Exhibit Control
ENGINEERING

Datatcon Watchout NetLinx Control Module

As of 30 August 2010

Exhibit Control Engineering
102 Waterview Circle
Forest, VA 24551
(434) 385-4144
email: exhibit_control@yahoo.com
WEB: www.exhibit-control.net

Exhibit Control
ENGINEERING

Technical Notes on Using ece's WatchOut Netlinx Module

OVERVIEW

The Netlinx module, 'Watchout Module, ece, RevO', was developed to control a PC running WatchOut software by Dataton. It is designed to use channel activation rather than string parsing and can perform most of the Watchout Display Cluster Protocol. There are two master programs; one is for serial control via RS232 and the other for TCPIP control over Ethernet. Use the Master Program, 'WatchOut Test Master, RS232 Ver, ece, Rev 0', for the RS232 control demo and the file, 'WatchOut Test wTCPIP,FGP,RevO', for Ethernet operations. Which should you use for your application?

We generally prefer serial control via RS232 over Ethernet for several reasons: first it has fewer failure nodes in the communications hardware (just the two serial ports and one comm line versus several wires (Cat5/6) connections, hubs/switches (and their power supplies) and characteristics of the Network and traffic). Secondly, TCPIP communications are more difficult to trouble shoot than hardwired RS232. But, to fully use all of Watchout's features you really need two communications channels (either one RS232 and one Ethernet port or two different Ethernet ports). The specific feature that needs the two lines of communication is what we call the "pyro" message feature. This feature lets the Watchout program send out serial messages tied to specific times in the show timeline. If you need this capability, you can consider designing the entire system in the Ethernet environment. However, we still suggest using RS232 for control and Ethernet for pyro commands.

Here are some thoughts on using pyro commands in your system. We have several media producers who like to have the flexibility to move when theatrical (pyro or other) effects happen within their Watchout program. The pyro command feature works well for many show applications. However, it is not as precise as something using SMPTE or MIDI timecode. Not even close. Where time code implies granularity down to the frame, this system’s limit is closer to three or four frames. But more importantly, it can only space events 0.30 seconds or greater apart. Generally, the system will not miss any pyro commands, even if they are very close together (approximately one tenth of a second apart. So it will produce all events associated with those cues, but it will space them out so they occur once every three tenths of a second. This means if there were nine cues very close together, there would be three seconds between when the first one occurred and when the last one occurred.

So here are some considerations to help improve the system’s performance. First, if there is a very precise cue you want to happen virtually at the same time for every show, ensure there are no other cues (or multiple cues) just prior to it. If you want a number of different events to happen at the same time, put all of the events in a single pyro cue. If you are just using pyro commands to tell the Master when the show and /or the attract section has ended, you can do this with CHANNEL_EVENTS. However, if you are using numerous pyro cues that are very close together in time, you will need more stringent management of CHANNEL_EVENTS.

Finally, if your system has a lot of stacked pyro cues, you need to manage the control of those events a little more tightly. To do this you will need to create a buffer variable to capture all of the pyro cue “channels” and then activate them each 0.3 seconds apart. If you do not normally create a “Mainline” Timeline, you will need to create a repeating Timeline that occurs every 0.3 seconds. If you already create a “Mainline” Timeline, make sure its period is 300 (300 milliseconds). In this Timeline you will need to check if there are any pulses in the “buffer” and execute them. See Appendix A for sample code.

The main principle of operation for this module (like all modules developed by ECE) is most operations are accomplished by pulsing a specific channel on the virtual device (vdvSHOW_PC) and getting feedback on the virtual feedback device (vdvSHOW_FB). The module is the only thing that actually talks to the real device (dvSHOW_PC). Many times, a variable must be set just prior to the pulse action to take advantage of all of Watchout’s capabilities.

We should note here, that generally the module only interacts with the Watchout Program, not with the PC (other than WOL). So the shortcut to the WatchPointe program should be in the “Windows” start-up menu so that it automatically starts up with every boot process. This also means that any problems stemming from the PC (like serial port locked-up or in-use, Watchpointe is not running or is stopped and restarted in a session, or any other “Windows” problems) may require that the PC be manually powered down and a manual forcing the Module to a “PC no power state” and starting the initiation process all over again to get everything re-synchronized. This is just the hazard of using “Windows” based platforms and only controlling the program running on it. This problem will usually be indicated by the kREADY channel not being on, even though everything else seems OK.

The next section covers all the parameters passed from the master program to the module.

PARAMETERS TO BE PASSED TO THE MODULE

DEV vdvSHOW_PC	-	The virtual Watchout PC
DEV dvSHOW_PC	-	The real device that talks to the PC (RS232 or TCPIP connection)
DEV vdvSHOW_FB	-	A virtual device for obtaining channel feedback.
DEV dvUTP	-	This is a device on the master (0:??:1) which will communicate the Wake On Lan (WOL) and through which the Pyro messages will be received. (Port 5000 for WOL and an arbitrary port you select (not 3039) for the pyro messages).

DEV dvTP	-	If you want to see the strings going and coming from the PC, this should be the touch panel where the variable text boxes are located (they are designated 1 & 2). Otherwise, define this as a dummy device number.
CHAR sCUE_NAME[20]	-	This variable holds the Named Cue you want to jump to and should be loaded just prior to pulsing the GO2CONTROL_CUE channel (7).
CHAR sSHOW_FILE[20]	-	This variable will hold the name of the show file you want to load and use, and should be loaded with the correct file just before pulsing the load channel (5).
INTEGER nPYRO_PORT	-	TCPIP port for Pyro messages (do not use 3039).
CHAR sUSER_DEFINED[20][20]	-	This variable holds an array of strings which are arbitrary (with the limit of 20 bytes). Have the producer of the Watchout show embed these strings, preceded with colons and terminated with carriage returns ($0D or 0X0D) in the Watchout timeline. When the string is fired out, Netlinx will pulse the corresponding channel. Simply write a channel-on event to do something when this happens. Common use is to put a cue at the end of the show, so that Netlinx knows when the show is finished.

LONG lSHOW_TIME	-	Load this variable just before using the GO2TIME channel (6). This will be the time the Watchout show jumps to. The number is milleseconds.
CHAR sSMPTE_TIME[12]	-	This is one of two variables that should be set just prior to using the TIMECODE_ON channel (13). It is the offset for the time code in the following format: 'HH:MM:SS.FFF' where FFF is milliseconds.
CHAR sARN_PARAMETER[25]	-	General purpose parameter that can be used with SET_STRING, channel (12).
INTEGER nSEQ_NUMBER	-	Each Watchout Master PC in the Netlinx Master program should have a unique sequence number between the total number of WatchOuts minus one to 0 which should be the first one.
INTEGER nNUMBER_OF_PCS	-	The number of Watchout PCs that are in the synchronized show.
CHAR sIP_ADDRESS[15]	-	IP address of WatchOut PC.
INTEGER nMAC_ADDR[][6]	-	This is an integer array that holds all of the MAC addresses for all the display Watchout PCs for one specific synchronized show. NOTE: There needs to be a module instantiated for every Watchout show which is not necessarily one for every Watchout PC.

INTEGER nTIME_CODE_MODE	-	This is the other variable that needs to be set just prior to using TIMECODE_ON channel (13). It should be set to one of the following: 0 - SMPTE receiver off 1 - auto detect format 2 - EBU 25fps 3 - SMPTE 29.97 NDF 4 - SMPTE 29.97 DF 5 - SMPTE 30 (“B&W”)
INTEGER nCONDITION	-	This variable needs to be set, just prior to load, channel (5), and ENABLE_LAYER, channel (8), whenever this feature is to be used. If the feature is not used, it should be zero.
INTEGER nFADE	-	Set this variable just before using the STANDBY_ON, channel (9), and STANDBY_OFF, channel (10) to lengthen the fade of these two actions. The default is zero.

Next we will look at all of the channel assignments used in this module.

CHANNEL CONSTANTS

```
(*     COMMANDS                          *)
kRUN                   =       1;
kHALT                  =       2;
kPING                  =       3;
kAUTHENTICATE          =       4;
kLOAD                  =       5;
kGO2TIME               =       6;
kGO2CONTROL_CUE        =       7;
kSTANDBY_ON            =       9;
kSTANDBY_OFF           =      10;
kGET_STATUS            =      11;
```

```
kSET_STRING              =      12;
kTIMECODE_ON             =      13;
kTIMECODE_OFF            =      14;
kENABLE_LAYER_COND       =      15;

kON                      =      27;
kOFF                     =      28;
KPWR_APPLED              =      29;  //  TURN ON TO SIGNIFY THE
                                      //  PC HAS 120vac POWER AND
                                      //  OFF WHEN IT DOESN'T.
(* STATUS                                            *)
kREADY                   =     101;
kBUSY                    =     102;
kSTANDBY                 =     110;
kERROR                   =     111; // GENERAL ERROR
kSHOW_FILE_ERROR         =     112; // FILE ERROR
kCONNECTED               =     150;

(* USER DEFINED CHANNELS    *)
KUSE_PYRO_SIGNALS        =     200; // 0 – NOT USED; 1 - USE
kUSER_1                  =     201; // THESE CHANNELS WILL PULSE
kUSER_2                  =     202; // FOR 0.5 SEC EVERY TIME
kUSER_3                  =     203; // THEIR PYRO MESSAGE IS
kUSER_4                  =     204; // RECEIVED.
kUSER_5                  =     205;
kUSER_6                  =     206;
kUSER_7                  =     207;
kUSER_8                  =     208;
kUSER_9                  =     209;
kUSER_10                 =     210;
kUSER_11                 =     211;
kUSER_12                 =     212;
kUSER_13                 =     213;
kUSER_14                 =     214;
kUSER_15                 =     215;
kUSER_16                 =     216;
kUSER_17                 =     217;
kUSER_18                 =     218;
kUSER_19                 =     219;
kUSER_20                 =     220;
kCOM_MODE                =     250; //  OFF – RS232; ON – TCPIP
```

```
KPYRO_PORT_CONNECTED     =     251;
kINOP                    =     252;
kDEBUG                   =     254; // NEEDS TO BE ON TO GET
                                    // MESSAGES INTO TOUCH PANEL
                                    // TEXT BOXES.
```

NOTES ON CHANNEL PARTICULARS

Many of the channels' functions are fairly intuitive, but some are not. This section explains the peculiarities of those that aren't intuitive. The following channels perform the Watchout function in their name and do not need a variable set before initiating them: kRUN, kHALT (stop), kPING, kSTANDBY_ON (video black on), kSTANDBY_OFF (video black off), kGET_STATUS and kUSE_PYRO_SIGNALS. Although the kSTANDBY_?? channels do not need a variable set, if you want to adjust the time it takes them to go from one full state to the other you will need to set the variable nFADE in milliseconds (i.e. 5000 is 5.0 seconds). Turn the kUSE_PYRO_SIGNALS channel on if you want NetLinx to look for these messages from the Watchout Program. Turning this channel on will initiate communications on the pyro port.

The kAUTHENTICATE channel is only used with TCPIP control and will be explained later. kLOAD is used to load a show which must be done first after establishing communications, as none of the other commands will work until there is a show loaded. The show name should be twenty characters or less and does not include its file extension ".watch". You should allow at least two seconds after a load command, maybe more if the files are very large or there are many show files on the Watchout PC, before you issue any of the operational commands.

The kGO2TIME and kGO2CONTROL_CUE are similar in function. They both cause the WatchOut show to jump to a new location. The kGO2TIME does not require anything embedded in the show; just set the variable lSHOW_TIME in milliseconds (i.e. 5000 is equal to 5.0 seconds) and pulse the channel and the video will move directly to that time in the show. If the show is running, it will continue to run from that point in the timeline. If the show is halted, it will freeze on that frame. The kGO2CONTROL_CUE is similar but it requires a cue name embedded in the show. Set the variable sCUE_NAME to the correct string and pulse the channel. It will jump the video to that spot in the timeline; if running, it will continue to run or if halted it will freeze frame there.

To understand the kTIMECODE_ON and kTIMECODE_OFF review their actions in the Watchout Manual. Their variables to set are nTIME_CODE_MODE and sSMPTE_TIME.

In Watchout you can identify layers as conditional by giving them a number greater than zero. Each conditional layer will have a sequence number: 1,2,3,4…… Think of each conditional layer as a bit in a digital word. Layer one is the first bit, layer two the second bit, layer three the third bit, etc. So if you wanted just conditional layer "one" only on, you would set nCONDITION equal to 1 (zero would be for no conditional layers on). If you wanted just layer "two" on, nCONDITION would be set to 2. However, if you only wanted layer "three" on you would set nCONDITION to 4. If you set it to 3, that would turn both layers one and two on and no others. This feature is most often used to turn closed captioning on and off on the fly while the show is running.

Before we talk about kON and kOFF, we need to talk about kPWR_APPLED. Neither the NetLinx Master nor the module knows when the Watchout PC has power applied to it. The module is written so that it can only do things incrementally as the state of the PC progresses from no power applied, power applied, boot-up, Windows boot-up and the Watchout Program is running. We use this channel to let NetLinx and the Module know when there is power applied to the PC. We could set this channel equal to a relay that controls power to the PCs, or turn it on and off in the program based on other information where we know the PCs will have power applied to them. If the PCs are always powered on (which is a fine solution) just set this channel on in DEFINE_START and leave it on all of the time..

Once we know the PC has power and the kPWR_APPLED channel is on, we can pulse the kON channel. This will cause the module to try to do a WOL for all the MAC addresses in the array nMAC_ADDR. This is a built in feature regardless if you are using TCPIP communications or not. So if you are not using TCPIP and are using some other method to start the PCs up (most likely have their BIOS set so they boot up when power is applied to them), we need to initiate that action before we pulse the channel kON.

Once kON is pulsed, the module will wait 60 seconds before allowing any other actions. When it is ready, the feedback channel “kON” will be on. If you notice that your Watchout PCs take longer than a one minute boot up into the WatchPointe program (which is unlikely) ensure you Master program waits the additional time before initiating further actions.

Once the system is up and operating (including connection and authentication if require – discussed elsewhere) you can now use the kOFF channel to gracefully power down the PC as long as the system has established communications and WatchPointe is running.

HOW TO INITIATE THE DEVICE AT SYSTEM POWER UP

1. Ensure all of these variables are instantiated in the DEFINE_START section. The data presented here is from the Demo master Programs:

```
sFILE_WFOCUS              =     "'Countdown59a'";
sCHANNEL_PROMPT[1]        =               "'10'";
sCHANNEL_PROMPT[2]        =               "'20'";
sCHANNEL_PROMPT[3]        =               "'30'";
sCHANNEL_PROMPT[4]        =               "'40'";
sCHANNEL_PROMPT[5]        =               "'50'";
sCHANNEL_PROMPT[6]        =              "'end'";
sWO_CUE                   =                "'2'";
lWO_TIME                  =                5000;
sSMPTE_TIME               =         "'-1:00:00'";
sARB_PARAMETER            =                  "";
nWO_NUMBER                =                   1;
nNUMBER_OF_PCS            =                   1;
sWO_IP_ADDRESS            =    "'192.168.1.111'";
nTIME_MODE_CODE           =                   3;
nENABLE_COND              =                   1;
nFADE_OUT                 =                5000;
nPYRO_MSG_PORT            =                6000;
nMACADDRESS_ARRAY[1][1]   =                 $00;
nMACADDRESS_ARRAY[1][2]   =                 $30;
nMACADDRESS_ARRAY[1][3]   =                 $1B;
nMACADDRESS_ARRAY[1][4]   =                 $48;
```

```
nMACADDRESS_ARRAY[1][5]    =                  $8A;
nMACADDRESS_ARRAY[1][6]    =                  $D1;
[vdvWATCHOUT_FB,kCOMM_MODE]=                    0;  // or 1
```

These variables will be passed to the module in the DEFINE_MODULE section.

2. Wait until the WatchOut PC has booted up. It boots up pretty quickly, but the module allows 60 seconds to be safe. Then PULSE[vdvWATCHOUT,kON], which sends out the WOL broadcast. If the Watchout PC and NetLinx are not networked together, manually power up the Watchout PC.

3. When using RS232 serial control to activate the Display Cluster Protocol, you must have a command file in the same file folder as the Watchout executable file. (See page 185 & 186 of the atchout Manual.) Create it in Notepad and save it as CMD.txt. You need <u>three</u> lines in the CMD.txt file and it is case sensitive:

   ```
   authenticate 1
   serialPort true “COM3”
   <empty line>
   ```

   In addition, the CMD.txt file needs to be referenced in the short cut to Watchpoint in the Start Up Folder. Example: <drive and path>/WatchPoint.exe CMD.txt

4. When using TCPIP to activate the Display Cluster Protocol, you must send an authentication command to the WatchOut PC you want as a Master. To do this, wait until the Watchout has booted-up which would normally be 60 seconds from pulsing [vdvWATCHOUT,kON] at which time [vdvWATCHOUT_FB,kON] should be equal to one. Then PULSE[vdvWATCHOUT,kAUTHENTICATE].

5. Then make sFILE_WFOCUS equal to the show file name you want to run and PULSE[vdvWATCHOUT,kLOAD] two (2) seconds after the authenticate. Then wait another two seconds (or more depending on the size of the file) and PULSE[vdvWATCHOUT,kRUN] action and stop with a PULSE[vdvWATCHOUT,kHALT] action or use any of the other show controls.

THE DEMONSTRATION PROGRAMS

With every licensed copy of the Watchout NetLinx Module, you will get the module .tko file, a G3 touch panel file, two demo Master files (one for RS232 and one for Ethernet), a sample CMD.txt file for RS232 operations and a sample Watchout Show which is basically a countdown from 60 seconds to zero. This file has embedded cues, a conditional layer which is a green rectangle behind the countdown display and pyro messages in it.

The Demo program uses touch panel button pushes to instigate all control of the Watchout program. However, in a real show control program you could also instigate actions based not only on button pushes but also by CHANNEL_EVENTS, DATA_EVENTS and TIMELINE_EVENTS. In the Demo program it was easier to explain the mechanics using button pushes.

The Watchout show program needs to be loaded on the display computer. This takes two computers with two Watchout licenses to accomplish. Loading a show is not covered in this manual but rather refer to the Watchout Manual. Ensure a short cut for the Watchpointe program is loaded in Start Programs folder. And, if you are trying the RS232 version, ensure the CMD.txt file is loaded in the executable file folder and is part of the target executable string in the Windows properties for the WatchPointe program. See pages 185 and 186 in the Watchout manual.

First set one of the master programs (RS232 or TCPIP) as the master in NetLinx Studio and load it into the Controller. Next using TP Design 3 save the touch panel file as HTML and FTP it to the Controller. In the HTML Conversion Options Window, ensure that HTML type is NetLinx and the base address is 255.

In a WEB browser, open the IP address of the Master which should return a screen like Figure 1.

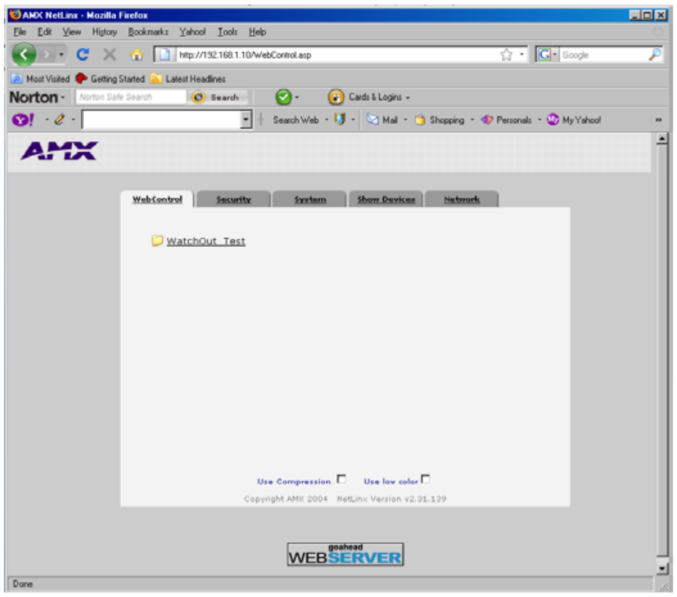


Figure 1.

Click on the File folder Watchout_Test and after a short time you should get the touch panel splash page as seen in Figure 2. To achieve this operation you will need Java loaded on your PC.

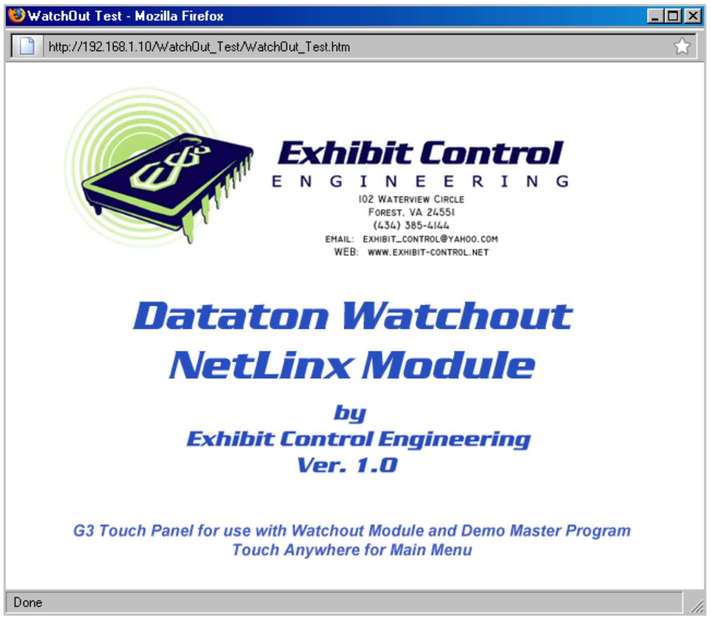
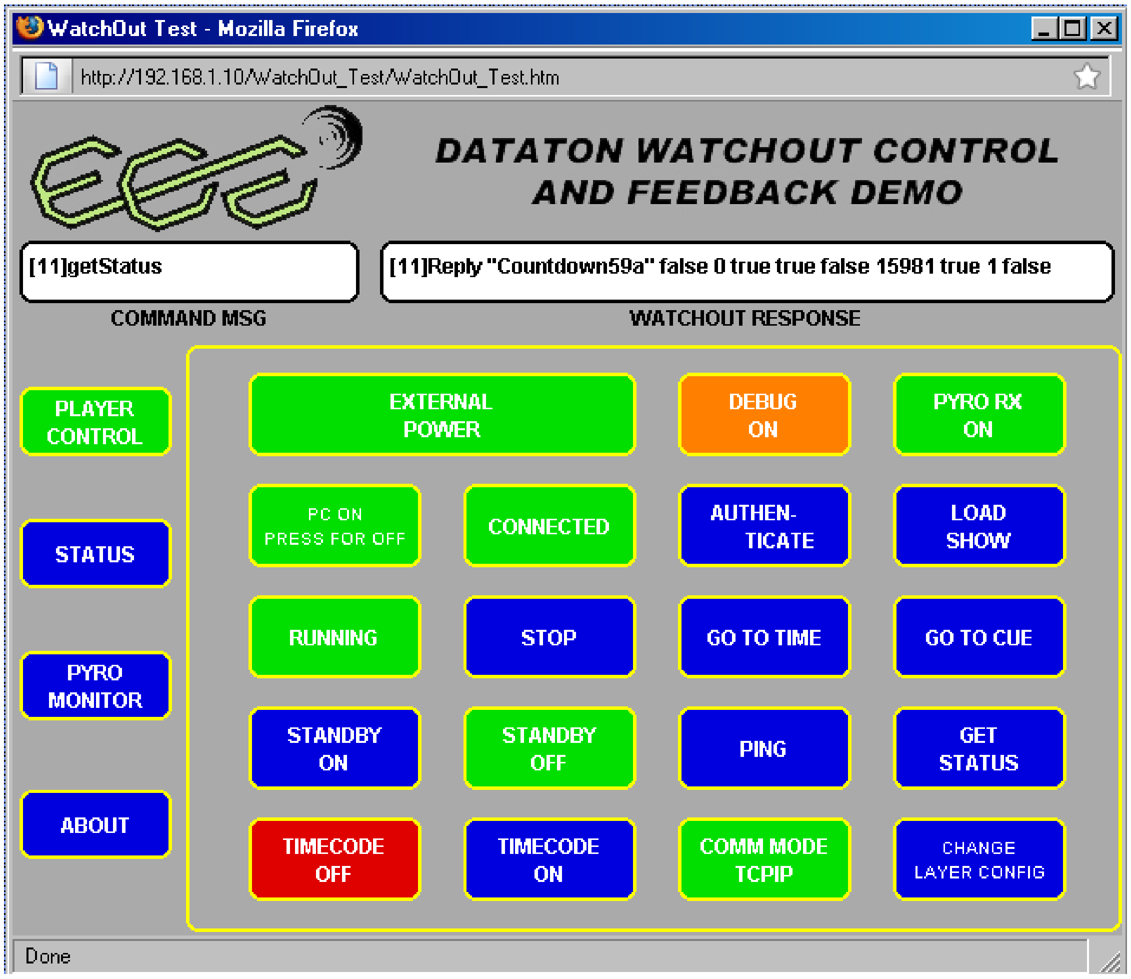

WatchOut Test - Mozilla Firefox
http://192.168.1.10/WatchOut_Test/WatchOut_Test.htm
Exhibit Control
ENGINEERING
102 Waterview Circle
Forest, VA 24551
(434) 385-4144
EMAIL: EXHIBIT_CONTROL@YAHOO.COM
WEB: WWW.EXHIBIT-CONTROL.NET
Dataton Watchout
NetLinx Module
by
Exhibit Control Engineering
Ver. 1.0
G3 Touch Panel for use with Watchout Module and Demo Master Program
Touch Anywhere for Main Menu
Done

Figure 2.

Click anywhere on this screen to get to the Main Menu as seen in Figure 3.



[11]getStatus

[11]Reply "Countdown59a" false 0 true true false 15981 true 1 false

Figure 3.

If there is no pop up page, press the button that says "Player Control".

During our testing of the module we discovered a flaw in the Watchout PC's WOL process. If power has been removed from the PC or it was last shut down with the power button on the PC case, the network card will not see the WOL string. So the very first time the PC is used after the above conditions the WOL will not work. We believe Dataton is going to correct this issue, but we are not sure when. So the first time you use this Demo we suggest you power the PC up with the PC power button and then do a graceful power down using the "Windows" start menu. Thus we should have the PC with 120vac power applied and in an off condition after a graceful shut down.

First thing we need to do is press the double wide “External Power” button on the touch panel so it is green which tells the Master that the PC has power. Next press the button that says “PC OFF PRESS FOR ON”. Nothing will happen on the touch panel, but if the IP address and MAC address data are correct in the Master program, the Watchou PC should start to boot up. This assumes the Watchout PC and the NetLinx master are networked together. Sixty seconds later that button should turn green and by that time the black Watchout logo screen should be on up on the video. If you want to see the communications between the Controller and the PC, press the “DEBUG” button to on.

The next steps are different depending if you are using RS232 or TCPIP. In Ethernet Mode (TCPIP) once the PC is booted up to Watchpointe we still do not have communications, so you must click the “CONNECT” button which should cause immediate connection if all IP addresses are correct and everything is on the same network. On the other hand, if you are using RS232, the “CONNECT” button will be highlighted the same time as the “PC ON” button because the RS232 connection is always there. Having said that, remember that any problem with the PC “comm” port will not be detected by NetLinx.

Once the connect issue is correctly resolved, the next step is also communication specific. If using TCPIP communications, we must “authenticate” by clicking on that button on the touch panel. For the RS232 mode we can skip this step because it is taken care of in the CMD.txt file on the PC.

The last step is to load the program. The show name was hard coded into the Master programs and does not have to be set. But, in general for the purposes of our Demo, we suggest you open a debug window and load all the variables you might want to change in the monitor window. Then you can change them to whatever you want just prior to a button push.

If you click on the touch panel “STATUS” button you will see a screen similar to Figure 4 pop up.

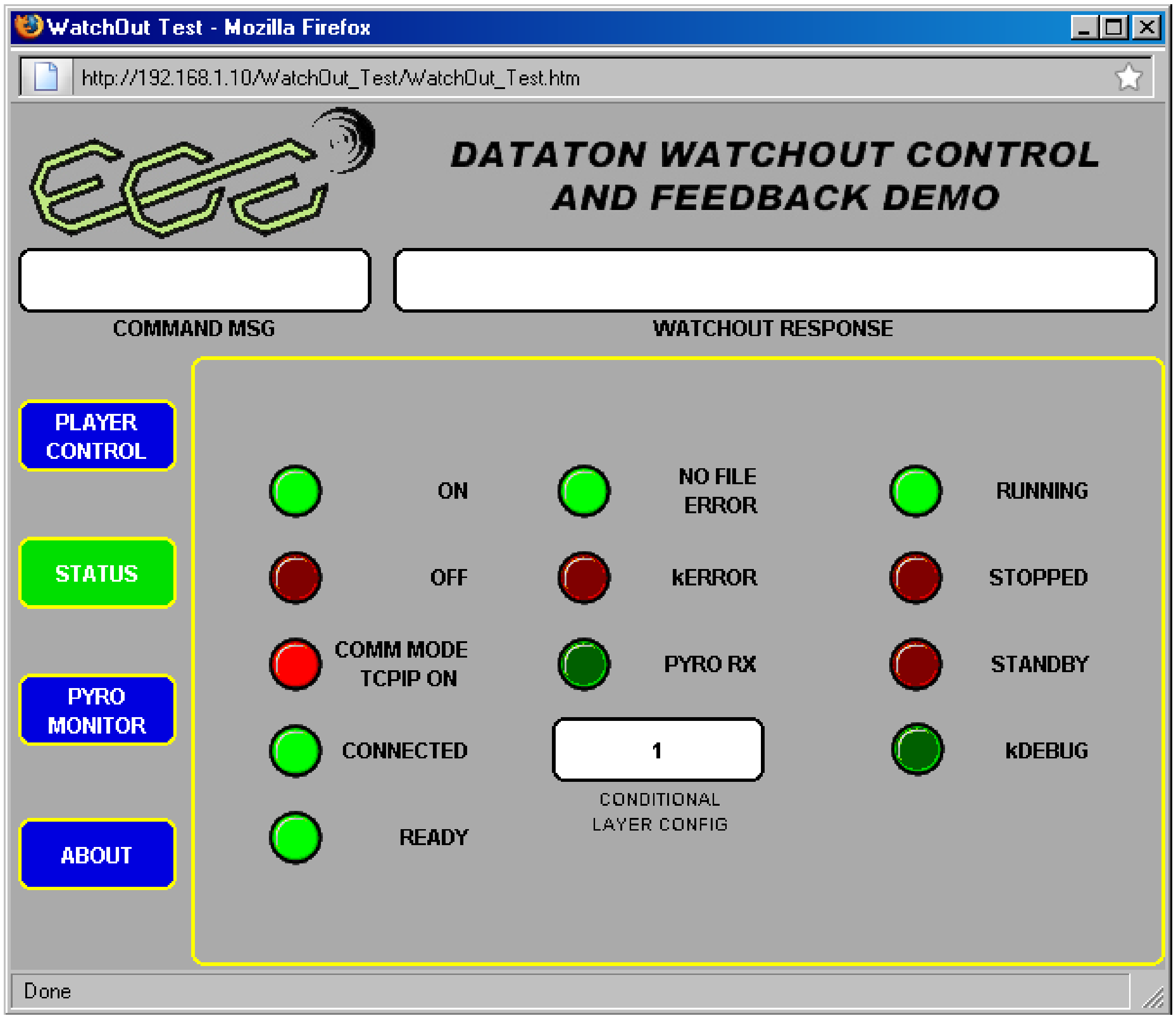


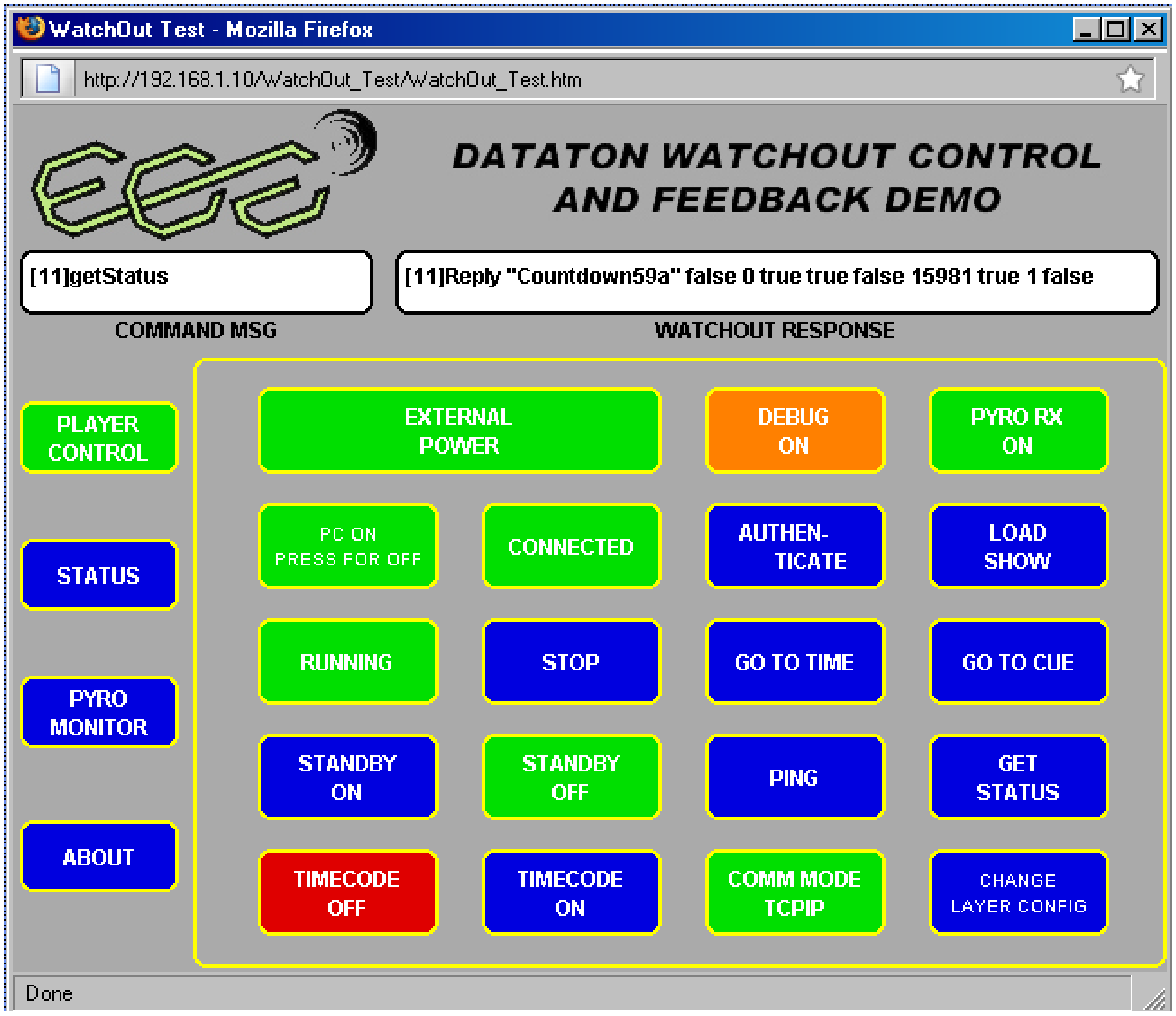
Figure 4.

In order for the program to work correctly the following LEDs on this page should reflect the following states:

- ON LED	-	on green
- OFF LED	-	off red
- CONNECTED LED	-	on green
- READY LED	-	on green
- NO FILE ERROR	-	on green
- kERROR	-	off red
-		

The other LEDs should indicate the state of the program: running or stopped; standby (video blacok on or off); comm Mode; whether pyro receive is on or off and, finally, display the number representing the conditional layers that are on.

Let's go back to the "Player Control" menu.



WatchOut Test - Mozilla Firefox

http://192.168.1.10/WatchOut_Test/WatchOut_Test.htm

DATATON WATCHOUT CONTROL AND FEEDBACK DEMO

COMMAND MSG: [11]getStatus

WATCHOUT RESPONSE: [11]Reply "Countdown59a" false 0 true true false 15981 true 1 false

Figure 3(Again). 1

If you have "DEBUG" turned on, all communications between the NetLinx Master and the Watchout PC will be displayed. Clicking the "RUN" and "STOP" buttons will cause the show to play or pause based on the button. If you press the "GO TO TIME" button, the show will go to the time hardcoded in the program which is 5.0 seconds (5000 millisec). The video screen should jump to approximately "55" in the countdown. You can change this time in NetLinx Studio in debug watching the variable lSHOW_TIME before pressing the button again to change the jump time. Similarly, if you press the "GO TO CUE" it will jump the video to the hardcoded cue “2”. You can change the variable sCUE_NAME to “1”, “3”, “4” or “5” to get this button to jumpt to other embedded cues in the Watchout Show program.

If you press the “STANDBY ON” button, you will watch the video fade from full up to full black in 5.0 seconds (5000 milliseconds) which is hard coded into the Master program. You can change this fade time by changing the variable nFADE to some other number. Conversely, pressing the “STANDBY OFF” button will cause the reverse action fade up from black to full screen in nFADE time.

Pressing the “TIME CODE ON” and “TIME CODE OFF” buttons will cause that action to take place, but with no other equipment you will not realize any change has taken place. See the Watchout manual for additional information.

Pressing the “PING” and “STATUS” buttons will casue those actions to occur. You can watch the results in the message text windows, if “DEBUG” is on.

If you change the variable nCONDITION from one to zero in NetLinx Studio and press the “CHANGE LAYER CONDITION” button, you will see the green rectangle behind the countdown disappear from the screen. Changing it back to one and hitting the button will cause the green rectangle to reappear.

If the Watchout PC and the NetLinx master are networked together, you can press the “PYRO Rx” button to on and see what its affect is. Press the Main Menu button on the left labeled “PYRO MONITOR” and you will see a screen similar to Figure 5.

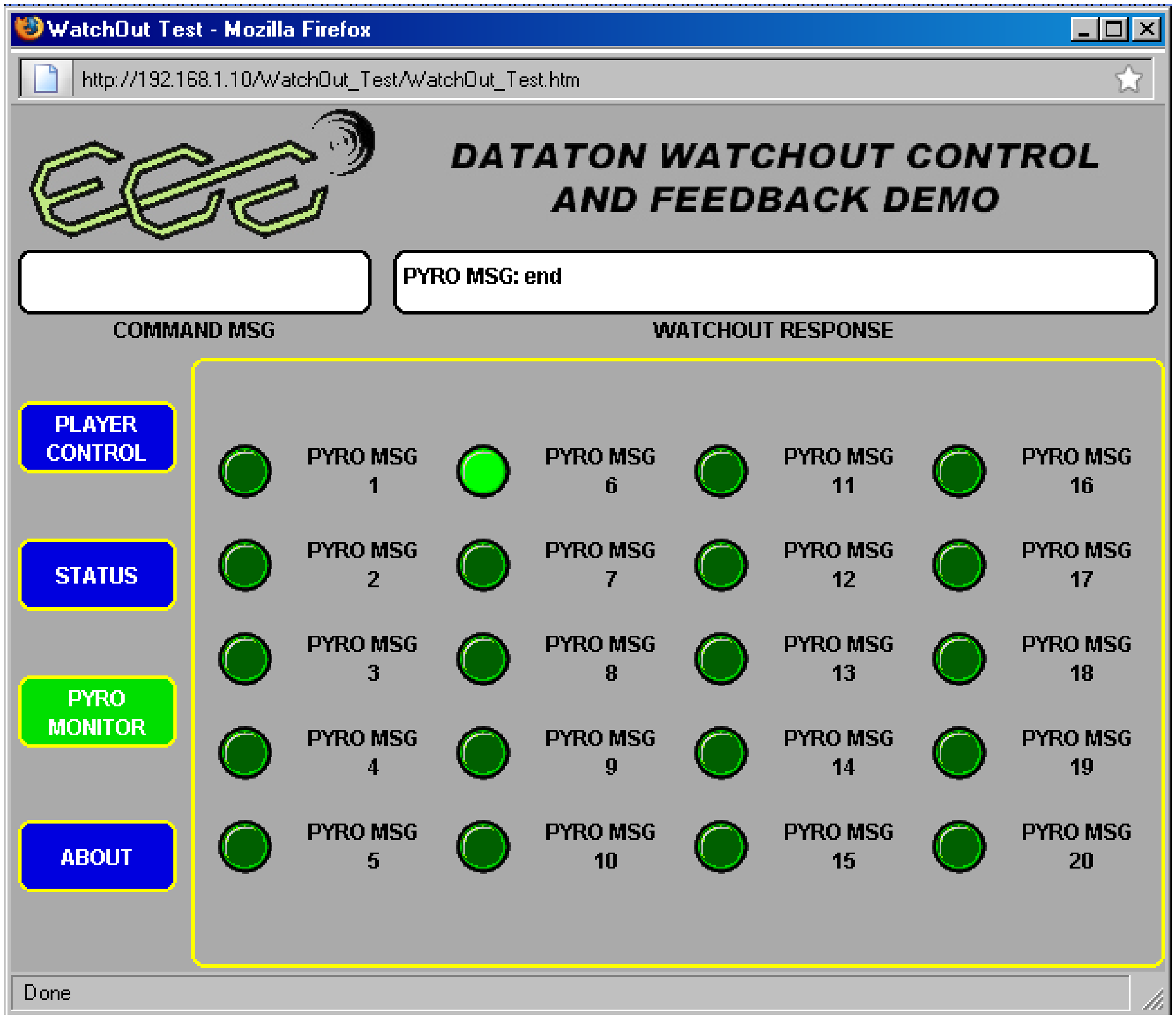


Figure 5.

Every time the NetLinx master receives a pyro message from the Watchout show, you will see its corresponding LED blink and the actual message in the Watchout Response window. The pyro messages embedded in the Watchout show are: “'10',$0D”; “'20',$0D”; “'30',$0D”; “'40',$0D”; “'50',$0D”; and, “'end',$0D”.

From here on out, you need to develop a master program to do what you want your show to do. Good luck from Exhibit Control Engineering.

APPENDIX A

POSITIVE CONTROL OF PYRO EVENTS

In Define Constants section add a LONG for the Mainline Timeline (if you do not already have it):

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

/* GENERAL CONSTANTS                                       */
CONSTANT LONG kMAINLINE_TL                  =          1;
```

In the Define Variables portion of the Master code include a variable for the “buffer” and a variable for the period of Mainline Timeline, if not already in your code. If it is in your code, ensure it is set to 250 (250 milliseconds):

```
(***********************************************************)
(*               VARIABLE DEFINITIONS GO BELOW             *)
(***********************************************************)

VOLATILE CHAR sPYRO_HIT_BUFFER[20];
VOLATILE LONG lMAINLINE_PERIOD[1]  =  250;
```

Build a function to process the pyro events one at a time:

```
(***********************************************************)
(*        SUBROUTINE/FUNCTION DEFINITIONS GO BELOW         *)
(***********************************************************)

DEFINE_FUNCTION PyroAction()
{
      STACK_VAR CHAR cWHICH_PYRO;
      STACK_VAR INTEGER nWHICH_PYRO;

      IF(LENGTH_STRING(sPYRO_HIT_BUFFER))
      {
            cWHICH_PYRO = GET_BUFFER_CHAR(sPYRO_HIT_BUFFER);

            IF([vdvWATCHOUT_PC_FB,kPYRO_DEBUG])
            {
                  nWHICH_PYRO = cWHICH_PYRO;
                  SEND_STRING 0,"'PYRO SHOW SEQUENCE = ‘,ITOA(nPYRO_SEQUENCE_PLACE)";
                  SEND_STRING 0,"'PYRO HIT ON CHANNEL ',ITOA(nWHICH_PYRO)";
            }
```

```
SWITCH (cWHICH_PYRO)
{
        CASE 1:
        {
                //  SOME ACTION
        }
        CASE 2:
        {
                //  SOME ACTION
        }
        CASE 3:
        {
                //  SOME ACTION
        }
        CASE 4:
        {
                //  SOME ACTION
        }
        CASE 5:
        {
                //  SOME ACTION
        }
        CASE 6:
        {
                //  SOME ACTION
        }
        CASE 7:
        {
                //  SOME ACTION
        }
        CASE 8:
        {
                //  SOME ACTION
        }
        CASE 9:
        {
                //  SOME ACTION
        }
        CASE 10:
        {
                //  SOME ACTION
        }
        CASE 11:
        {
                //  SOME ACTION
        }
        CASE 12:
        {
                //  SOME ACTION
        }
        CASE 13:
        {
                //  SOME ACTION
        }
        CASE 14:
        {
                //  SOME ACTION
        }
```

```
                    CASE 15:
                    {
                          //  SOME ACTION
                    }
                    CASE 16:
                    {
                          //  SOME ACTION
                    }
                    CASE 17:
                    {
                          //  SOME ACTION
                    }
                    CASE 18:
                    {
                          //  SOME ACTION
                    }
                    CASE 19:
                    {
                          //  SOME ACTION
                    }
                    CASE 20:
                    {
                          //  SOME ACTION
                    }

              }
        }
}
```

In the Startup Section, start the Mainline Timeline, if not already in your code:

```
(***********************************************************)
(*                STARTUP CODE GOES BELOW                  *)
(***********************************************************)

WAIT 120
{
       TIMELINE_CREATE(kMAINLINE_TL,lMAINLINE_PERIOD,1,TIMELINE_ABSOLUTE,TIMELINE_REPEAT);
}
```

In the Events Section of the Code have CHANNEL_EVENTS for the pyro channels:

```
///////////////////////////////////////////////////////////
////     WATCHOUT PYRO CHANNEL EVENTS                  ////
///////////////////////////////////////////////////////////
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,201]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,202]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,203]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,204]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,205]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,206]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,207]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,208]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,209]
```

```
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,210]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,211]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,212]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,213]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,214]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,215]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,216]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,217]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,218]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,219]
CHANNEL_EVENT[vdvWATCHOUT_PC_FB,220]
{
	ON:
	{
		STACK_VAR INTEGER nPYRO_MSG_NO;

		nPYRO_MSG_NO = CHANNEL.CHANNEL - 200;

		sPYRO_HIT_BUFFER = "sPYRO_HIT_BUFFER,nPYRO_MSG_NO";
		PyroAction();
	}
}
```

In the Events section of the code define your MainLine Timeline, if you don’t already have one:

```
////////////////////////////////////////////////////////////
////     TIMELINE EVENTS                                ////
////////////////////////////////////////////////////////////

TIMELINE_EVENT[kMAINLINE_TL]
{
	IF(LENGTH_STRING(sPYRO_HIT_BUFFER))
	{
		PyroAction();
	}
}
```