-down list of all drop groups currently defined in this project.
   b. Select "**group_1**". This selection associates the ENZO draggable button with the "group_1" Drop Group.

   As a result, the ENZO button will treat the LEFT DISPLAY and CENTER DISPLAY drop target buttons as valid targets (since they are both members of "*group_1*").
2. Select the iPAD draggable button.
   a. In the *General* tab of the Properties window, click on **Drop Group** to access the drop-down list of all drop groups currently defined in this project.
   b. Select "**group_1**". This selection associated the selected draggable button with the "group_1" Drop Group.

   As a result, the iPAD button will treat the LEFT DISPLAY and CENTER DISPLAY drop target buttons as valid targets (since they are both members of "*group_1*").
3. Select the LAPTOP draggable button.
   a. In the *General* tab of the Properties window, click on **Drop Group** to access the drop-down list of all drop groups currently defined in this project.
   b. Select "**group_2**". This selection associated the selected draggable button with the "group_2" Drop Group.

   As a result, the LAPTOP button will treat the CENTER DISPLAY drop target button as it's only valid targets (since it is a the only member of "*group_2*").

4. Select the COMPUTER draggable button.
   a. In the *General* tab of the Properties window, click on **Drop Group** to access the drop-down list of all drop groups currently defined in this project.
   b. Select "**group_3**". This selection associated the selected draggable button with the "group_2" Drop Group - the same Drop Group that includes the Drop Target button in this example.

   As a result, the COMPUTER button will treat the CENTER DISPLAY and RIGHT DISPLAY drop target buttons as valid targets (since they are both members of "*group_3*").
5. Select the HDMI draggable button.
   a. In the *General* tab of the Properties window, click on **Drop Group** to access the drop-down list of all drop groups currently defined in this project.
   b. Select "**group_3**". This selection associated the selected draggable button with the "group_2" Drop Group - the same Drop Group that includes the Drop Target button in this example.

   As a result, the HDMI button will treat the CENTER DISPLAY and RIGHT DISPLAY drop target buttons as valid targets (since they are both members of "*group_3*").

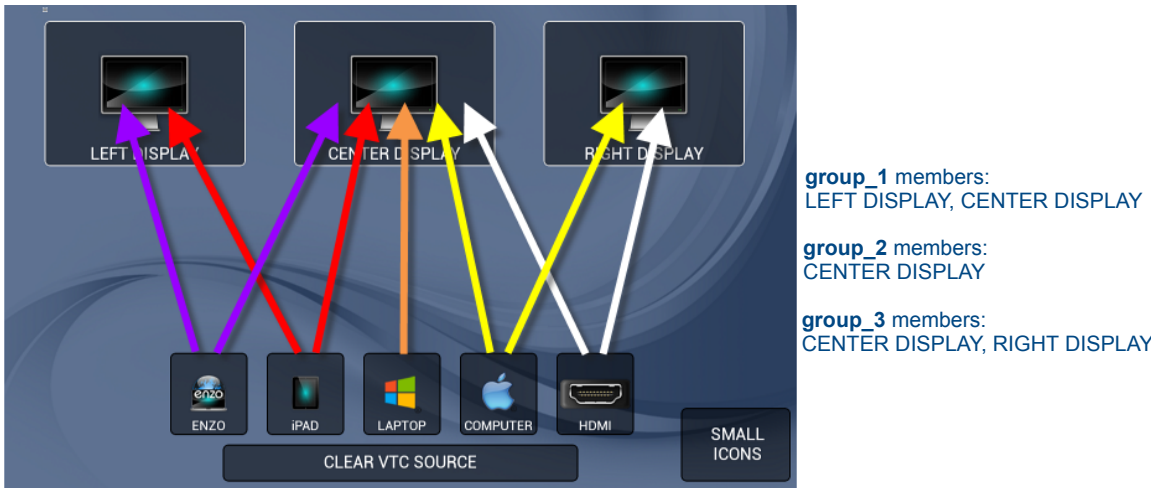


FIG. 56 Drop Target and Draggable Buttons - Drop Group assignments

Set Draggable Button Properties - Programming

Each of the draggable buttons needs to be configured with unique Address and Channel Codes. For this example, set the Address/Channel Codes as shown below (FIG. 57):

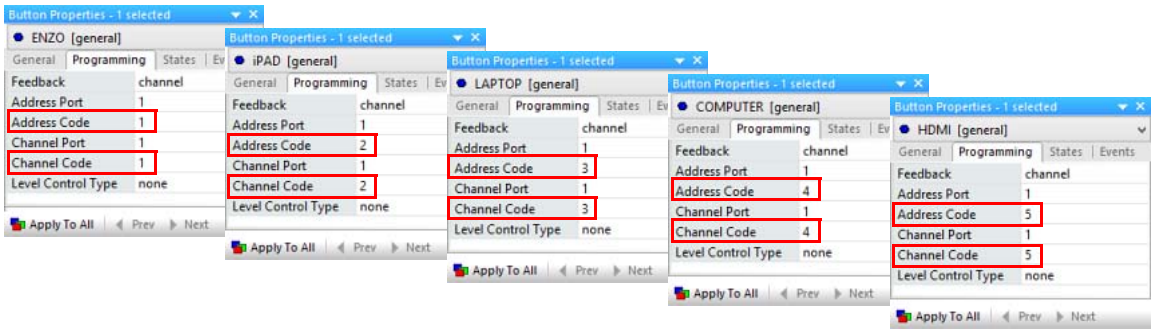


FIG. 57 Draggable Buttons - Programming Properties

- On the ENZO draggable button, set the *Address Code* to **1** and set the *Channel Code* to **1**.
- On the iPAD draggable button, set the *Address Code* to **2** and set the *Channel Code* to **2**.
- On the LAPTOP draggable button, set the *Address Code* to **3** and set the *Channel Code* to **3**.
- On the COMPUTER draggable button, set the *Address Code* to **4** and set the *Channel Code* to **4**.
- On the HDMI draggable button, set the *Address Code* to **5** and set the *Channel Code* to **5**.

Set Draggable Button Properties - States

In this example, each of these buttons will represent a different type of input (source) device. Edit the buttons to add text and icons to indicate the specific device represented by each button:

1. Use the *Text* (States) property to add labels to each of the buttons. Select each button and under *All States*, enter the following labels: **ENZO**, **iPAD**, **LAPTOP**, **COMPUTER** and **HDMI**.
2. Use the *Bitmaps* (States) property to apply an appropriate icon to each of the buttons.
   - Note that all images must first be imported in to the project via the Resource Manager in order to be available to apply to buttons or pages in the project. The images used in this demo are pre-loaded in the TP5 project file.
   - Select each button and under *All States*, apply the following bitmaps: **icon-enzo.png**, **icon-iPad.png**, **icon-windows8.png**, **icon-apple.png** and **SZ9_icon-wallplate-hdmi.png**.
   - In this example, the *Bitmap Justification* is set to **top-middle** for all five draggable buttons.

Set the remaining *States* properties for the draggable buttons as shown in FIG. 58:

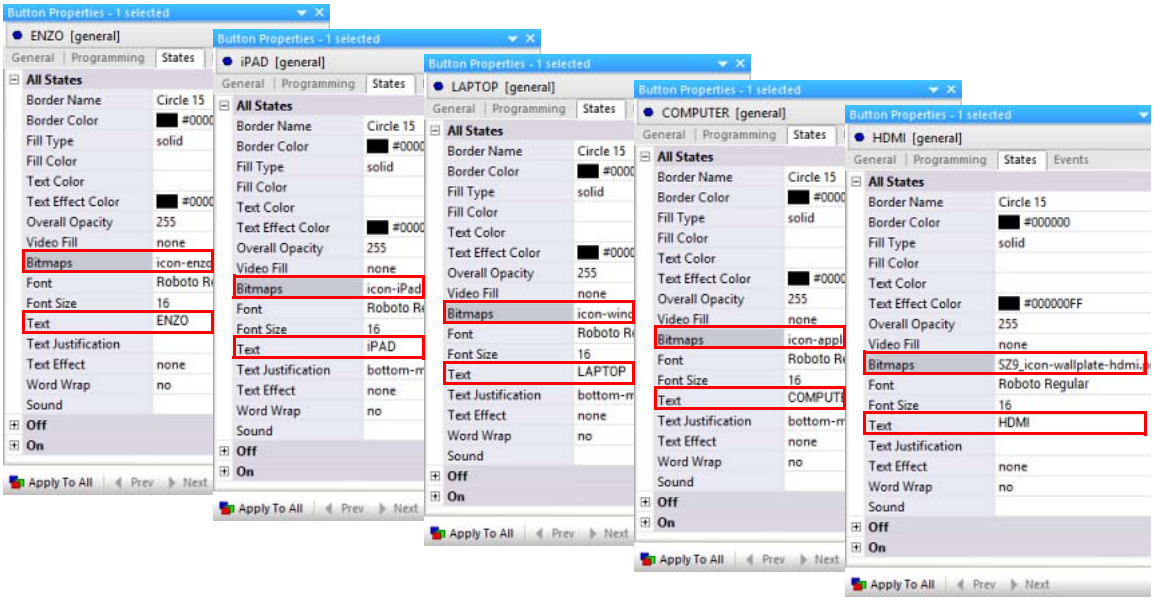


FIG. 58 Draggable Buttons - States Properties (All States shown)

For this example, the draggable buttons should look similar to the buttons shown below (FIG. 59):

FIG. 59 Draggable Buttons (Representing four Input Devices)

Set Draggable Button Properties - Events

TPDesign5 (1.3.23 or higher) supports a set of new Events for Draggable buttons: **Drag Start** and **Drag Cancel** (see the *Drag and Drop-Specific Events* section on page 4 for details):

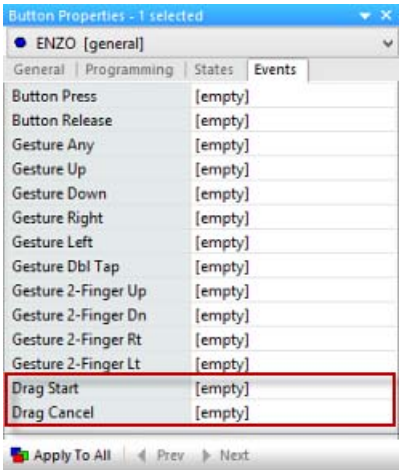


FIG. 60 Events for Draggable Buttons

Button Event Actions are listed in the *Edit Event Actions* dialog. Use the **Add Action** option in this dialog to create new custom (event) actions via the *Edit Custom Action* dialog (FIG. 61):

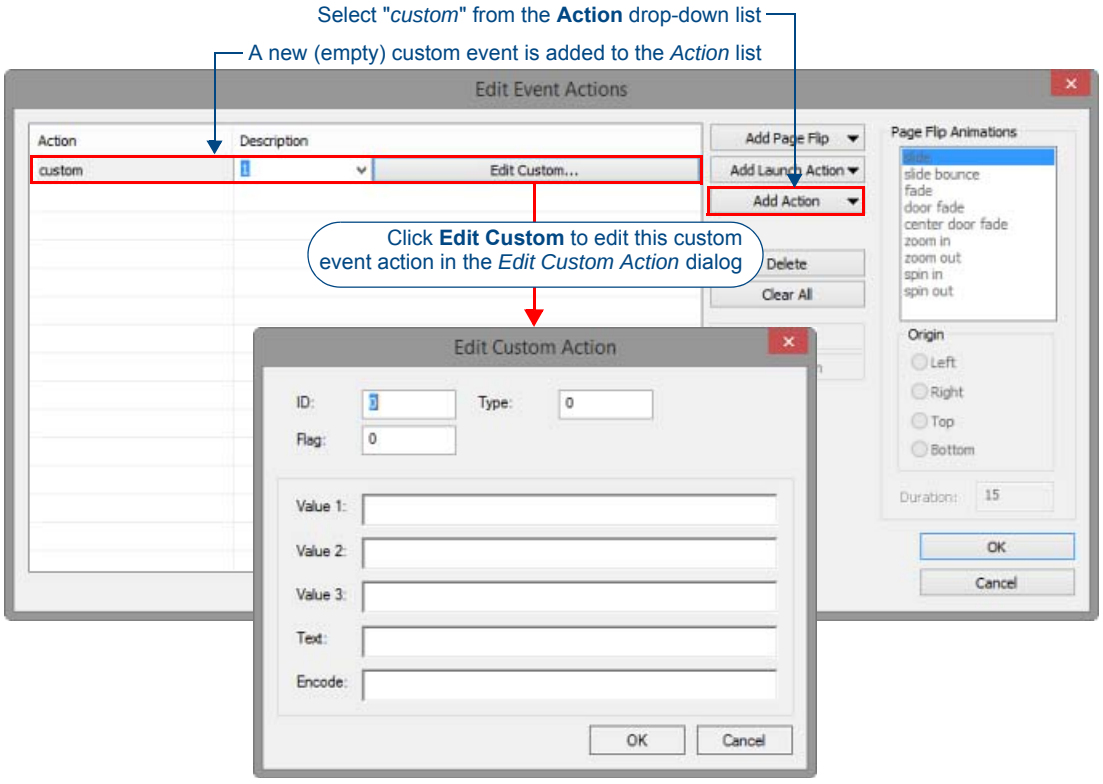


FIG. 61 Edit Event Actions dialog indicating a new (empty) custom event

Configure the "Drag Start" Event for Draggable Buttons

The following steps describe how to configure custom events that will be sent to the Master, and picked up with our NetLinx code (which then handles the visual changes). Refer to page 48 to view the accompanying code.

1. Select a draggable button (for example, the "ENZO" button).
2. In the *Events* tab of the Properties window, select the **Drag Start** event and click the browse (...) button to open the *Edit Event Actions* dialog.

3. Click **Add Action**, and select "*custom*" from the drop-down list (FIG. 62):

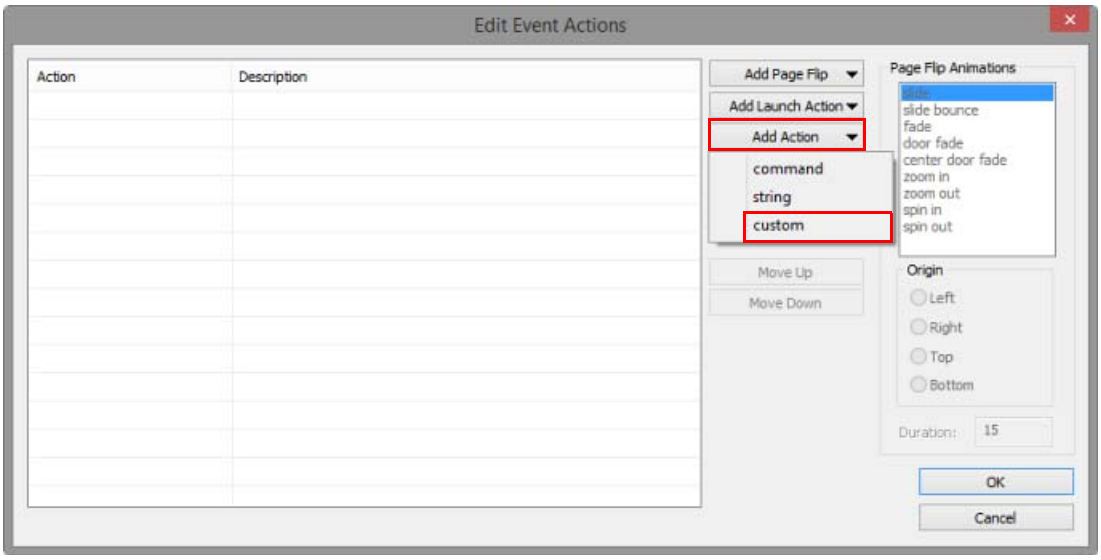


FIG. 62 Edit Event Actions dialog - Add Action drop-down list

4. This adds a new (empty) custom event action to the *Action* list (FIG. 63):

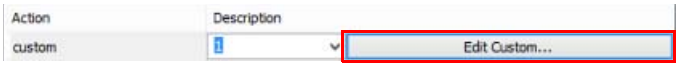
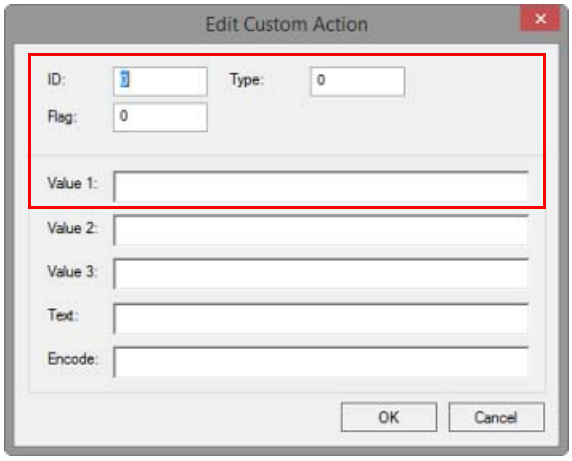


FIG. 63 Edit Event Actions dialog - new (empty) custom event action

5. Click **Edit Custom** to open the *Edit Custom Action* dialog (FIG. 64). Use the fields in this dialog to define the event action for the selected button/draggable button event.



Note: In this example, the *Value 2*, *Value 3*, *Text* and *Encode* fields are not used.

FIG. 64 Edit Custom Action dialog

6. To configure the *Drag Start* Event for the selected button, enter the **ID**, **Type**, **Flag** and **Value 1** fields according the table below. These fields must be configured for each draggable button.
7. Click **OK** to save changes and close the *Edit Custom Action* dialog and return to the Edit Event Actions dialog.
8. Press **OK** to save changes and close this dialog.

The Custom Action settings for each draggable button in this demo are provided in the table below. Note that the *Drag Start* settings are identical for all buttons, with the exception of the **ID** value, which identifies each button:

Custom Action Settings for "Drag Start" Event	
"ENZO" button	• ID: **1** • Type: **1** • Flag: **0** • Value 1: **${dragChannelCode}** • Text: **${dragGroupName}**
"iPAD" button	• ID: **2** • Type: **1** • Flag: **0** • Value 1: **${dragChannelCode}** • Text: **${dragGroupName}**
"LAPTOP" button	• ID: **3** • Type: **1** • Flag: **0** • Value 1: **${dragChannelCode}** • Text: **${dragGroupName}**
"COMPUTER" button	• ID: **4** • Type: **1** • Flag: **0** • Value 1: **${dragChannelCode}** • Text: **${dragGroupName}**
"HDMI" button	• ID: **5** • Type: **1** • Flag: **0** • Value 1: **${dragChannelCode}** • Text: **${dragGroupName}**

Once they have been configured The custom event properties are displayed in the *Drag Start* property for the selected button (FIG. 65):

FIG. 65 Drag Start Event indicating sample data

Configure the "Drag Cancel" Event for Draggable Buttons

The following steps describe how to configure custom events that will be sent to the Master, and picked up with our NetLinx code (which then handles the visual changes). Refer to page 48 to view the accompanying code.

1. Select a draggable button (for example, the "ENZO" button).
2. In the *Events* tab of the Properties window, select the **Drag Start** event and click the browse (...) button to open the *Edit Event Actions* dialog.
3. Click **Add Action**, and select "*custom*" from the drop-down list (see FIG. 62 on page 43):
4. This adds a new (empty) custom event action to the *Action* list (see FIG. 63 on page 43):
5. Click **Edit Custom** to open the *Edit Custom Action* dialog (see FIG. 64 on page 43). Use the fields in this dialog to define the event action for the selected button/draggable button event.

To configure the *Drag Cancel* Event for the selected button, enter the **ID**, **Type**, **Flag** and **Value 1** fields according the table below. These fields must be configured for each draggable button.

The Custom Action settings for each draggable button in this demo are provided in the table below. Note that the *Drag Cancel* settings are identical for all buttons, with the exception of the **ID** value, which identifies each button:

Custom Action Settings for "Drag Cancel" Event	
"ENZO" button	• ID: **1** • Type: **5** • Flag: **0** • Value 1: **${dragChannelCode}**
"iPAD" button	• ID: **2** • Type: **5** • Flag: **0** • Value 1: **${dragChannelCode}**
"LAPTOP" button	• ID: **3** • Type: **5** • Flag: **0** • Value 1: **${dragChannelCode}**
"COMPUTER" button	• ID: **4** • Type: **5** • Flag: **0** • Value 1: **${dragChannelCode}**
"HDMI" button	• ID: **5** • Type: **5** • Flag: **0** • Value 1: **${dragChannelCode}**

Once they have been configured The custom event properties are displayed in the *Drag Cancel* property for the selected button (FIG. 66):

FIG. 66 Drag Cancel Event indicating sample data

5) Add a "SMALL/LARGE ICONS" Button

This example includes a button that allows the end-user to toggle between small and large "target valid" and "target invalid" icons on the drop target buttons, when a draggable button has started a drag (FIG. 67):

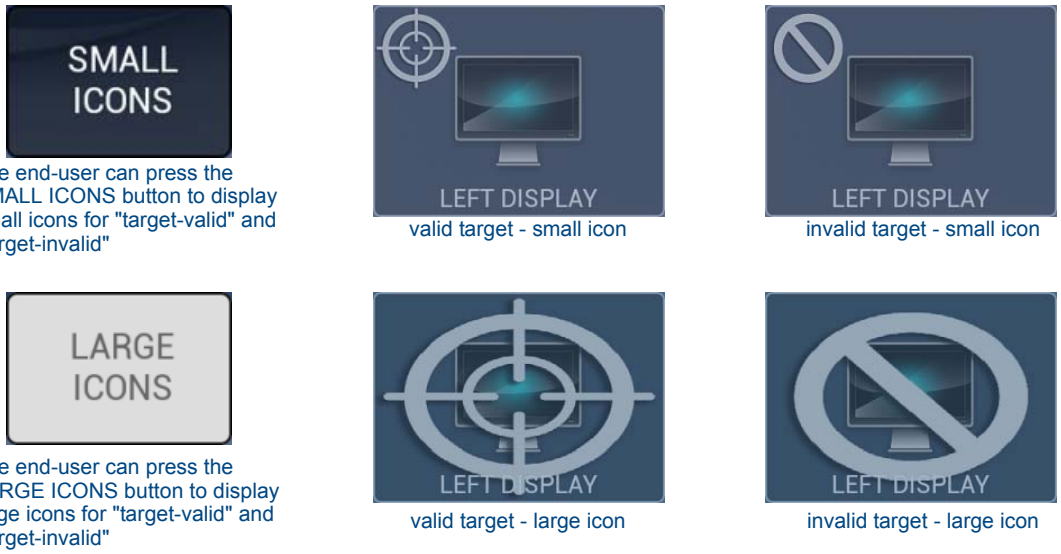


FIG. 67 SMALL ICONS and LARGE ICONS - as they will appear on the touch panel

Create a "SMALL/LARGE ICONS" Button

1. Use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **general**.

Set the remaining *General* properties for the "SMALL/LARGE ICONS" as shown in FIG. 68:

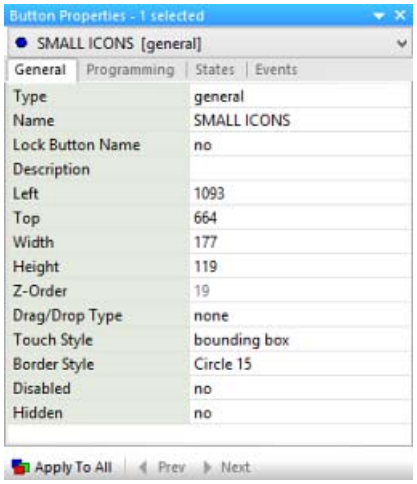


FIG. 68 "SMALL/LARGE ICONS" Button - General Properties

Set "SMALL/LARGE ICONS" Button Properties - Programming

Set the *Programming* properties for the "SMALL/LARGE ICONS" button as shown in FIG. 69:

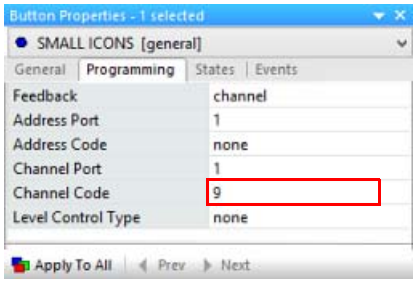


FIG. 69 "SMALL/LARGE ICONS" Button - Programming Properties

- On the "SMALL/LARGE ICONS" button, set the *Channel Port* to **9**.

Set "SMALL/LARGE ICONS" Button Properties - States

Set the *States* properties for the "SMALL/LARGE ICONS" button as shown in FIG. 70:

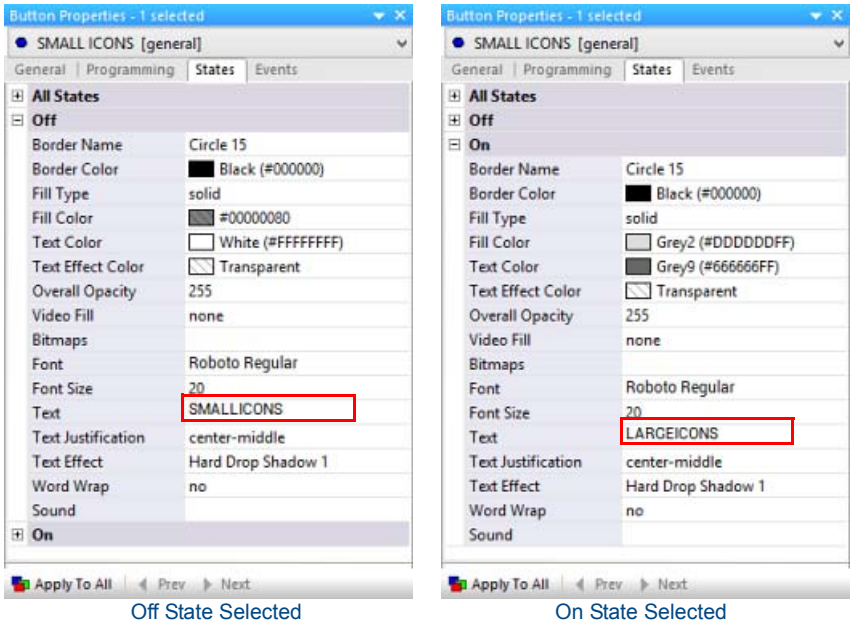


FIG. 70 "SMALL/LARGE ICONS" Button - States Properties

- Set the *Text* property on State 1 to "SMALL ICONS"
- Set the *Text* property on State 2 to "LARGE ICONS"

6) Add a "CLEAR DISPLAY SOURCE" Button

This example includes the option for the user to "clear" the current input (Source) device setting on the Displays (FIG. 71):

FIG. 71 CLEAR DISPLAY SOURCE button

To add a button that supports this option:

Create a "CLEAR DISPLAY SOURCE" Button

1. Use the Button Draw tool to create a new button.
2. Set the button's *Type* (General) property to **general**.

Set "CLEAR DISPLAY SOURCE" Button Properties - General

Set the remaining *General* properties for the "CLEAR DISPLAY SOURCE" buttons as shown in FIG. 72:

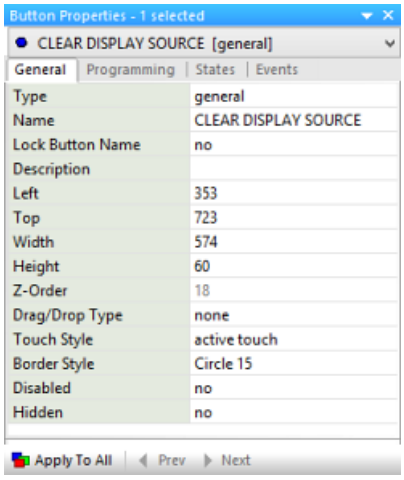


FIG. 72 "CLEAR DISPLAY SOURCE" Button - General Properties

Set "CLEAR DISPLAY SOURCE" Button Properties - Programming

Set the *Programming* properties for the "CLEAR DISPLAY SOURCE" button as shown in FIG. 73:

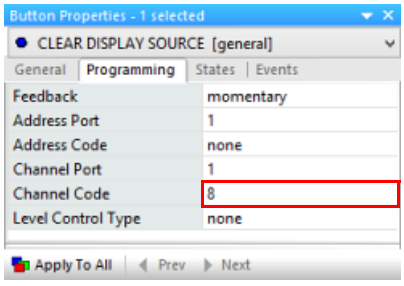


FIG. 73 "CLEAR DISPLAY SOURCE" Button - Programming Properties

On the "CLEAR DISPLAY SOURCE" button, set the *Channel Code* to **8**.

7) Write NetLinx Code To Respond To Custom Event

The NetLinx Code below utilizes the custom events that were configured in the TP file for "behavior" changes on the drop target buttons via the states configured earlier in this section.

1. Use NetLinx Studio 4 to add the following code to the NetLinx program loaded on the Master:

```
PROGRAM_NAME='MASTER'

DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT
//dropTargets
INTEGER leftDT   = 16
INTEGER centerDT = 17
INTEGER rightDT  = 18
//draggables
INTEGER btnDG1 = 1
INTEGER btnDG2 = 2
INTEGER btnDG3 = 3
INTEGER btnDG4 = 4
INTEGER btnDG5 = 5

DEFINE_VARIABLE
//an array to store our dropTarget buttons
INTEGER dTBTNS[] = {leftDT ,centerDT ,rightDT}
//an array to store our draggable buttons
INTEGER dgBTNS[] = {btnDG1 ,btnDG2 ,btnDG3 ,btnDG4,btnDG5}
//an array to store draggable buttons belonging to group_1
INTEGER dgG1[] = {btnDG1,btnDG2}
//an array to store draggable buttons belonging to group_2
INTEGER dgG2[] = {btnDG3}
//an array to store draggable buttons belonging to group_3
INTEGER dgG3[] = {btnDG4,btnDG5}
//to track small/large Icon
INTEGER nLargeIcon = 0
//to store draggable address from start event
INTEGER nDragAddress = 0
//to store which group we are in
INTEGER ngroupID = 0

DEFINE_MUTUALLY_EXCLUSIVE
([dvTP,1]..[dvTP,5])

//In this example the groups are defined as follows
//        - buttonAddresses 1,2  are assigned: group_1
//        - buttonAddress   3    are assigned: group_2
//        - buttonAddresses 4,5  are assigned: group_3
//        - leftDT   [16] will accept draggables from: group_1
//        - centerDT [17] will accept draggables from: group_1, group_2, group_3
//        - rightDT  [18] will accept draggables from: group_3

DEFINE_EVENT

DATA_EVENT[dvTP]
{
   ONLINE:
   {
      nLargeIcon = 0

      //By default ^BDC is enabled, let's disable it
      SEND_STRING dvTP,"'BDC-0,0,0,0,0'"

      //Let's make sure we are starting in state 1
      SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',1,1,0'"
      SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',1,1,0'"
      SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',1,1,0'"
   }
}
```

```
//Custom event for START [1]
//Any time a draggable is initiated (long press, dragShadow appears)
//a START event is sent.
//CUSTOM_EVENT[dvTP,ID,Type]
CUSTOM_EVENT[dvTP,dgBTNS,1]
{
   //to store the draggable group name
   CHAR cDragGroup[DATA_MAX_VALUE_LENGTH]

   //Get the dragButtonAddress from the customEvent
   nDragAddress = custom.value1
   cDragGroup = custom.text

   if(COMPARE_STRING(cDragGroup,'group_1'))
   {
      ngroupID = 1
      if(nLargeIcon)
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',4,4,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',4,4,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',5,5,0'"
      }
      else
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',2,2,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',2,2,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',3,3,0'"
      }
   }
   else if(COMPARE_STRING(cDragGroup,'group_2'))
   {
      ngroupID = 2
      if(nLargeIcon)
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',5,5,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',4,4,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',5,5,0'"
      }
      else
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',3,3,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',2,2,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',3,3,0'"
      }
   }
   else
   {
      ngroupID = 3
      if(nLargeIcon)
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',5,5,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',4,4,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',4,4,0'"
      }
      else
      {
         SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',3,3,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',2,2,0'"
         SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',2,2,0'"
      }
   }
}

//Since we are signaling what groups are valid on the START event,
//there is no need to handle ENTER or EXIT events.
```

```
//Custom event for DROP [4]
//A DROP event occurs when a draggable has been released within the boundaries
//of a valid dropTarget. A valid dropTarget is a dropTarget that has a group
//which the draggable is assigned to.
CUSTOM_EVENT[dvTP,dTBTNS,4]
{
    SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',1,1,0'"
    SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',1,1,0'"
    SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',1,1,0'"
    //turn on the source(draggable)
    ON[dvTP,nDragAddress]
}
//Custom event for CANCEL [5]
//A CANCEL event occurs when a draggable has been released over anything that
//is not a VALID dropTarget.
CUSTOM_EVENT[dvTP,dgBTNS,5]
{
   SEND_COMMAND dvTP,"'^ANI-',ITOA(leftDT),',1,1,0'"
   SEND_COMMAND dvTP,"'^ANI-',ITOA(centerDT),',1,1,0'"
   SEND_COMMAND dvTP,"'^ANI-',ITOA(rightDT),',1,1,0'"
}

BUTTON_EVENT[dvTP,8] //CLEAR DISPLAY SOURCES
{
   PUSH:
   {
      OFF[dvTP,1]
      OFF[dvTP,2]
      OFF[dvTP,3]
      OFF[dvTP,4]
      OFF[dvTP,5]
   }
}

BUTTON_EVENT[dvTP,9] //SMALL/LARGE ICON DEMO MODE
{
   PUSH:
   {
   [dvTP,9] = ![dvTP,9]
   nLargeIcon = [dvTP,9]
   }
}
```

2. Save changes.

The NetLinx code shown above is included in the NetLinx Studio Workspace file (AdvancedDragAndDropExample.apw) that is in the Drag and Drop Demo.ZIP file.

8) Use NetLinx Studio 4 to Compile and Transfer the Project Files

Use NetLinx Studio 4 to compile the code and transfer the project files to the Master:

1. At the top of the AdvancedDragAndDropExample.axs source code file, change the ***dvTP*** value to match the device number of your touch panel (FIG. 74):

```
PROGRAM_NAME='MASTER'

DEFINE_DEVICE
dvTP = 10001:1:0
```

FIG. 74 dvTP Device Number value - Change to match the device number of your Touch Panel

2. Compile the code (select **Build > Build Active System**).
3. Transfer the *DragAndDropNoGroups.apw* workspace file to the NetLinx Master:
   a. Select **Tools > File Transfer** to open the *File Transfer* dialog.
   b. Open the *Send* tab and clear any files that are listed by clicking **Remove All**.
   c. Click **Add** to open the *Select Files for File Transfer* dialog.
   d. Select the top-level **Projects** folder to select all files in the workspace for transfer (FIG. 75):

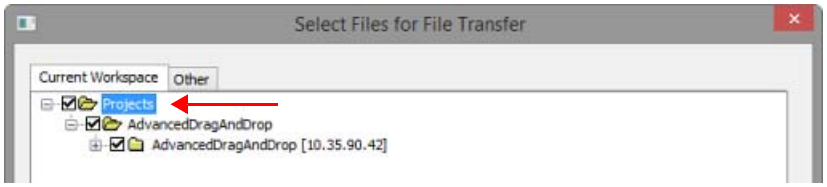


FIG. 75 Select Files for File Transfer dialog

   e. Select **OK** to return to the *File Transfer* dialog (FIG. 76):

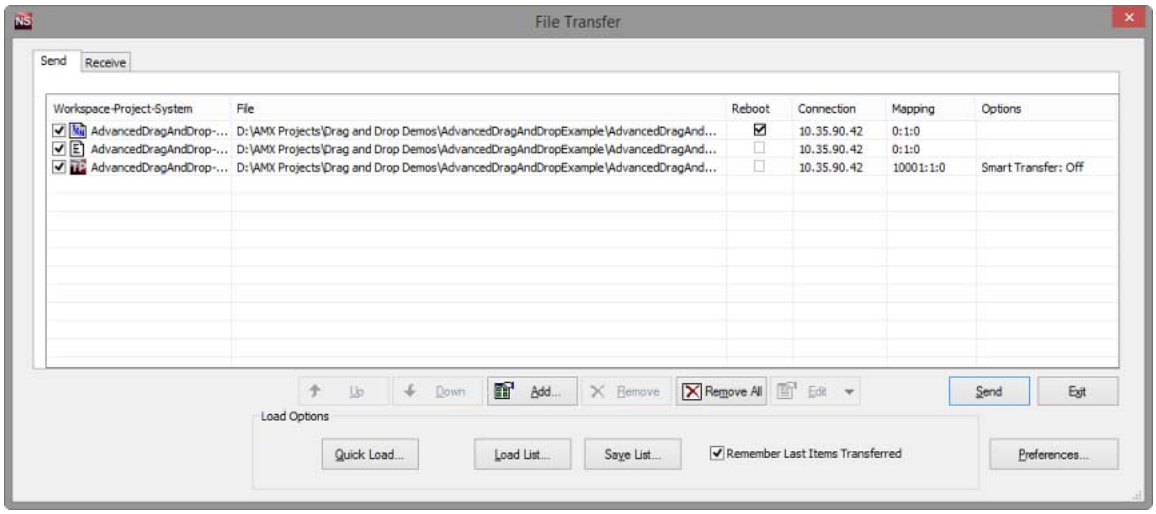


FIG. 76 File Transfer dialog - indicating files in the DragAndDropNoGroups.apw workspace queued for transfer

   f. Click **Send** to initiate the file transfer.
   g. The progress of the transfer is indicated in the Output Bar.

End Result

The result of this demo is a touch panel page with five draggable buttons representing source (input) devices and three drop target buttons representing output devices (Displays):

FIG. 77 Drag and Drop Demo - Page Layout

- The LEFT DISPLAY, CENTER DISPLAY and RIGHT DISPLAY buttons are drop target buttons representing three output devices (Displays) that can potentially accept the sources represented by the five draggable buttons as inputs:
  - The ENZO and iPAD draggable (source) buttons are configured to use LEFT DISPLAY and CENTER DISPLAY as valid drop targets.
  - The LAPTOP draggable (source) button is configured to use CENTER DISPLAY (only) as valid drop target.
  - The COMPUTER and HDMI draggable (source) buttons are configured to use CENTER DISPLAY and RIGHT DISPLAY as valid drop targets.
- These buttons can each be dragged onto the drop target buttons individually. When one of the draggable buttons is released within the bounds of a drop target, NetLinx code receives the custom events and makes visual changes to reflect the validity of the drag. If the drop target is valid for the selected draggable button, the target Display switches to the source represented by the draggable button that was dropped.

  Note that the LEFT DISPLAY (drop target) button indicates that it is a *valid* drop target for the ENZO and iPAD draggable buttons (FIG. 78):

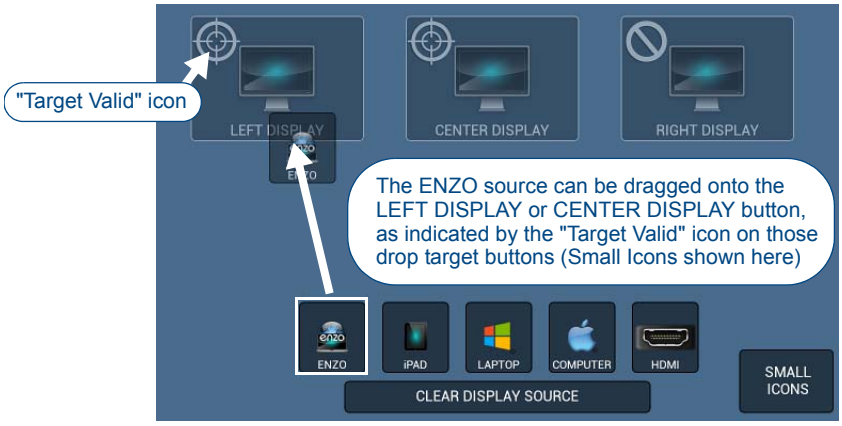


FIG. 78 Drag and Drop Demo - Dragging the COMPUTER Source button onto the LEFT DISPLAY Drop Target button

When the ENZO button is released, the LEFT DISPLAY uses it as it's new input. Note that the currently selected input is indicated with a highlighted source button (FIG. 79):

FIG. 79 Drag and Drop Demo - COMPUTER is the currently selected Source for the LEFT DISPLAY

- The LAPTOP draggable button is configured such that the LEFT DISPLAY button is an *invalid* target, therefore this buttons cannot be released on the LEFT DISPLAY drop target button.

  Note that the LEFT DISPLAY and RIGHT DISPLAY drop targets indicates that they are both in*valid* drop targets for the LAPTOP draggable button (FIG. 80):

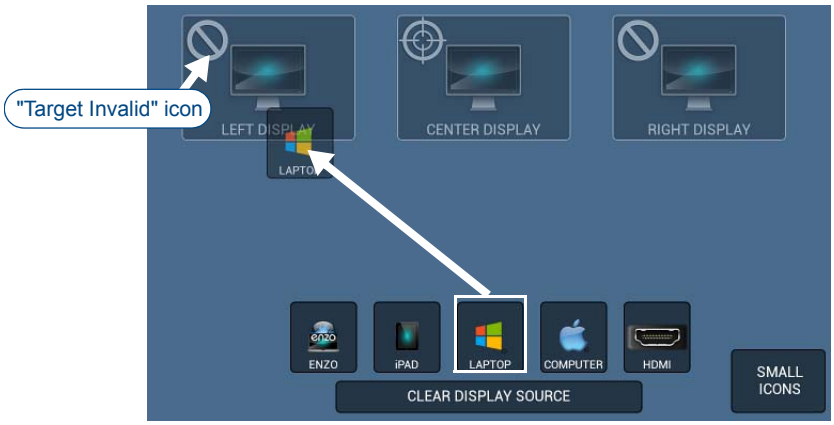


FIG. 80 Drag and Drop Demo - iPAD not allowed as a Source for the LEFT DISPLAY

- The CLEAR VTC SOURCE button will clear the current input setting on all displays when pressed.
- The SMALL ICONS button will toggle the "valid target" and "invalid target" icons (on the Display buttons) from small icons to large icons (FIG. 81):

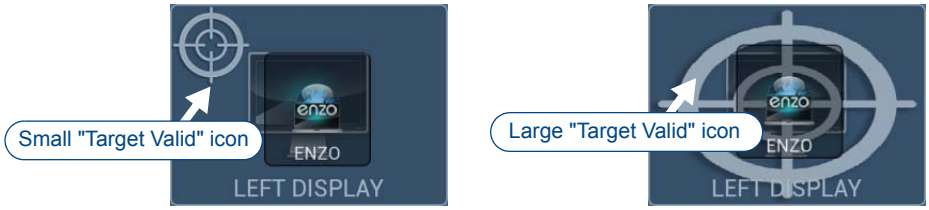


FIG. 81 Target Valid Icons (Small/Large)

Increase Your Revenue through education + knowledge

In the ever-changing AV industry, continual education is key to success. AMX University is dedicated to ensuring that you have the opportunity to gather the information and experience you need to deliver strong AMX solutions. Plus, AMX courses also help you earn CEDIA, NSCA, InfoComm, and AMX continuing education units (CEUs).

Visit AMX University online for 24/7/365 access to:

- *Schedules and registration for any AMX University course*
- *Travel and hotel information*
- *Your individual certification requirements and progress*

3000 RESEARCH DRIVE, RICHARDSON, TX 75082 USA • 800.222.0193 • 469.624.8000 • 469-624-7153 fax • 800.932.6993 technical support • www.amx.com