# DOFLinx Guide for V8.12

## What is DOFLinx

DOFLinx is a utility program designed to work with the DOF (Direct Output Framework) R3 (from version 6.20 onward this is the R3++ version of DOF) to provide some additional functionality for pinball emulators that don't do it directly. It started life just adding flipper force feedback to Pinball FX2 and was called DOFFX2.

Some of the features are:

- Basic Fire flipper solenoids / contactors from with Pinball FX2, Pinball FX3, Pinball Arcade, etc using keyboard intercept
- Full force feedback for Pinball FX2 and FX3 for games that have been memory mapped
- Link to Future Pinball and process messages to enable fully scripted force feedback
- Add flipper force feedback and RGB easily to tables with no force feedback coded
- Have RGB devices automatically cycle colours based on a variety of triggers
- Link a key to turn on one or more outputs while pressed
- Link a key to set an RGB device to a colour
- Link a key to turn an RGB device to a colour while pressed then revert back
- Game specific colour sets
- Control your LEDs and solenoids from various places, just for fun
- Respond to messages sent from other scripts or applications to trigger outputs and colours
- Settings menu to control relays and other devices in your cabinet that turn on / off features
- Link to B2S backglasses from Pinball FX2 and FX3
- Drive addressable LEDs setup in the DOF from Pinball FX2 and FX3 and Future Pinball triggers
- Run PUPlayer videos from FP and FX2 / 3
- Produce nudge into FX3 from Pinscape accelerometer
- Support for XBox (XINPUT) and Joystick (DINPUT) controllers
- Joystick based analogue joystick to trigger FX2 and FX3 ball launch
- Drive Xbox rumble motors from table trigger events in FX2, FX3 and FP
- Connection to MAME for inputs and outputs
- Drive Pixelcade devices
- SSF 7.1
- Drive DMD's via DOF2DMD using DMDExt

DOFLinx can operate in a number of ways, in fact it can do them all in combination to get the best effect. There are a couple of levels of force feedback available, full force feedback where specific triggers make DOF devices "fire" and light up. Then there is simple force feedback, where flipper keys as the input are "listened" for and when press "fire" DOF devices. When fully configured DOFLinx will attempt to use the most detailed level of force feedback first, then progressively drop back to simple force feedback dependant on what's been configured and what is available.

Full force feedback is available via Future Pinball tables scripted for DOFLinx and for Pinball FX2 / 3 games with memory map files.

Simple force feedback "listens" for flipper (and other configured key) presses and activates the configured DOF devices. It can also run a background colour display providing automated colour for tables with nothing specifically coded.

The fall-back process is quite simple. For Future Pinball DOFLinx will first check if the table has been coded for DOFLinx, if so you have a fully active and full featured force feedback link. If not DOFLinx will check if the table has been flagged specifically, or simply setup in the absence of a full force feedback link, to selectively use simple force feedback.

The fall-back process for Pinball FX2 and FX3 is similar. DOFLinx will first check for a trigger configuration file for the table being played. If one is present, then full force feedback is turned on and scripted actions are taken when the various trigger events occur. If no trigger configuration file exists then simple force feedback is enabled.

## Requirements

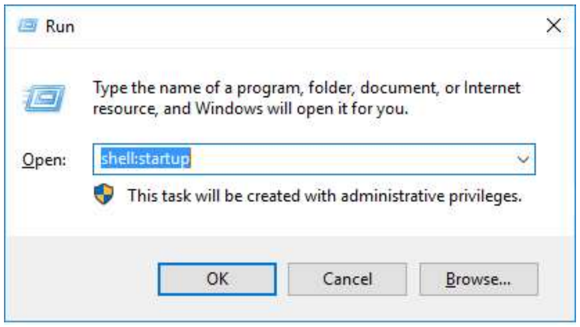
- A pinball cabinet – but then who doesn't have one of these
- DOF R3++ installed and running (Look here https://www.vpforums.org/index.php?showtopic=39557&hl=)
- At least one LEDWiz, Pinscape, PacLed64, Ultimate I/O, PinControl1, PinControl2, PacDrive, SainSmart, PinOne or Dude's Cab controller board
- If you want, a Teensy (or similar) setup for addressable LEDs in DOF
- Keyboard input (or emulator) to your favourite pinball emulator
- Possibly some Xbox or DINPUT controllers
- Some patience

## Installation

Have a read of the "Start Here – DOFLinx Quicker Guide"

In summary

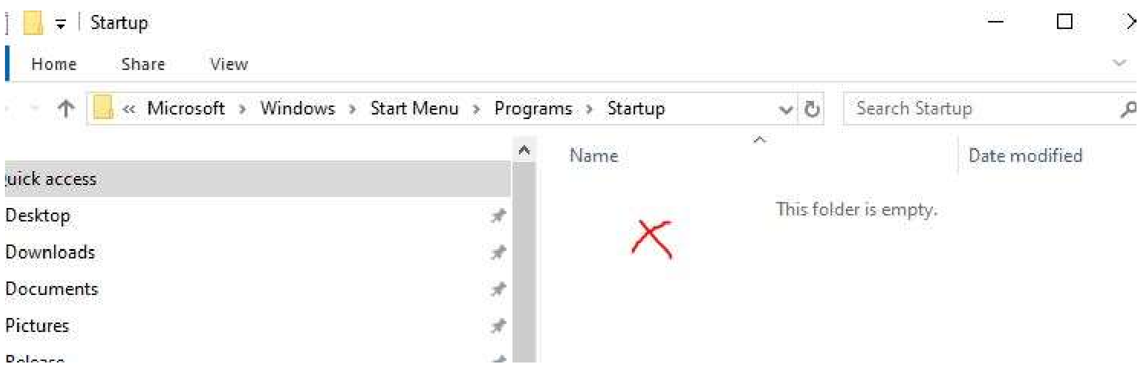
1. Create a folder for DOFLinx, let's call it C:\DOFLINX
2. Download DOFLinx from here https://www.vpforums.org/index.php?app=downloads&showfile=12318 into C:\DOFLINX
3. Unblock the downloaded DOFLinx.Vxx.ZIP file
4. Unpack C:\DOFLINX\DOFLinx.Vxx.ZIP into the same folder
5. Copy the premade DOFLinx-Shortcut.lnk to startup
   To open the startup folder in Windows use Win-R and type in shell:startup


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: shell:startup

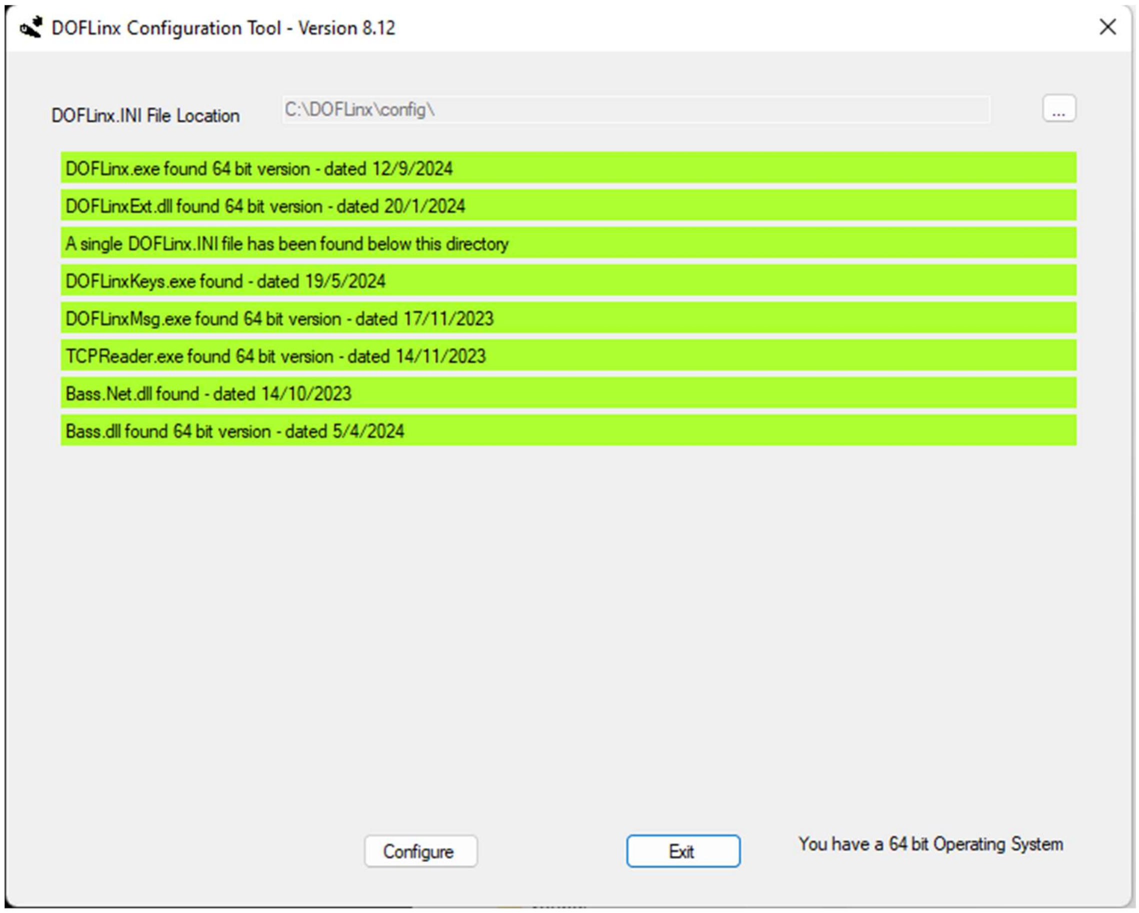
This task will be created with administrative privileges.



This folder is empty.

When Explorer comes up with the startup folder right click in the blank area on the right and paste. This will copy your newly created shortcut for DOFLinx.

6. Run DOFLinxConfig
   a. Check that you do not have any errors and that you understand any warnings
   b. Press the "Configure" button and work through all of your settings



DOFLinx.INI File Location C:\DOFLinx\config\

DOFLinx.exe found 64 bit version - dated 12/9/2024
DOFLinxExt.dll found 64 bit version - dated 20/1/2024
A single DOFLinx.INI file has been found below this directory
DOFLinxKeys.exe found - dated 19/5/2024
DOFLinxMsg.exe found 64 bit version - dated 17/11/2023
TCPReader.exe found 64 bit version - dated 14/11/2023
Bass.Net.dll found - dated 14/10/2023
Bass.dll found 64 bit version - dated 5/4/2024

You have a 64 bit Operating System

7. Reboot
8. Start your game emulator (Pinball FX, MAME, Pinball FX3, etc)

## 32 or 64 Bit

DOFLinx has both 32 and 64 bit versions. They have identical functionality and should be selected as per your choice and requirement. Of course, to use the 64 bit version you must also have a 64 bit version of Windows installed. The standard for all new installations is 64 bit.

More information is in the DOFLinx Quicker guide.

## Upgrading

As per installation, just unpack the new EXE file over the top of the old one. Of course, you might need to kill off the DOFLinx process if you have it running at all times as a set and forget.

Notes

a) From Version 2.0 all lists in the INI that were space separated are now comma separated. This means all lists are comma separated. Space separated lists no longer work from version 7.0
b) At installation or upgrade, you may need to "unblock" the EXE file if you are using Windows 10. Just right click, properties, and unblock.
c) From version 7.0 there is no need to run with administrator rights unless you run any of your pinball emulators with administrator rights.
d) From version 8.12 the installation structure recommended changed for pinball users from collocating with your DirectOutput installation to a separate \DOFLinx folder. This coincides with the change in DirectOutput to having an installation where 32 and 64 bit versions of DOF can coexist better.

## Help

Forum support is available herewww.vpforums.org/index.php?showforum=104

## Starting it up

DOFLinx can be run is two ways:

### *Set and Forget (Recommended)*

This is where DOFLinx runs all the time in the background and simple "wakes up" when it sees a process that is configured to run with. By default, that process is 'Pinball FX2'. DOFLinx uses SFA resources and can comfortably "sleep" while you run VP or FP, etc.
To use in this mode, simply add a shortcut to the program into your Windows start up folder.

This is the mode I designed to utility to work in, and the one I strongly recommend using.

### *Launch Before*

In this mode you start the DOFLinx program as a 'Launch Before' application from your menu system (ie PinballX). You must then stop the DOFLinx process via a 'Run After' script as well. The best way to stop DOFLinx is to send it a message using the DOFLinxMsg utility or something like a VBS script. Killing the process itself is always a bit ugly.

Both modes have merit, it really comes down to how you want to run things on your cabinet. Here are a few comparisons.

| ***Set and Forget*** | ***Launch Before*** |
|---|---|
| One setup | The process is not running when you use other programs |
| No need to start and stop processes | |
| Can use different colour sets for Pinball FX2 / 3 | |
| No timing hassles that can occur when you stop and start DOFLinx | |
| Ability to use attract mode | |
| Needed to show MAME marquees from the menu on Pixelcade or DMD via DOF2DMD | |

Regardless of the mode, you may want to consider the mix of programs you have running that are trying to control your devices. Too many things trying to control your output board may cause you and your PC some confusion.

## Command Line Options

There are a few optional command line options for DOFLinx. They are:

### *DISABLEDOF=YES*

This parameter set to YES will disable the search and startup of DOF based controllers. It is used for people running DOFLinx on desktop and VR configurations where there are no DOF based toys being run but you do want B2S backglasses with FX2/3 or you want Xbox rumble output.

### *DOF2DMD=NO*

Used to stop the automatic startup of DOF2DMD when DOFLinx starts. Use this if you have a cabinet and front end that runs both pinball using your DMD and MAME all together. Not required on a MAME or pinball only cabinet. You shouldn't run this command line argument until after you are sure everything is setup correctly as some of the more informative DOF2DMD startup messages from DOFLinx will be suppressed.

### *DUDESCAB=NO*

Add this parameter if you do not want DOFLinx to search for a Dude's Cab controller.

### *FTDI=NO*

Add this parameter if you do not want DOFLinx to search for any FTDI (SainSmart devices). This parameter is intended for those people with other non-Pinball based FTDI chip based devices connected, ie Fan-o-Matic Pro. If DOFLinx detects a FTDI controlled device that is not a SainSmart relay board, because FTDI devices can only have one program controlling them, there is a remote risk of conflict, hence the ability to switch this test off.

### *GAME_NAME=*

This option is technically valid in both modes, but realistically only useful in Launch Before mode. For Set and Forget mode you can message a GAME_NAME to the running process.

Adding this parameter to the command line allows you to force the Game Name for the GAME_COLOUR parameter palette selections. Read about the GAME_COLOUR parameter below.

### *PATH_DOF=*

If for some reason you need to force the path to where the appropriate DIrectOutput.dll resides. In normal operation this should not be required.

### *PATH_INI=*

The path to locate INI files in.  This value can be set as a command line argument, via a real-time command or in an INI file.  Obviously if you wish to locate your DOFLinx.INI in this path then you must use the command line to set this value.

The path can be absolute, ie C:\DOFLinx\INI\ or relative from the location where the program executes from, ie if the program is in C:\DirectOutpu and you have a DOFLinx_INI folder below that the value can be DOFLinx_INI\

### *PINONE=NO*

Add this parameter if you do not want DOFLinx to search for a PinOne controller.

### *PINSCAPE=NO*

Add this parameter if you do not want DOFLinx to search for a Pinscape controller (native Pinscape, not a Pinscape in LEDWiz emulation mode).  This parameter is really intended for people running DOFLinx from V5.0 onward who have the DOF Framework branch that does not have Pinscape support.  Regardless of having a Pinscape device or not, if you do not have a DOF Framework version that supports Pinscape you need this setting or you will get a .NET framework error.

### *SUP_INI=*

This option is valid in both modes, but realistically only useful in Launch Before mode as command line option.  For Set and Forget mode you can message a SUP_INI file to the running process.

To enable game specific settings you can place common parameters (see below for parameter descriptions) in the default DOFLinx.INI file and other in XXX.INI.  You then supply DOFLinx with that supplementary INI file via this command line option.  Ie DOFLinx SUP_INI=MyGame where MyGame.ini contains your game specific key and colour information.

If you want to keep things neat and tidy you can create a sub-folder under your DOF folder for INI files.  If you do this just add a path name to the SUP_INI= parameter.  If your folder is MyINIs, then SUP_INI="MyINIs\TheINI".  The path is relative to where you start the program from, being the DOF folder.

Note: Don't add the ".ini" to the end of the name.

## The Parameter Files

By default, all parameters are in the DOFLinx.INI file.  Many parameters have a default, so may not be required.  Check each parameter in the section below.

When using DOFLinx in a Launch Before setup you may want different keys to do different things for a different emulator or even at the table level.  To do this, place the parameters you don't want to change in DOFLinx.INI and the ones you want to change in XYZ.INI where XYZ is any valid file name.

Some parameters can be overridden by redefining them in your SUP_INI file, others cannot.  Honestly, I haven't tested them all.  The design is for them to appear once only.  If you want to try overriding, feel free to give it a go, I have done so for KEY_TO_RGB and it works like a treat.

Note: If you're running DOFLinx as Set and Forget, and edit your DOFLinx.INI file, you will need to restart the DOFLinx process.  This can be done by simply rebooting, or kill it off in Task Manager and restart it by double clicking on it.  Remember, unless you are in debug mode you will not see a window pop up in front of you, so avoid trying to start it twice!

## The Parameters

### *Device / Port*

A device / port setting is used by many of the parameters.  It is always in the form DOOO, where D= the device number as assigned by DOFLinx (if in doubt run in debug and read what it is).  They start at 1 and go up from there.  So, if you've only got one SainSmart, LedWiz, etc then it is almost certainly 1.  The OOO= is the output port number.  This can be from 1 up to the maximum supported by your device, ie LedWiz its 32.  So DOOO as 1009 is device 1, output port 9, 2027 is device number 2, port 27, 1120 is device 1, output port 120.  Prior to V7.23 the format was DOO as opposed to the current DOOO.  This change was made to accommodate output ports >=100.  Both DOO and DOOO formats are currently supported.

Note that a LedWiz or KL25Z device can have an ID number that is different to the device number assigned by DOFLinx.  For example, ID #8 may initialise as device #2.  So, the second device that comes with a KL25Z has the ID of 8 and will be normally be addressed as 2001 to 2032 (note this may vary based on how many devices you have and how they are setup.  If in doubt look at the DOFLinx.LOG file, the device ID's and numbers are always in there from start-up regardless of the DEBUG= flag.

### *Key numbers*

Key numbers, represented by in the commands below, can be either keyboard codes or Xbox keys.

#### Keyboard

This is the key scan code to monitor, shown as KK.  Codes can be found here
https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx

So, for the left shift key KK is A0 (A zero), and right control key KK is A3.

#### Xbox Controller

The code to indicate an Xbox key has three parts;

1. "X" to indicate Xbox
2. # is the controller number from 1 to 4
3. CC the 1 or 2 digit code for the button

So "X1A" (no quotes in the INI file) is Xbox controller #1 "A" button.  Valid button codes are;

DU – Dpad Up
DD – Dpad Down
DL – Dpad Left
DR – Dpad Right
ST – Start
BK – Back
LT – Left Thumb
RT – Right Thumb
LS – Left Shoulder

RS – Right Shoulder
A – A button
B – B button
X – X button
Y – Y button

## Joy Stick Controller

The code to indicate a joystick has three parts;

1. "J" to indicate a joystick
2. ## is the joystick number from 1 to 16
3. CC is the DPAD button or button number

So "J01DD" (no quotes in INI file) is joystick #1 DPAD Down button, and "J0204" is joystick #2, button number #4.

DU – Dpad Up
DD – Dpad Down
DL – Dpad Left
DR – Dpad Right
XX – is the joystick button number from 1 to 32 with leading zeros up to and including 9, ie 01,02, etc.

### *ATTRACT_MARQUEE_TIME=X*

Sets the time in milliseconds that marquees will display when Attract Mode is running. To not display marquees when in Attract Mode set this parameter to 0 or just leave it out of your INI file. Do not set this parameter to less than 1000 (1 second) as you will flood the device trying to display the marquee.

### *ATTRACT_MODE=[0/1]*

Manually turn attract mode on or off. On is 1 and off is 0, so turning attract mode on is ATTRACT_MODE=1

### *ATTRACT_SETUP=DDD,N,DOOO,CCCC,U,X[,N,DOOO,CCCC,U,X]*

Load up an attract mode sequence, DDD is the unit of delay in milliseconds between 100 and 10000 inclusive, so 1000 sets a unit of delay of 1 second.

The next section of five items is repeated as many times as you want.

N is the item number of the current display item. You must have at least one display item numbered 1.

DOOO is the device / output to act on.

CCCC is the colour to display. RANDOM is valid for a RGB device, ON / OFF is used for mono devices.

U is the number of units to delay. So, if U = 3 and DDD=500 then this item will be displayed for 1500mS or 1.5 seconds. Set this to 0 if the action is to happen immediately, ie turning off the last displayed colour to create a chain effect.

X is the next item to display. For a loop make the last display item 1 to start from the beginning again.

### *ATTRACT_START_DELAY=X*

Set this to a value above zero to be the number of milliseconds of inactivity until DOFLinx starts your Attract Mode activities. If you are not using Attract Mode simply leave this parameter out of your INI file. For most something like ATTRACT_START_DELAY=30000 is a good setting, this means after 30 seconds attract mode actions start. If you have attract mode in your front end menu as well, then make these timeout values align.

Inactivity is defined as, not having an emulator that DOFLinx monitors, time since the last menu system navigation message was received, time since the right-click menu was activated.

### *AUTO_MX=[0|1}*

Default is 0, can be left out of INI file. 0=off, 1=on.

Setting this parameter on (1) will make DOFLinx use some basic default MX displays for Pinball FX2, Pinball FX3 and Future Pinball DOFLinx tables that are not explicitly coded to trigger MX commands. It will use default MX displays for the five flashers, flippers, slingshots, bumpers, strobe and beacon.

If a Pinball FX2, Pinball FX3 or Future Pinball table has been coded with MX commands then this parameter will have no effect for that table and the coded commands will be used.

### *BUTTON_COLOUR_CHANGE=NNNNN,CCCC[,NNNNN,CCCC]*

An option list of button name and colour pairs to change the colour of a button set via LINK_xx buttons. This is used to change the set colour of a button after its been defined using LINK_xx.

So if in DOFLinx.INI you setup the Start button to be green using LINK_ST=101,Green then find that in a particular game you want the Start button to be blue use
BUTTON_CHANGE_COLOUR=BUT_ST,Blue

While logically not required in DOFLinx.INI, if set in the DOFLinx.INI file, buttons must be defined before they can be used. So, you must set LINK_P1= and LINK_CN= higher in your DOFLinx.INI file before using a command like above.

### *BUTTON_LIGHT_ACTION=BBB,AAA,O,N,T,C[,BBB,AAA,O,N,T,C]*

This parameter is most likely used in a game file, ie FX3 file. It links a table trigger to a button LED action. The Button LED can be RGB or MONO.

BBB is the button identifier, ie BUT_LF for left flipper.

AAA is the action from the table, common actions for FX3 are LeftFlipper, RightFlipper, Slingshot_Left, Bumper_01,NudgeLeft, etc.

O defines if this action links to the ON or OFF of the action, so LeftFlipper,ON will trigger when the left flipper activates.

N is the action to take, valid codes for button LEDs are 1,2,3,6,7, where:

- 1 = on
- 2 = off
- 3 = on for X milliseconds
- 6 = off for X milliseconds
- 7 = show alternate colour for X milliseconds

T = time in milliseconds for the action

C = alternate colour if action #7 is used. For other actions just ensure there is a valid colour there, ie Black

So BUTTON_LIGHT_ACTION=BUT_LF,LeftFlipper,ON,2,0,Black,BUT_LF,LeftFlipper,OFF,1,0,Black means that the left flipper button LED is turned off when the flipper turns on, then the second action turns on the left flipper button LED when the left flipper turns off. The effect is that the button is normally lit, then turns off as you hold in the flipper button, then turns back on when you release the button.

### *BUTTONS_LIT=NNNNN[,NNNNN]*
### *BUTTON_LIT_ADDED=NNNNN[,NNNNN]*

An optional list of buttons to light when DOFLinx activates due to a PROCESS being detected. NNNNN is in the form of BUT_P1 for player 1, etc. So lighting Player 1 and the coin button will require BUTTONS_LIT=BUT_CN, BUT_P1

BUTTONS_LIT clears the list of buttons to light then adds the buttons provided whereas BTTONS_LIT_ADDED adds buttons to the existing list. Beware of your logic when adding buttons to ensure you don't keep adding the same buttons over and over as games start / stop.

If set in the DOFLinx.INI file buttons must be defined before they can be used. So you must set LINK_P1= and LINK_CN= before using the above sample command.

### *CLOSE_DEVICES_BETWEN_ACTIVE=[0|1]*

Default is 0, can be left out of INI file, 0=off, 1=on

Don't set this to "1" (on) unless advised to do so. It will force the closing of devices between periods of activity. A very similar mode of operation occurs when a SainSmart device is detected since FTDI drivers only allow for one program to control the device at any point in time, hence DOFLinx releases the device when at all possible.

## *COLOUR_PALETTE=CCCCC,CCCCC,CCCCC, etc*

Not required, default is to use the full colour palette in the DIRECTOUTPUTCONFIG file.

This parameter allows you to set the current colour palette used by the RGB devices (RGB_OUTPUT). For example, if you want to restrict to greens and yellows then

> "COLOUR_PALETTE=Lime,Green,Yellow,Gold,Lawn_Green,Chart_Reuse,Green_Yellow,Forect_Green,Dark_Green,Yellow_Green,Light_Green"

would do it.

This is intended for use in the STARTUP section of *.MAME and *.FX files to set the colour palette to a restricted set of colours for the MAME game being run.

It is possible to use the GAME_COLOUR= command and add all MAME games to your DOFLinx.INI file and have them selected by game name. The problem with that is the list will become super long and you have to maintain two configuration areas.

## *CLOSE_ROMS=0/1*

Not required in the INI file, default is 1

The default is for DOFLinx to close DOF ROMs between uses. This ensures that other software can use DOF unhindered. This was the default action until v7.16 when this option was introduced. For pinball cabinets leaving this as default makes sense as most front ends are setup on pinball cabinets now to use DOF for menu effects and each table uses a different ROM.

For an arcade machine the setup can be quite different. At the time of writing, all MAME files (300+) use the doflinx ROM for MX effects. By not closing the ROM each time the transition between games can be sped up by 5-10 seconds each time (the amount of time it takes DOF to load a larger ROM). It also allows for the ROM to remain open (you can explicitly do this in other ways) outside of when the emulator is running. Having the ROM active allows many other effects and button setups to continue unhindered.

If the parameter is set to leave the ROM open it will still close when DOFLinx exits, is explicitly told to close the ROM by using the FF_ROM="" command, or when a new ROM is requested that doesn't match the one currently loaded.

## *DEBUG= [ 0|1|2]*

Default is 0, can be left out of INI file. 0=off, 1=on showing DOFLinx window, 2=on not showing DOFLinx window.

Setting DEBUG on optionally makes the DOFLinx window visible and always creates the DOFLinx.LOG file. It makes all messages visible in the window for =1. The same messages are written to the DOFLinx.LOG file for =1 and =2.

Other than logging and the window being visible, nothing is different between having debug on or off. Of course, you wouldn't want debug on during normal use as it will leave the window visible, create a very big log file and log file writes are a slow thing to do so may cause delay / stuttering.

DOFLinx logs the first few start-up actions every time regardless of this parameter.  The reason, well it hasn't read the flag yet, so key start-up info is sent to the log just in case.

The DOFLinx.LOG file can be safely deleted at any time.

### *DEBUG_TCP_PORT=pppp*

No default, not required

When the DEBUG parameter is set to anything other than 0 messages are added to DOFLinx.LOG.  If the DEBUG_TCP_PORT= is set to a valid TCP port, for example DEBUG_TCP_PORT=8001 then the messages sent to DOFLinx.LOG are mirrored as an output on this TCP port.  So once DEBUG_TCP_PORT is set, messages can be monitored using a TCP port reading utility such as NC.  A utility that handles DOFLinx messages, TCPReader, is included with the DOFLinx distribution.

DEBUG_TCP_PORT= can be both a DOFLinx.INI parameter or a command line parameter in your shortcut to run DOFLinx.  Setting this parameter via the command line will ensure all messages from the very beginning are logged.

With DEBUG=2 and DEBUG_TCP_PORT=8001 DOFLinx will log all messages to both the DOFLinx.LOG file and mirrored as output on TCP port 8001.  With this configuration DOFLinx will not be displayed on your cabinet screen and logging can be monitored in real time.

### *DELAY_BETWEEN_KEY_PRESS=*

Default is 500, can be left out of ini file.

This is the delay in milliseconds between key presses for KEY_TO_COLOUR, KEY_TO_OUTPUT_TIMER, KEY_TO_COLOUR_TIMER commands being triggered.  So, at its default, 500mS, if a key is detected as pressed again within 500mS the trigger action will not be reinitiated.

This delay is required given that DOFLinx is not the program in focus and hence is monitoring the keyboard as opposed to receiving keystrokes.  As such, a single quick key press can be detected multiple times without this delay.

### *DIRECTOUTPUTCONFIG=DRV:\PATH\FFFF.INI  or*
### *COLOUR_FILE= DRV:\PATH\FFFF.INI*

No default, required parameter.

This is the full file name, including path, to you directoutputconfig.ini file, ie "C:\directoutput\config\DirectOutputConfig.ini" , or "c:\DOFLinx\config\colours.ini" without the quotes.  The only reason a pointer to this file exits is to get the DOF colours, so technically it could be any file so long as it has the section from your directoutputconfig.ini file that looks like:

```
[Colors DOF]

White=#FFFFFFFF
Red=#FF0000FF
Lime=#00FF00FF
```

……

If you do make up a separate colour file and point to that, then you can also add your own custom colours. Just keep them in the same format and use any name you like less than 21 characters. You can then use those custom colours with the likes of GAME_COLOUR or KEY_TO_RGB.

If you have multiple DOF devices, then you will have multiple directoutputconfig files. Just find the one with a section that looks like the above and use that (yes I know they spell colour in some strange way, American I think?).

***IMPORTANT NOTE: This parameter must come before any colours are assigned. To avoid any issues, the easiest thing is to make sure this is the first parameter in the DOFLinx.INI file.***

## *DIRECTOUTPUTGLOBAL=DRV:\PATH\FFFF.INI*

No default, optional parameter.

Required if you intend to use addressable LEDs.

This is the full file name, including path, to your DOF R3 GlobalConfig.xml file, ie "C:\directoutput\config\GlobalConfig_b2sServer.xml" without the quotes.

You may not have a global config XML file for DOF, if not, do not panic, and do not include this parameter. I ran for years without one. This file got created on my cabinet when I added a Teensy addressable LED controller.

## *FF_DOF=CCCC,DDD[,CCCC,DDD]*

An optional parameter, most likely used in a supplementary INI file

One or a series of DOF commands to trigger events setup in ROMs vis the DOF.

You need to have selected a DOF ROM via FF_ROM= before this command can be used.

CCCC is a command to trigger in the DOF directoutputconfigxx.ini file (normally a MX effects command), ie E99

DDD is the duration. If the duration is set in the DOF directoutputconfigxxx.ini file then this value will be -1 to use the predefined duration. If the event has no duration, ie an effect for MX LEDs when a flipper is down, then the duration is a time in milliseconds for the effect to stay on.

A duration (DDD) of 0 (zero) will turn off an effect previously turned on, ie setting a flipper effect on for 10000mS (10 seconds) on flipper down, but wanting the effect to stop on flipper up.

## *FF_PC A,O,DDDDD*

Mainly used in *.MAME and possibly *.FX files.

For initiating an action on a Pixelcade device. "A" is the action, "O" is the output device, "DDDD" is the detail.

Valid actions:

T = Text, DDDD = The text to display. Remember this is in the format used in a URL, to a space is %20

U = User command. A native Pixelcade V1 or V2 command as a URL without the host name and no leading / . ie the native command `http://pixelcade.local:8080/console/stream/nes` becomes FF_PC,U,E,`console/stream/nes`

Valid devices

C = LCD

E = LED

## *FF_PUP_EVENT=EEE,VV[,EEE,V]*

An optional parameter, most likely used in a supplementary INI file

One or a series of PUP events to be triggered. EEE is the event name and V the value passed with that event identifier.

## *FF_ROM=RRRRR*

An optional parameter, most likely used in a supplementary INI file

RRR is the ROM name setup within the DOF Config tool.

This is used to set the ROM for KEY_TO_ROM= commands. If the ROM name was “doflinx” this command would look like FF_ROM=doflinx

## *FIXED_OUTPUTS= DOOO[,DOOO,DOOO]*

## *Replaces the old BUTTONS_ON=DOO[,DOO,DOO]*

Default is no outputs activated. This can be left out of the INI file.

A comma separated list of outputs to turn on when DOFLinx “wakes up”. This is designed to turn on the LEDs within buttons that can be used within the game being started. It can of course be used to turn on an output for any purpose. The output remains on until DOFLinx goes back to “sleep”

This command can use DOOO for output device and output port.

BUTTONS_ON=1016

## *FIXED_OUTPUTS_COLOUR= DOOO,CCCC[,DOOO,CCCC,DOOO,CCCC]*

## *Replaces the old BUTTONS_ON_COLOUR=DOO,CCCC[,DOO,CCCC,DOO,CCCC]*

Default is no outputs activated. This can be left out of the INI file.

A comma separated list of outputs and colours to turn on when DOFLinx “wakes up”. This is designed to turn on the RGB LEDs within buttons that can be used within the game being started. It can of course be used to turn on other RGB outputs for any purpose. The output remains on until DOFLinx goes back to “sleep”

This command can use DOOO for output device and output port.

FIXED_OUTPUTS_COLOUR=2018,Blue

## *FP_ATTEMPT_LINK=[0|1]*

Default is 0, this parameter can be left out of the INI file. 0=off, 1=on

If one of your PROCESSES= is "Future Pinball" then with this parameter set DOFLinx will try and establish a link with FP for tables that have been appropriately DOF'ed. There is a section below that details how to install the required FP script and DOF a FP table. What can I say, get DOF'ing!

If the FP link is activated you will need to setup LINK_xx parameters, see below.

## *FP_LINK_WAIT_TIME=*

Default is 40000, this parameter can be left out of the INI.

This sets the amount of time that DOFLinx will look for a Future Pinball table running DOFLinx code. DOFLinx code can only be recognised once Future Pinball has completely loaded the table and its ready to play, that is the progress bar has finished.

Setting this value too high will cause DOFLinx to stop responding (don't worry its still working) and potentially still be waiting once you start playing a non-DOFLinx'ed FP table. It should only be extended if you have a slower PC. To get an idea of what to set this to, time your cabinet from table selection from your front end until ready to play the slowest DOFLinx'ed table you have. Once you know how long that is, add 10 seconds just in case, times the result by 1000 and you have your value.

So if your cabinet takes 36 seconds to load a particular DOFLinx'ed table, then add 10 to get 46, times by 1000 to get 46000 and use FP_LINK_WAIT_TIME=46000

If you know that your cabinet always loads FP tables within 15 seconds to ready to play, then you could drop this value to 25000. This will allow DOFLinx to stop looking for a table link and move onto checking other methods of operation such as flag files for FP sooner. Probably not worth it, but you could.

## *FORCE_ACTIVE=[0|1]*

Default is 0, this parameter can be left out of the INI file. 0=off, 1=on.

When set to 1, DOFLinx is awake the entire time its running. It negates the need to check for a PROCESSES process.

This is a brute force way to use the utility but can provide ease of use in Launch Before mode for emulators other than Pinball FX2. Use this option with caution, ie get everything else going first, then think about why you want to do this. A better way, rather than using this, is to add the other emulators you want DOFLinx to be active for to your PROCESSES= list and leave this parameter out of your INI or set to 0.

*FX_LEFT_NUDGE_POINT=*
*FX_FORWARD_NUDGE_POINT=*
*FX_RIGHT_NUDGE_POINT=*
Not required, this parameter can be left out of the INI file, only used with FX3 nudge.

The point at which movement of the joystick from the Pinscape accelerometer register as a nudge in the respective direction.  Note the nudge point for LEFT will be -ve.

Nudge trigger points can be discovered and tested using the "Test Joy Stick" function on the right click menu when in DEBUG mode.

*FX2_FOCUS=*
Not required, defaults to 1

Enabled is 1, disabled is 0.  When enabled DOFLinx shifts focus to FX2 every 0.5 seconds for 60 seconds after FX2 starts up.  This is to assist with focus sometimes doing odd things as FX starts and then we start an active back glass or other things.

*FX2_WINDOW_WAIT_TIME=*
Not required, defaults to 10000

This value in milliseconds can be used to manipulate the length of time that DOFLinx scans for a valid game window after FX2 starts.  The default is a timeout of 10 seconds if a valid game window is not found.  If you are using the grid rather than a front end like PinballX then you may wish to set the maximum higher, thus giving yourself time to select a game from the grid.

*FX3_FOCUS=*
Not required, defaults to 1

Enabled is 1, disabled is 0.  When enabled DOFLinx shifts focus to FX3 every 0.5 seconds for 60 seconds after FX3 starts up.  This is to assist with focus sometimes doing odd things as FX starts and then we start an active backglass or other things.

*FX3_TRIGGER_RELEASE_DELAY*
Not required defaults to 0

This value in milliseconds determines how long a trigger from FX3 must have changed state (ie turned on or off) until DOFLinx believes it.  This was added to allow the control of false off triggering, particularly when holding the flipper on.  A value of 50mS should be enough for most situations without causing any noticeable delay.

*GAME_COLOUR=NNNN,CCCCC,CCCCC,CCCCC*
Not required, default is to use the full colour palette in the DIRECTOUTPUTCONFIG file.

This parameter allows you to limit the colour palette used by the RGB devices (RGB_OUTPUT) for a specific game. For example, if you want to restrict to greens and yellows for 'Plants Vs Zombies' a line like

> "GAME_COLOUR=Plants VS. Zombies,Lime,Green,Yellow,Gold,Lawn_Green,Chart_Reuse,Green_Yellow, Forest_Green,Dark_Green,Yellow_Green,Light_Green"

would do it.

The Game Name is automatically detected for Pinball FX2 and Pinball FX3 when launched with the [TABLEFILE] as a parameter, ie the default mode from PinballX. If you wish to use a separate palette via GAME_COLOUR for other emulators you will need to force the GAME_NAME via message just before you launch that emulator, ie "DOFLinxMsg GAME_NAME=SomeTable".

Even with a limited palette via this option, KEY_TO_RGB can still use any colour in the full range.

### *JOY_PORT=*

Not required, this parameter can be left out of the INI file, only used with FX3 nudge. Default = 1.

The port number of the joystick port, between 1 and 16, that the Pinscape accelerometer maps to.

### *JOY_X_OFFSET=*
### *JOY_Y_OFFSET=*

Not required, these parameters can be left out of the INI file, only used with FX3 nudge. Default = 0.

These values are the offset from zero of the resting position of the joystick. They can be found using the "Test Joy Stick" function from the right-click test menu when DOFLinx is in DEBUG.

### *KEY_TO_COLOUR=KK,DOOO,CCC,KK,DOOO,CCC,KK,DOOO,CCC*

Default is no KEY_TO_COLOUR activity, this parameter can be left out of the INI file.

A comma separated list in sets of three parameters key(KK), output (DOOO), colour to set(CCC).

This is used to configure an RGB device to a set colour when a key is pressed. For example, set you under cabinet RGB to display Red when a nudge key is pressed.

### *KEY_TO_COLOUR_TIMER=KK,DOOO,CCC,MMMM[,KK,DOOO,CCC,MMMM]*

Default is no KEY_TO_COLOUR_TIMER activity, this parameter can be left out of the INI file.

A comma separated list in sets of four parameters key (KK), output(DOOO), colour to set (CCC) and the time in milliseconds to display that colour (MMM).

This is used to set an RGB to a defined colour for a set time from when a key is pressed. The RGB will revert back to colour that it was prior, or would have been in the RGB sequence when the timer expirers. While active any RGB_OUTPUT activity on that RGB port is suspended.

## *KEY_TO_COLOUR_TOGGLE=KK,DOOO,CCC[,KK,DOOO,CCC,KK,DOOO,CCC]*
Default is no KEY_TO_COLOUR_TOGGLE activity, this parameter can be left out of the INI file.

A comma separated list in sets of three parameters key(KK), output (DOOO), colour to set(CCC).

This is used to set an RGB to the defined colour while the key is pressed and then revert back to the colour that was there prior, or would have been in the RGB sequence, when the key is released. While pressed any RGB_OUTPUT activity on that RGB port will be suspended.

## *KEY_TO_COMMAND=NNN,CCCC[,NNN,CCCC]*
Default is no KEY_TO_COMMAND activity, this parameter can be left out of the INI file.

A comma separated list of button names (BUT-ST, BUT_B1, etc) and Actions as defined in the [COMMAND] section of a FX3 or MAME file.

To make button #1 (fire) trigger a “shoot” command you would have
KEY_TO_COMMAND=BUT_B1,shoot

Then define the “shoot” action, such as;

```
[COMMAND]
shoot|ON|FF_DOR E216,-1
```

## *KEY_TO_OUTPUT=KK,DOOO[,KK,DOOO,KK,DOOO]*
Default is no key KEY_TO_OUTPUT activity, this parameter can be left out of the INI file.

A comma separated list in sets of keys (KK) and the corresponding output (DOOO) to turn on when its pressed. The output remains on while the key is pressed.

This can be used to turn on a button LED when the button is pressed, making the cabinet seem more interactive. It can be used to turn on a relay when a button is pressed activating a toy or relay for other purposes.

Keys can be defined more than once meaning that one key can be set to turn on multiple outputs while pressed, ie light its own LED and run a relay or toy.

## *KEY_TO_OUTPUT_TIMER=KK,DOOO,MMMM[,KK,DOOO,MMMM]*
Default is no key KEY_TO_OUTPUT_TIMER activity, this parameter can be left out of the INI file.

A comma separated list in sets of keys (KK) and the corresponding output (DOOO) to turn on for (MMMM) when its pressed. The output remains on until MMMM milliseconds have expired.

This can be used to turn run a shaker, turn on a beacon for s short period as a key is pressed.

Keys can be defined more than once meaning that one key can be set to turn on multiple outputs for different time periods.

### *KEY_TO_ROM=KK,CCC,DDD[,KK,CCC,DDD]*

Default is no KEY_TO_ROM activity, this parameter can be left out of the INI file.

To use this command you must have previously set a ROM with FF_ROM=

This command is designed to allow the use of addressable LEDs with emulators other than FP, FX or VP such as MAME.

A comma separated list in sets of keys (KK) with the corresponding ROM command (CCC) and duration (DDD) for that command.

ROM commands must be set in the respective DirectOutputConfigXX.ini file. For MX ROM commands this is likely to be DIrectOutputConfig30.ini. The command will look something like E120 or E410. The duration is set in milliseconds for the command to be active. There are two special durations, 0 to turn the command off, and -1 for the command to run using its default time contained in the DirectOutputConfigXX.ini file.

The command is use looks like KEY_TO_ROM=A0,E130,-1,A1,110,500

### *KEY_TO_XBOX_RUMBLE= KK,C,M,SS,DDD[,KK,C,M,SS,DDD]*

No default, only required if using this functionality.

This functionality allows Xbox rumble controls to be driven from pressed keys. By setting this parameter you can drive the selected Xbox motor on the selected controller at any valid speed for a settable number of milliseconds.

Load a series of keys and rumble motors, intensity and durations.

KK is the key identifier to trigger this rumble.

C is the Xbox controller from 1 to 4. You can use the GamePad test screen in DOFLinx (available when in DEBUG and visible) to determine the controller number.

M is the motor, so L, R or B for both.

SS is the motor speed. Valid values for an XBox motor speed are 0 to 65535 (100%)

DDD is the duration in milliseconds.

KEY_TO_XBOX_RUMBLE=21,1,L,50000,1000,J0103,1,R,30000,1500 will turn on either the left rumble motor to a speed of 50000 (about 75%) for 1 second when the Page Up key is pressed, and turn on the right rumble motor to a speed of 30000 (about 50%) for 1.5 seconds when the 3rd button on joystick #1 is pressed.

### *L_FLIPPER_KEY=KK*

No default, required parameter.

This is the key code for the left flipper key.

### *L_FLIPPER_OUTPUT=DOOO*

No default, required parameter.

This is port to trigger for the solenoid / contactor when the left flipper key is pressed

*LINK_xx=DOOO,FFFFF,MMMM,III[, DOOO,FFFFF,MMMM,III,DOOO,FFFF,MMMM,III]*
No default, required for the FP and FX links.

A comma separated list of sets of output (DOOO), Default on time for the device (FFFF), maximum on time for the device (MMMM) at intensity (III). You can have multiple sets of these three just in case you need multiple outputs for the one logical device.

The default on time is used when an FP or FX author triggers the device with a default on time, quite normal for solenoid based devices. Using the default on time allows you to set the best time for your device. The maximum on time allows you to stop a FP author from turning on your solenoid, etc for a super long time and not turning it off. Turning on a solenoid / contactor for too long may cause it to overheat. It is normal for a FP and FX author to turn on your flipper solenoid on key down and off at key up, as such, don't set the maximum on-time for flippers to anything too short, I'd suggest at least 10000 for 10 seconds.

Intensity can be in the range of 0 to 255 where 0 is off (ie the device will never turn on despite being triggered), through to 255, being fully on. In most circumstances this will be 255. Situations where it might be a different value are for something like a shaker motor that needs to run at 50% power, in that case set it to 128. This value will only work properly for output devices (and boosters if applicable) that use proportional outputs, for example LedWiz, PacLED.

For xx being as per below. ie LINK_MC=1030,75,1000,255 sets the mid field centre solenoid / contactor to device #1, port 30 with a default time of 75ms, a maximum on time of 1 second, with the intensity set to 255 being fully on.

LF = Left flipper
RF = Right flipper
LS = Left slingshot
RS= Right slingshot
ML = Mid field left solenoid
MC = Mid field centre solenoid
MR = Mid field right solenoid
BL = Back left solenoid
BC = Back centre solenoid
BR = Back right solenoid
SH = Shaker motor
GR = Gear motor
KN = Knocker
FN = Fan
BE = Bell
C1 = Chime 1 (high note)
C2 = Chime 2 (mid note)
C3 = Chime 3 (low note)

*LINK_xx=DOOO,MODE,TTTT,III[,DOOO,MODE,TTTT,III]*
No default, required for the FP and FX link.

A comma separated list of sets of output (DOOO), MODE being ON or FLASH, this allows for DOFLinx to pulse the output or just turn it on. Some physical devices will flash themselves, whereas others need the output to be pulsed. TTTT is only relevant for MODE FLASH, it is the time for each on then off pulse. So a TTTT of 100Ms will turn the output on for 100Ms then off for 100Ms.

Intensity can be in the range of 0 to 255 where 0 is off (ie the device will never turn on despite being triggered), through to 255, being fully on. In most circumstances this will be 255. Situations where it might be a different value are for something like a beacon that needs to run at 50% power, in that case set it to 128. This value will only work properly for output devices (and boosters if applicable) that use proportional outputs, for example LedWiz, PacLED.

SR = Strobe
BK = Beacon

LINK_SR=1005,ON,0,255 – This will set device #1, port #5 to be turned on when the strobe is activated

LINK_SR=1005,FLASH,150,255 – This will turn device #1, port #5 to be on for 150Ms then off for 150Ms over and over for the time the device is turned on from the FP table.

### *LINK_BUT_xx=DOOO,CCCC,KK[,DOOO,CCCC,KK]*

No default, required for the FP, FX and MAME link.

A comma separated list of devices to activate when the associated button LED illumination message is sent as an action. Multiple devices are possible, ie you have more than one coin button and want to turn them all on.

The button can be either a single LED or RGB colour based button. For a single LED illuminated button set CCCC as the colour to MONO LINK_BUT_EB=1002,MONO,4A . Otherwise the colour you want the button to be illuminated as, LINK_BUT_EB=1002,YELLOW,4A. Where the input key is "J" (4A)

The third parameter is the key code (KK) that this button produces when pressed. So if the key is a number 1 then its 31, if it's a number 5 then its 35. These are the same key codes used elsewhere in DOFLinx. Having the keycode is important if you wish to use key alias elsewhere in DOFLinx. For example, if you wish to use the KEY_TO_COLOUR_TIMER= and not specifiy a key code, you can specify a button name, ie BUT_P1. This means that in all of your SUP.INI, MAME, FX3 files you can use a key alias and not have to define the key code everyone. Much simpler for transportable configuration files that we can all share.

For xx being as per below. ie LINK_BUT_ST=1003,Cyan,31 sets the Start button illumination LED to device #1, port 3, makes the colour of the RGB button LEDs cyan, and defines the Start Button input key code(ie the key stroke when you press the start button) as "1". If the button is a colour button then the device and port is the red LED of the RGB LED within the illuminated button.

BUT_ST = Start button
BUT_EB = Extra Ball button
BUT_EX = Exit button
BUT_CN = Coin button
BUT_LB = Launch Ball button

| | |
|---|---|
| BUT_FR | = Fire Button |
| BUT_P1 | = Player 1 |
| BUT_P2 | = Player 2 |
| BUT_P3 | = Player 3 |
| BUT_P4 | = Player 4 |
| BUT_PS | = Pause |
| BUT_RE | = Reset |
| BUT_LF | = Flipper Left |
| BUT_RF | = Flipper Right |
| BUT_ML | = Magna Save Left |
| BUT_MR | = Magna Save Right |
| BUT_B1 – BUT_B8 | = Arcade buttons |
| BUT_CH | = Cheat |
| BUT_MN | = Menu |

### *LINK_xx=DOOO,CCCC,[,DOOO,CCCC] the old way of doing it, retained for compatibility from V7.10*

No default, required for the FP, FX and MAME link.

A comma separated list of devices to activate when the associated button LED illumination message is sent as an action. Multiple devices are possible, ie you have more than one coin button and want to turn them all on.

The button can be either a single LED or RGB colour based button. For a single LED illuminated button set CCCC as the colour to MONO or just leave it blank, LINK_EB=1002,MONO or LINK_EB=1002, . Otherwise the colour you want the button to be illuminated as, LINK_EB=1002,YELLOW.

For xx being as per below. ie LINK_ST=1003 sets the Start button illumination LED to device #1, port 3. If the button is a colour button then the device and port is the red LED of the RGB LED within the illuminated button.

| | |
|---|---|
| ST | = Start button |
| EB | = Extra Ball button |
| EX | = Exit button |
| CN | = Coin button |
| LB | = Launch Ball button |
| FR | = Fire Button |

### *LINK_xxxx=DOOO[,DOOO]*

No default, required for the FP, FX and MAME link.

A comma separated list of devices to activate when the associated flasher message is sent. Multiple devices are possible, ie you have more than one centre flasher and want to turn them all on.

For xxxx being as per below. ie LINK_FLCN=1003 sets the red LED of the flasher to device #1, port 3.

| | |
|---|---|
| FLOL | = Flasher outer left |
| FLIL | = Flasher Inner Left |
| FLCN | = Flasher Centre |
| FLOR | = Flasher Outer Right |

FLIR = Flasher Inner Right

## *LOAD_SOUND=NNN,V,SPEAKER(S),File1[,FileX]*

Used in the [STARTUP] section of a game file to load SSF sounds. Unless you are customising this will not be relevant to you.

NNN is the label used to identify this sound when using FF_SSF in a table.FX file.

V is volume from 1 to 100%, mostly this will be 100 unless you need the sound to be quieter relative to other SSF sounds.

Speaker(s) specifies where to output the sound. Available options are:

| | 7.1 | 5.1 | 2 Speaker |
|---|---|---|---|
| FRONT | Yes | | |
| SIDE | Yes | Becomes FRONT | Becomes FRONT |
| REAR | Yes | | Becomes FRONT |
| CENTRE | Yes | | Becomes FRONT |
| LEFT_FRONT | Yes | | |
| RIGHT_FRONT | Yes | | |
| LEFT_SIDE | Yes | Becomes FRONT_LEFT | Becomes FRONT_LEFT |
| RIGHT_SIDE | Yes | Becomes FRONT_RIGHT | Becomes FRONT_RIGHT |
| LEFT_REAR | Yes | | Becomes FRONT_LEFT |
| RIGHT_REAR | Yes | | Becomes FRONT_RIGHT |

File1 to FileX are the WAV files to load for this sound. You at a minimum one file. When there is more than one sound file loaded, they will play sequentially each time this sound is called.

## *MAME_FOCUS=*

Not required, defaults to 1

Enabled is 1, disabled is 0. When enabled DOFLinx shifts focus to MAME every 0.5 seconds for 60 seconds after MAME starts up. This is to assist with focus sometimes doing odd things as MAME starts and then we start an active back glass or other things.

## *MAME_HISCORE_FOLDER=*

This is the folder where you can find the MAME high score files. Usually this parameter is not required unless you use a non-standard folder for high scores. DOFLinx use in order;

1. The folder given in this parameter, if not blank
2. MAME_PATH + \hiscore\ - the default from about MAME 0.175
3. MAME_PATH +\hi\ - the original default folder

## *MAME_LINK_WAIT_TIME=*

Default=5000, can be left out of INI

This sets the maximum number of milliseconds to wait for MAME to allow a network connection after the process starts. A couple of seconds should really be enough.

### *MAME_PROCESS=*

No default, can be left out of INI.

This is the process name of the MAME emulator you are using.  It must also appear in the PROCESSES= as an activating process.  The MAME process needs to be specifically named as there are so many variants of the MAME executable.  For most people on a Windows 10 platform using the core MAME emulator this will be MAME_PROCESS=MAME

### *MAME_SCORE_CHECK_PERIOD=*

Default is 250mS, can be left out of INI

This is the maximum delay between score checks for a MAME game.

### *MAME_TRIGGER_OUTPUT_PATH=*

No default, can be left out of INI.

If this path is set, when a MAME game runs a text-based output file will be created.  If the MAME game has outputs, then they will be in a file with the same name as the MAME game.  If there are no outputs sent, then a blank file, again with the same name as the game, will be created in a subdirectory of this path called “NoTriggers”.

The trigger names can then be used to build XX.MAME files to trigger actions through DOFLinx.

The trigger “pause” is suppressed from this output as it can be manually generated in every game by pressing the MAME Pause key.  It’s a general MAME output, not a specific game trigger.  The Pause trigger is still valid, just not potentially placed in every trigger file generated via DOFLinx.

### *MAX_FLIPPER_ON=SSSS*

Default 5000, can be left out of INI file.

This is the maximum time, in milliseconds, that a flipper output can remain on without the flipper key being repressed.  This is to provide protection for those worried about burning out solenoids / contacts if you hold in the flipper key too long.

### *NIGHT_MODE=[1|0]*

Default is 0, can be left out of INI file.

This setting is designed to be used in real-time, ideally via a message sent to DOFLinx.  Setting this on (1) will disable noise making devices (flippers, solenoids, knocker) and let you play without disturbing the rest of the household.

When a FP table with a link is active it will disable the device mechanical feedback and re-enable the table sounds.  Night Mode can be turned on / off at any time and will notify FP in real-time.

### *NUDGE_CHECK_BASIC_KEYBOARD=*

Not required, this parameter can be left out of the INI, the default is 0 or disabled.

Setting this to 1, on, will enable checking of the Pinscape nudge checking when DOFLinx is operating in its most basic mode checking keyboard keys for a process that triggers it in the PROCESSES= parameter. This allows other emulators to use the joystick nudge checking.

### *NUDGE_LEFT_KEY=*
### *NUDGE_FORWARD_KEY=*
### *NUDGE_RIGHT_KEY=*

Not required, this parameter can be left out of the INI file, only used with FX3 nudge.

These are the keys setup within FX3 to trigger keyboard nudge. The default keys are L-Ctrl, R-Ctrl and Space. Sadly, the L-Ctrl and R-Ctrl keys do not work, as such, the keys will need to be changed in FX3 to something else. I suggest “L” and “R”. While in the form KK, these are only keyboard keys, they cannot be Xbox buttons.

### *NUDGE_LEFT_INPUT=*
### *NUDGE_FORWARD_INPUT=*
### *NUDGE_RIGHT_INPUT=*

Not required, this parameter can be left out of the INI file, only used with FX3 nudge.

If like me, you change the keys in FX3 to trigger nudge (as per NUDGE_???_KEY= above) but don’t want to reprogram your nudge buttons away from L-Ctrl and R-Ctrl because you use them for other inputs, ie VPX or PinballX, then setup these parameters and DOFLinx will parse the Ctrl key presses to the new FX3 nudge keys. These are in the format of KK, so can e keyboard keys or Xbox buttons.

### *NUDGE_WAIT=*

Not required, this parameter can be left out of the INI file, default is 1500

This is the minimum delay between nudge events from the Pinscape. It allows you to stop getting multiple nudge events from the single nudge. The minimum this can be set to is 1500Ms (1.5 seconds)

Normally you should not need to play with this, but hey why not give it a try.

### *OUTPUT_GAME_NAME=[0|1]*

Default is 0, can be left out of INI file. 0=off, 1=on.

When turned on it will force the Game Name to be written to the debug log regardless of the DEBUG setting. This is handy when you are setting up GAME_COLOUR palettes and want the game name without a full DOFLinx.LOG being created.

### *PATH_B2S_SERVER=*

The path where your B2SServerEXE.exe is located.  The default is C:\VP\

This value can be set via real-time command or in an INI file.

This parameter is required if you plan to use B2S back glass files with Pinball FX2 or Pinball FX3.  A common place to find this file would be C:\VP\Tables\ if you have a nice neat VP setup.

### *PATH_FX2=*

The path to locate FX2 files in.  This value can be set via a real-time command or in an INI file.

The path can be absolute, ie C:\DOFLinx\FX2\ or relative from the location where the program executes from, ie if the program is in C:\DirectOutput and you have a DOFLinx_FX2 folder below that the value can be DOFLinx_FX2\

Not setting a value or setting it to nothing means the FX2 files are to be located in the same folder as DOFLinx.exe

### *PATH_FX3=*

Same as above but for FX3 files.

Usually C:\DOFLinx\FX3\

### *PATH_FX=*

Same as above but for FX files (Pinball FX).

Usually C:\DOFLinx\FX\

### *PATH_FX2_B2S=*

The path to the directb2s files for FX2 games.  This value can be set via real-time command or in the INI file.

The path can be absolute, ie C:\DOFLinx\FX2_B2S\ or relative from the location where the program executes from, ie if the program is in C:\DirectOutput and you have a DOFLinx_FX2_B2S folder below that the value can be DOFLinx_FX2_B2S\

The value must end with a backslash \

Not setting a value or setting it to nothing means the B2S files are to be located in the same folder as DOFLinx.exe

### *PATH_FX3_B2S=*

Same as above but for FX3 B2S files.  If you already have PATH_FX2_B2S= setup, you can point this to the same folder.  About 90% of B2S files are the same between FX2 and FX3 and it saves having two copies.

### *PATH_FX_B2S=*
Same as above but for FX B2S files (Pinball FX).

### *PATH_HI2TXT=*
The path to where Hi2Txt has been installed.  Hi2Txt is used to get high scores from MAME.

### *PATH_INI=*
The path to locate INI files in.  This value can be set as a command line argument, via a real-time command or in an INI file.  Obviously if you wish to locate your DOFLinx.INI in this path then you must use the command line to set this value.

The path can be absolute, ie C:\DOFLinx\INI\ or relative from the location where the program executes from, ie if the program is in C:\DirectOutput and you have a DOFLinx_INI folder below that the value can be DOFLinx_INI\

Not setting a value or setting it to nothing means the INI files are to be located in the same folder as DOFLinx.exe

### *PATH_MAME=*
The path to locate MAME files in.  This value can be set via a real-time command or in an INI file.

The path can be absolute, ie C:\DOFLinx\MAME\ or relative from the location where the program executes from, ie if the program is in C:\DirectOutput and you have a DOFLinx_MAME folder below that the value can be DOFLinx_MAME\ .  Absolute path names such as C:\DOFLinx\MAME\ are best.

The value must end with a backslash \

Not setting a value or setting it to nothing means the MAME files are to be located in the same folder as DOFLinx.exe

### *PATH_PIXELCADE=*
The path to where Pixelcade has been installed. Ie PATH_PIXELCADE=C:\Pixelacde\

If you don't have Pixelcade then don't add this parameter to your INI file.

If the path is set and pixelweb is not running, then DOFLinx will attempt to run pixelweb in the background for you.

### *PINCONTROL2_UNIVERSE=*
***DEPRECIATED – Now able to pick this up from your cabinet.xml from V6.50***

No default, can be left out on INI

If you have a PinControl2 board you will need to set this parameter and PINCONTROL_BROADCST. The normal setting required will be PINCONTROL2_UNIVERSE=0, if you need anything else you will have been given that parameter value with your board.

### *PINCONTROL2_BROADCAST=*

***DEPRECIATED – Now able to pick this up from your cabinet.xml from V6.50***

No default, can be left out on INI

If you have a PinControl2 board you will need to set this parameter and PINCONTROL_UNIVERSE. The normal setting required will be PINCONTROL2_BROADCAST=192.168.178.50

, if you need anything else you will have been given that parameter value with your board.

### *PIXELCADE_GAME_START_HIGHSCORE=1/0*

Default 0, can be left out of the INI

Set this to 1 DOFLinx will display MAME high scores when a game starts.

### *PIXELCADE_MENU_HIGHSCORE=1/0*

Default 1, can be left out of INI

When set to 1 then DOFLinx will display MAME game high scores on the Pixelcade device as you move through your selections on your front end menu.  Currently front end menu systems supported are – Pinup Popper.
When 0 / false then only the appropriate marquee will be displayed with no high scores.

### *PIXELCADE_REPLACE_LED_MARQUEE_AFTER_ANIMATION=1/0*

Default 0, can be left out of INI

Possibly the longest parameter ... although I'm sure I can beat it later!

1 = yes / true, 0 = false / no

Determines if DOFLinx will redisplay the MAME game marquee after an animation command in the game.MAME file is executed.  So if an explosion is shown when you hit a ship, do you want the marquee for the game to be displayed on the LED screen after its finished.
Having this on makes sense when you only have an LED screen.  If you have an LCD and LED then the marquee will already be on the LCD, so it makes sense to keep the LED for animations only and not duplicate the marquee.  But then, here we go, its your choice!

### *PROCESSES=PPPP,PPPP,PPPP*

Default Pinball FX2, can be left out of INI file.

This is the name(s) of the process(es) that will "wake up" DOFLinx.  By default, it is 'Pinball FX2' only. When the process goes away again DOFLinx goes back into its idle state.

The process names are as they can be seen in Task Manager.  For example

```
PROCESSES=Pinball FX2,Future Pinball,Pinball FX3,PinballFX-Win64-Shipping
```

### *PROCESS_ADD=*
No default, can be left out of INI file.

A comma separated list of process names to add to the PROCESS= set while DOFLinx is running.

### *PROCESS_DEL=*
No default, can be left out of INI file.

A comma separated list of process names to remove from the PROCESS= set while DOFLinx is running.

### *PLUNGER_AXIS=A*
No default, can be left out of INI file.

"A" is the axis to read the plunger position from. At present the only valid option is "Z" (no quotes) indicating the Z-axis for a plunger. This is used to allow a plunger to trigger ball launch in FX3.

### *PLUNGER_JOYSTICK_NUMBER=XX*
No default, can be left out of INI file.

Valid values are 1 to 16 being the joystick ID # that the plunger device is connected to.

### *PLUNGER_KEY=KK*
No default, can be left out of INI file.

The key number to send to FX3 when the plunger passes the PLUNGER_PULL_POINT and then returns back past the PLUNGER_RELEASE_POINT.

To send the default launch key of FX3 being <ENTER> this line should read PLUNGER_KEY=0D

### *PLUNGER_PULL_POINT=NNNNN*
No default, can be left out of INI file.

If PLUNGER_AXIS and PLUNGER_JOYSTICK_PORT are set, this is to point at which the plunger being pulled is registered. Values can be determined from the Pinscape setup tool or DOFLinx in DEBUG mode using the Test Joystick function.

If your plunger has a Z-Axis range of 0 at rest through to 65000 at full pull, then PLUNGER PULL_POINT could be 40000 and PLUNGER_RELEASE_POINT=20000. You will need to test to determine the best values. There should be a reasonable gap between PLUNGER_PULL_POINT and PLUNGER_RELEASE_POINT to avoid accidental triggering.

### *PLUNGER_RELEASE_POINT=NNNNN*
No default, can be left out of INI file.

See PLUNGER_PULL_POINT

### *PROCESS_TO_ACTIVE_TIME=MMMM*
Default is 2000ms, can be left out of INI file.

This allows you to customise the time between the detection of a process to wake up DOFLInx as defined in PROCESSES= , and when DOFLinx 'wakes up'. The reason to tweak this is to allow for odd start-up processes whereby the PROCESSES process is detected quickly but configuration messages are taking a while to be sent through by something like the DOFLinxMsg utility as a 'Launch Before' process. If in doubt, leave it alone.

### *QUIT_AFTER_PROCESS=[0|1]*
Default is 0, can be left out of INI file. 0=off, 1=on.

When turned on, DOFLinx will shutdown after the monitored process that woke it disappears. This can be used in Launch Before mode to stop DOFLinx without needing to be killed by batch file, etc.

So if you are monitoring the default process of 'Pinball FX2' and start up DOFLinx via a Launch Before option, with the parameter set to 1, DOFLinx will shutdown when Pinball FX2 stops and you return to your menu, ie PinballX.

This mode of operation is not recommended.

### *R_FLIPPER_KEY=KK*
See L_FLIPPER_KEY and just think right.

### *R_FLIPPER_OUTPUT=DOOO*
See L_FLIPPER_OUTPUT and just think right.

### *RGB_MIN_TIME=SSSS*
Default is 5000, this parameter can be left out of the INI file.

The time, in milliseconds, used by the RGB_STYLE to change RGB_OUTPUT.

### *RGB_OUTPUT=DOOO,DOOO,DOOO*
No default, leave if out or blank if you don't want or have RGB devices.

A comma separated list of DOOO for the RGB devices you want to cycle through colours. The DOOO is the port of the red device, the Green is assumed to be DOOO + 1 and the Blue assumed to be DOOO + 2.

The RGB_OUTPUT sets RGB colours from the entire, or GAME_COLOUR palette, according to RGB_STYLE, RGB_TRIGGER and RGB_MIN_TIME parameters.

### *RGB_STYLE=[RAINBOW|RAINBOW_ALTERNATE|RANDOM|CCCC]*

Default is RAINBOW, this can be left out of the INI file.

This parameter determines how colours are selected for the RGB_OUTPUT devices.

RAINBOW cycles through the current colour palette list from beginning to end. So that could be the default entire colour palette, or the sub-set defined by GAME_COLOUR.

RAINBOW_ALTERNATE cycles through the current colour palette list from beginning to end. So that could be the default entire colour palette, or the sub-set defined by GAME_COLOUR. Uses the next two colours from the palette and applies then alternately to each RGB device in the list. If the two colours were red and blue, and you had four RGB devices then device #1 and #3 would be red and #2 and #4 would be blue.

RANDOM, as the name implies, picks the next colour at random from the current colour palette.

RANDOM_DIFF, picks a new random colour from the current colour palette that is different for each RGB device provided.

RAINBOW_DIFF, picks a new random colour from the entire colour range that is different for each RGB device provided.

CCCC, is any valid colour name in the entire colour palette. This has the effect of setting the RGB output to a fixed colour. Note you must set the RGB_TRIGGER to FIXED

### *RGB_TRIGGER=[FLIPPER|TIME|FIXED]*

Default is TIME, this parameter can be left out of the INI file.

This parameter is not relevant if RGB_STYLE has been set to CCCC being a fixed colour as there is nothing to change.

When set to FLIPPER, at each flipper press, not more often than RGB_MIN_TIME, the RGB_OUTPUT colour will change using the RGB_STYLE.

When set to TIME, the RGB_OUPUT colour will change ever RGB_MIN_TIME.

When using a fixed colour via RGB_STYLE=CCCC set this parameter to FIXED.

### *SCORE_SCREEN=*

Default is 0, off

Seet to 1 to enable the displaying of the building MAME score screen. This screen can then be dragged to any monitor or position.

Note, this is not related to displaying scores on Pixelcade devices.

### *SETTING=IIII,DOOO,LLLL,DDD,S[,DOOO,PPP]*

No default, only required if using the settings functionality.

This item is repeated for each setting required. Settings are displayed in the order they appear in the INI file.

IIII is a unique identifier for the setting. This is used internally to differentiate each setting, as well as for the suffix in the registry if the setting is saved. It can be any length, but probably best to keep it below 10 characters. There is a special unique identifier being “NIGHT”, this does not require a DOOO setting, it allows the toggling of DOFLinx’s night mode setting

DOOO is the device port to turn on / off for this setting. In the case of the special identifier “NIGHT”, just leave the field empty

LLLLL is the label that appears on screen for this setting. For example, “Under cabinet lighting”.

DDD is the default setting, either “ON” or “Off” without the quotes

S is either “Y” or “N” without the quotes indicating if this setting should be saved upon exit and reloaded in the same state next time the system starts.

If an optional second output device / port and pulse width (PPP) is provided, then this setting is treated as a latch based setting. That means the first port will pulse to turn the setting device on, and the second port will pulse to turn the setting device off. This sort of setting device requires a latching relay, so a relay with some additional circuitry. You use a latching relay circuit when you need your setting device’s status to survive the DOF framework outputs being reset due to different programs taking control.

A latching relay circuit is ideal when you have a DOFLinx settings item turning the power on / off to internal devices. In my case the power supply to my addressable LED array. The power being on / off needs to persist programs starting stopping that may impede the use of a simple single output being turned on to control the power relay.

The pulse (PPP) for setting / unsetting the latch is in milliseconds. It shouldn’t really be larger than 100, that is 1/10th of a second.

### *SETTING_ACTIVATE=KK[,KK,KK]*

No default, only required if using the settings functionality.

One or more key codes used to activate the settings pop up menu. If more than one key code is used then all keys must be pressed at the same time to achieve activation. For example SETTING_ACTIVATE=A0,A1,A2,A3 would require four keys to be held down simultaneously being left shift, right shift, left control, right control. For many this would be the pressing of both flipper and magna save buttons.

### *SETTING_CHANGE=KK*

No default, only required if using the settings functionality.

The key code identifying the key used to change setting values.

### *SETTING_DOWN=KK*

No default, only required if using the settings functionality.

The key code identifying the key used to move down in the settings menu. Ie SETTING_DOWN=A1 would set the right shift, often the right flipper, for this task.

### *SETTING_EXIT=KK*

No default, only required if using the settings functionality.

The key code identifying the key used to exit the settings menu.

### *SETTING_UP=KK*

No default, only required if using the settings functionality.

The key code identifying the key used to move up in the settings menu. Ie SETTING_UP=A0 would set the left shift, often the left flipper, for this task,

### *SETTING_MESSAGE=TTTTTTT*

No default, only required if using the settings functionality.

### *SKIP_RGB_FLASHER_TEST=[0|1]*

The default is 0, meaning the test is performed.

Normally you would not add or use this parameter.  It is for a specific override of the test that removes flashers (LINK_FLxx) from the RGB_OUTPUT list.  In some cases you can define devices output devices as both RGB and flashers.  Normally you want DOFLinx to make a device only operate as either a RGB device or flasher device.  So don't use it unless you really know why.

### *SSF_ENABLED=[0|1]*

The default is 0 and this can be left out of your INI file

For enabling / disabling SSF, 0 is disabled, 1 is enabled.

### *SSF_DEVICE=N*

The default is -1 and this can be left out of your INI file.  This is the number of the output sound device according to Windows.  Using -1 gives you the default Windows output sound device.  Otherwise use a valid output device number, this is usually done if you are not using the default output sound device.  With DEBUG= set to 1 or 2, DOFLinx will display your output sound devices, their number and what is the default device.

## *SSF_SOUND_PATH=*

This is the path to the sounds used by the LOAD_SOUND= command in your table.FX file. The DOFLinx installation zip places the sounds setup by default in the \sounds folder.

## *TRIGGER_DEBOUNCE_TIME=*

The default is 100Ms. Only required if you want to set it away from the default.

This parameter sets the minimum time between triggers. This was added for FX3 where some very quick retriggering for things like flippers was being generated. It stops the same trigger from causing an action within the time set.

## *XBOX_RUMBLE= XX,C,M,SS,DDD[,XX,C,M,SS,DDD]*

No default, only required if using this functionality.

This functionality allows Xbox rumble controls to be driven from linked toy events in FP and FX2/3. By setting this parameter you can drive the selected Xbox motor on the selected controller at any valid speed for a settable number of milliseconds.

XX is the toy trigger event, valid events are;

LF = Left flipper
RF = Right flipper
LS = Left slingshot
RS= Right slingshot
ML = Mid field left solenoid
MC = Mid field centre solenoid
MR = Mid field right solenoid
BL = Back left solenoid
BC = Back centre solenoid
BR = Back right solenoid
SH = Shaker motor
GR = Gear motor
KN = Knocker
FN = Fan

C is the Xbox controller from 1 to 4. You can use the GamePad test screen in DOFLinx (available when in DEBUG and visible) to determine the controller number.

M is the motor, so L, R or B for both.

SS is the motor speed. Valid values for an XBox motor speed are 0 to 65535 (100%)

DDD is the duration in milliseconds.

XBOX_RUMBLE=LS,1,L,50000,1000,RS,1,R,50000,1000 will turn on either the left or right rumble motor to a speed of 50000 (about 75%) for 1 second when the left or right slingshots are hit.

Note: For the various ‘colour’ based commands the special colour name “Random” can be used. This will cause a random colour from the current colour palette to be displayed. The current colour palette is by default all colours unless a set of GAME_COLOUR= has been set up, then that is the current colour palette.

## Real Time Parameters and Commands

A number of commands and many parameters can be changed while DOFLinx is running. This is made possible by sending commands via Windows inter-process communication. A small utility DOFLinxMsg.exe is included as one way to do this. Another way is a simple VBS script, again an example is included.

### *Real-Time Commands*

These commands allow you to control DOFLinx or perform some action outside of those setup per game to occur. Commands are sent from the messaging utility like:

```
DOFLinxMsg QUIT
```

Multiple commands can be placed on one line. They need to be separated by a #. So for some colour and a shaker:

```
DOFLinxMsg OUTPUT_NOW_TIMER=115,1500#OUTPUT_COLOUR_NOW=109,5000,Yellow
```

The commands are:

#### ATTRACT_MODE=0/1

Turn attract mode on / off.

#### GAME_FLAG_FILE=PPPP\FFFF

PPPP is the path and FFFF is the file.
When an emulator process is in the PROCESSES list and a GAME_FLAG_FILE is provided, DOFLinx will only wake up if the file name provided, with the extension “.DOFLINX” exists.
This allows for emulators that have some tables correctly DOF’ed to be added to the PROCESSES list. Then in the ‘Launch Before’ section you run something like:

```
DOFLinxMsg GAME_NAME_FILE=[TABLEPATH]\[TABLENAME]
```

If you then create a file in your tables folder with the same name as the table you want DOFLinx to wake up for with the extension of “.DOFLINX” then it will provide rudimentary DOF with no additional effort. Of course, I do encourage people to continue to add DOF to VP and FP tables, that’s proper DOF. It is quite handy for custom tables, just to give them a little more ‘feeling’.
This is designed for use when DOFLinx is running in Set and Forget mode.

#### LIGHT_BUTTONS_NOW

Used to immediately turn on all of the buttons that have been defined by BUTTONS_LIT= . This is handy when you define some RGB buttons that you want to turn on when your front end is running but DOFLinx is not fully active because no emulator is running. Just issue the command via DOFLinxMsg or similar.

### KILL=XXXXX

This sends a Windows kill message to all instances of the process name provided. It does not wait for the process to exit, it just sends the message and continues.

```
DOFLinxMsg “KILL=Pinball FX2”
```

The double quotes are used on the above since the process has a space in its name. This sample command will send a kill message to all instances of Pinball FX2 running at the time. You would use this as Launch After command when exiting Pinball FX2 back to the menu if you want to ensure that all instances are closed. I use it for those occasions when PinballX doesn’t manage to close things on exit.

### OUTPUT_NOW_TIMER=DOOO,MMMM,DOOO,MMMM, etc

DOOO is the device / port combination, and MMMM is how long that output will be left on in milliseconds. So, “DOFLinxMsg OUTPUT_NOW_TIMER=1015,2000” will turn on device 1, output 15 for 2 seconds. This will occur regardless of DOFLinx being awake for a game or not, so you can have a DOF device triggered by anything you can think of and create a batch file for.

### OUTPUT_NOW_COLOUR=DOOO,MMMM,CCCC,DOOO,MMMM,CCCC, etc

DOOO is the Red device / port combination for the RGB device, MMMM is how long the RGB displays colour CCCC. This will occur regardless of DOFLinx being awake for a game or not.

### OUTPUTS_OFF

Turn off all outputs now. Handy if you have LED buttons that you turn on at different points and want them all off when you do something like exit your front end. This command can easily be issues vis DOFLinxMsg.

### QUIT

Shutdown DOFLinx immediately. This is much nicer than killing the process from the operating system. It allows DOFLinx to neatly close devices, pipes and return other resources to the operating system.

### STOP_DOF2DMD

Sends the shutdown command to DOF2DMD. This is far preferable to trying to kill the process. Likely used if you have a mixed arcade and pinball cabinet and want to stop DOF2DMD after you’ve played a MAME game.

### SUP_INI=

This is the same as the command line argument, but parsed in real-time. It allows a SUP_INI file to be processed without the need to stop and restart DOFLinx. The parameters processed in real-time must abide by the same rules as messages, ie BUTTONS_ON can only be changed while DOFLinx is asleep. If DOFLinx is awake when parsed this parameter, it will be ignored.

### SHOW_IN_TASKBAR=[0|1]

DOFLinx starts up with this set to 0 (Off)

Setting this parameter to 1 (On) will display DOFLinx in the Windows Taskbar rather than hide it completely. Very handy when you are playing around to get things working, etc.

SHOW_VARIABLES

Displays all the key variables in the log file to assist with debugging. This command takes a moment to process, so may cause a slight pause in operation.

### *Parameters That Can Be Changed Only When DOFLinx is Sleeping*

Setting these parameters via the messenger will replace the setting from the INI until you change it again or restart DOFLinx.

BUTTONS_ON
FORCE_ACTIVE
KEY_TO_COLOUR
KEY_TO_COLOUR_TIMER
KEY_TO_COLOUR_TOGGLE
KEY_TO_OUTPUT
KEY_TO_OUTPUT_TIMER
L_FLIPPER_KEY
L_FLIPPER_OUT
MAX_FLIPPER_ON
OUTPUT_GAME_NAME
R_FLIPPER_KEY
R_FLIPPER_OUT
RGB_MIN_TIME
RGB_OUTPUT
RGB_STYLE
RGB_TRIGGER
QUIT_AFTER_PROCESS

### *These parameters can only occur in the DOFLinx.INI file*

DIRECTOUTPUTCONFIG
GAME_COLOUR
PROCESSES

### *Sending Commands To DOFLinx*

Commands are sent to DOFLinx via a named pipe “DOFLINX”. This means you can use the utility provided (DOFLinxMsg), or anything else that will write to the named pipe. A file Sample.vbs is included that shows how to use VB Script to write directly to the named pipe. This allows a lot of flexibility in how you send commands to DOFLinx.

## Testing Your Setup

Of course you can dive straight in, fire it up and kick off your emulator. This can be a bit confusing, and the recycle time to change and test again can get annoying. Instead set DEBUG=1, start-up DOFLinx and press the “Fake Emulation On” button, it will now change to a “Fake Emulation Off” button.

Once activated you can try out all of your keys, colours, etc.  If you need to change things, just use "Quit", edit your INI and go again.  When you're done, edit your INI back to DEBUG=0.

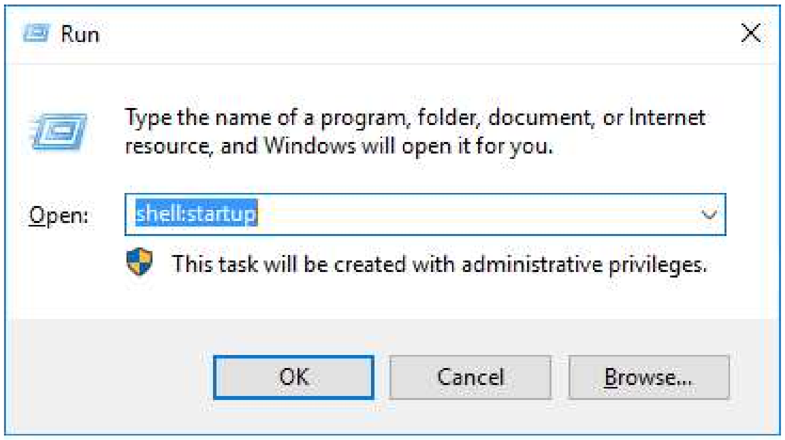
## Other Use~~less~~ful Info

- DOFLinx was written by DDH69 in 2016 because he wanted his force feedback flippers to work within Pinball FX2
- All of this made possible by DOF – thanks Swisslizard
- The main discussion on this utility happens over at www.vpforums.org
- SainSmart support testing by doogie2301
- Numerous (OK, lots) of good ideas and testing from TerryRed
- PacLed64 testing by GInsonic
- KL25Z emulating LEDWiz testing by davidlinch and Mace
- Inspiration and encouragement for full Pinball FX2 force feedback from gigalula
- Pinscape DOF Framework coding by mjr
- Pinscape native testing by me
- Ultimate I/O DOF Framework coding by rambo3
- Ultimate I/O testing by BritXpatUSA
- B2S testing by Outhere and Terryred
- Addressable LED work with TerryRed
- Most of the hard work for FX3 by gigalula – thank you
- PinControl2 testing by Charly Harper and BambiPlattfuss
- Some FX3 keyboard simulation ideas by Tom Speirs
- PinControl1 testing by Charly Harper
- PacDrive testing by Bjonne
- Zen and freezy for PinballFx integration
- This is not for commercial use.  If you have needs, make contact.

## Examples & Screen Shots

### *Setup as 'Set and Forget'*

Go to your start-up folder



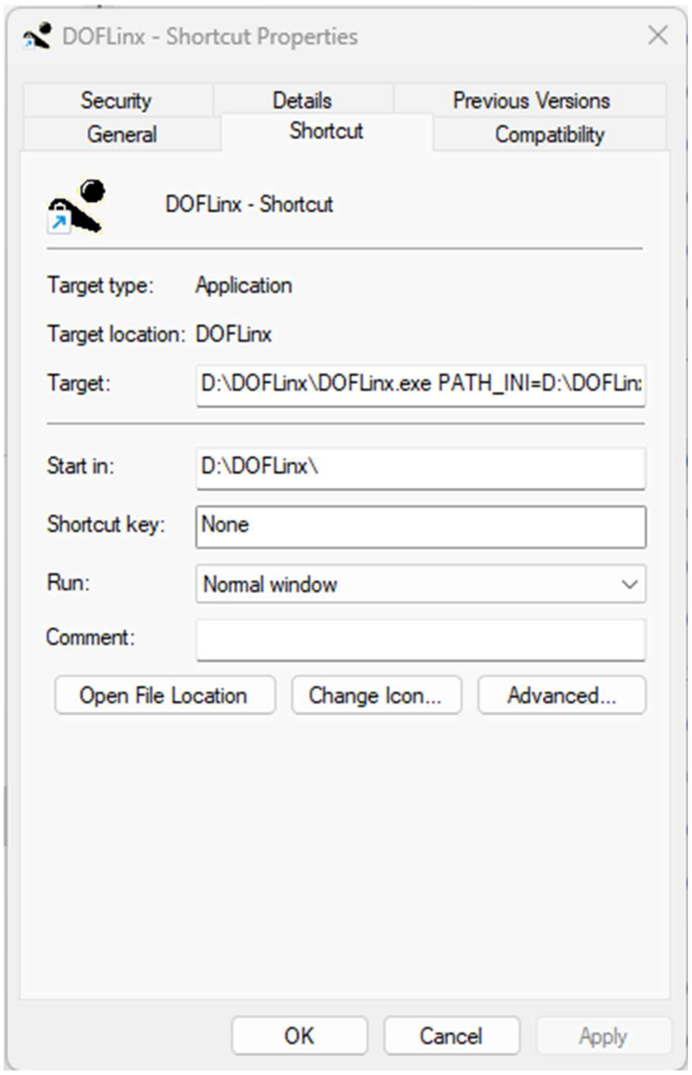
Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: shell:startup

This task will be created with administrative privileges.

Create a shortcut



DOFLinx - Shortcut Properties

DOFLinx - Shortcut

Target type: Application

Target location: DOFLinx

Target: D:\DOFLinx\DOFLinx.exe PATH_INI=D:\DOFLin

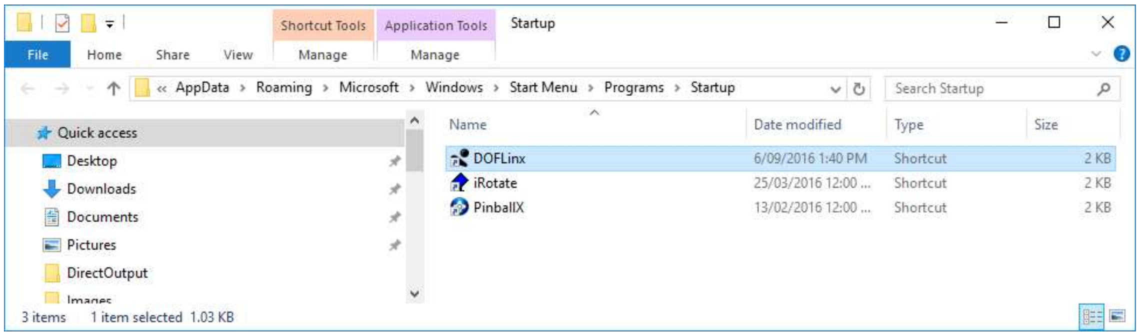
Start in: D:\DOFLinx\

Shortcut key: None

Run: Normal window

Comment:

End up with it in your start-up items

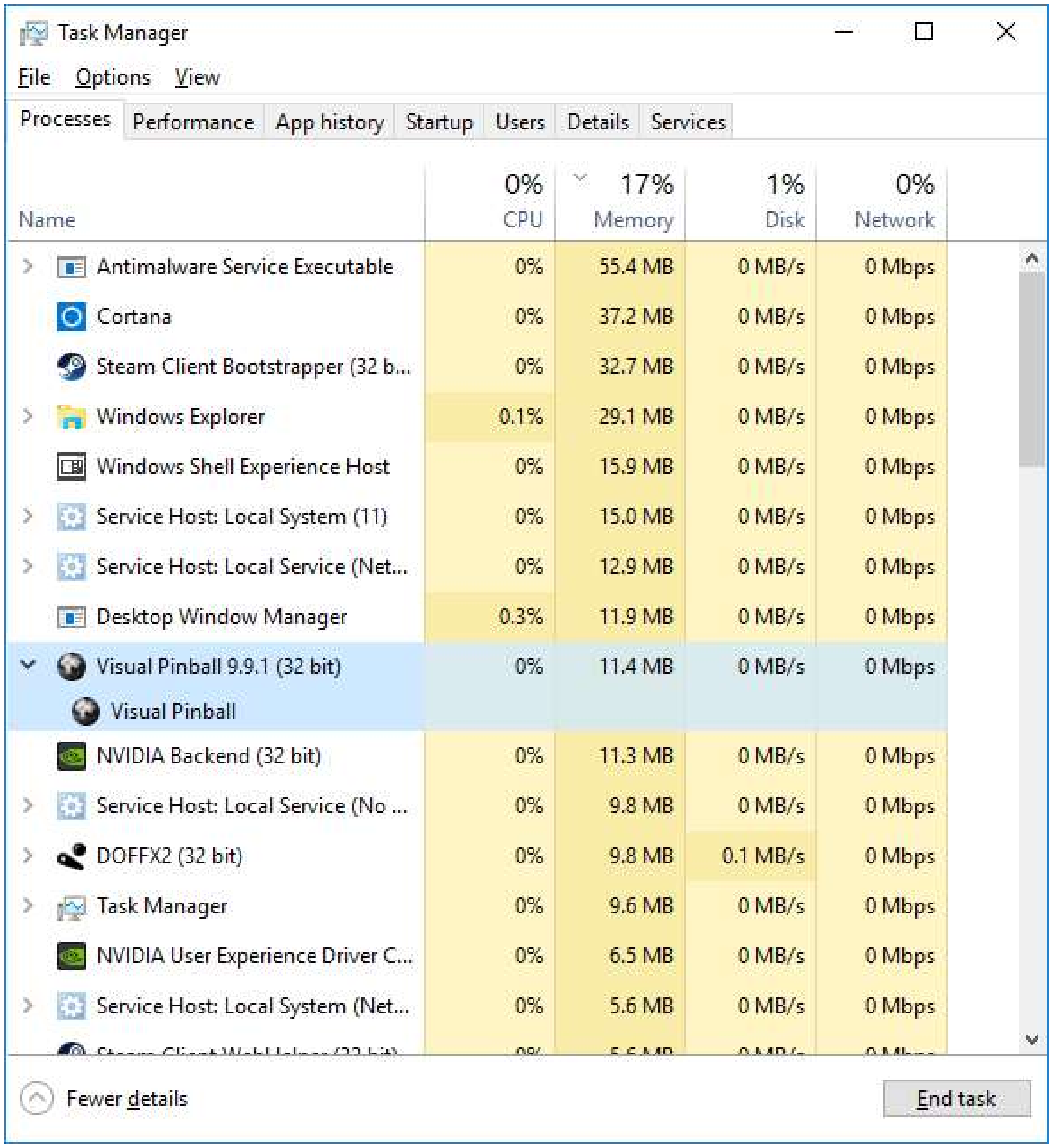


| Name | Date modified | Type | Size |
|---|---|---|---|
| DOFLinx | 6/09/2016 1:40 PM | Shortcut | 2 KB |
| iRotate | 25/03/2016 12:00 ... | Shortcut | 2 KB |
| PinballX | 13/02/2016 12:00 ... | Shortcut | 2 KB |

## *Where to Find the Names for the "PROCESSES=" Parameter*

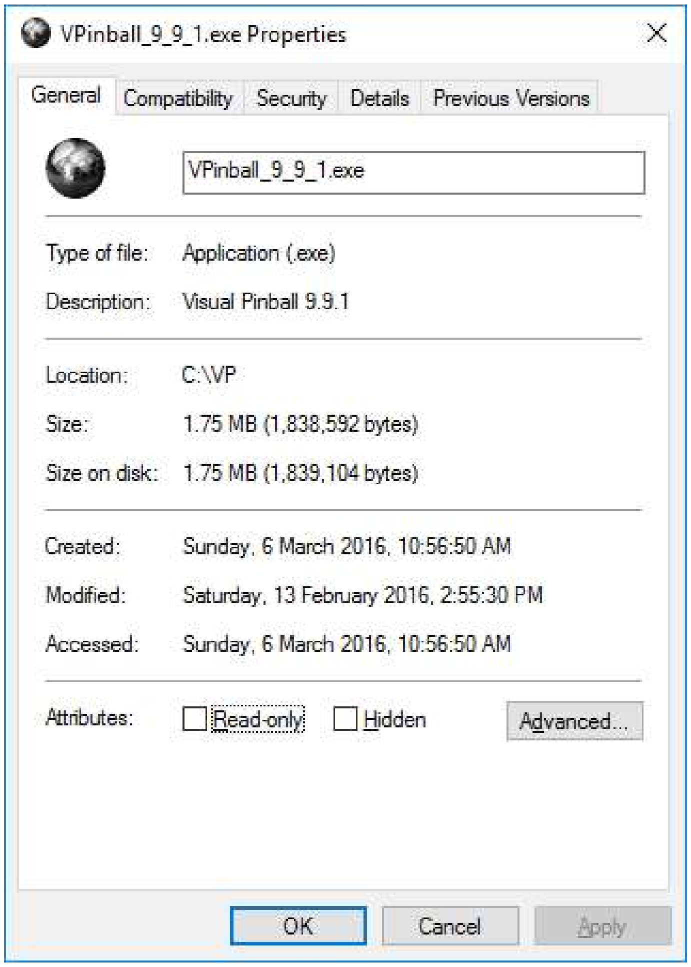
There are two ways to do this;

1. Via DOFLinx itself
   a. Start DOFLinx with DEBUG=1
   b. Right click on the message logging list box
   c. Select the "Show Running Processes" menu option
   d. Note down the relevant name from the list
2. Look in Task Manager as per the images below;


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

| Name | 0% CPU | 17% Memory | 1% Disk | 0% Network |
|---|---|---|---|---|
| Antimalware Service Executable | 0% | 55.4 MB | 0 MB/s | 0 Mbps |
| Cortana | 0% | 37.2 MB | 0 MB/s | 0 Mbps |
| Steam Client Bootstrapper (32 b... | 0% | 32.7 MB | 0 MB/s | 0 Mbps |
| Windows Explorer | 0.1% | 29.1 MB | 0 MB/s | 0 Mbps |
| Windows Shell Experience Host | 0% | 15.9 MB | 0 MB/s | 0 Mbps |
| Service Host: Local System (11) | 0% | 15.0 MB | 0 MB/s | 0 Mbps |
| Service Host: Local Service (Net... | 0% | 12.9 MB | 0 MB/s | 0 Mbps |
| Desktop Window Manager | 0.3% | 11.9 MB | 0 MB/s | 0 Mbps |
| Visual Pinball 9.9.1 (32 bit) | 0% | 11.4 MB | 0 MB/s | 0 Mbps |
| Visual Pinball | | | | |
| NVIDIA Backend (32 bit) | 0% | 11.3 MB | 0 MB/s | 0 Mbps |
| Service Host: Local Service (No ... | 0% | 9.8 MB | 0 MB/s | 0 Mbps |
| DOFFX2 (32 bit) | 0% | 9.8 MB | 0.1 MB/s | 0 Mbps |
| Task Manager | 0% | 9.6 MB | 0 MB/s | 0 Mbps |
| NVIDIA User Experience Driver C... | 0% | 6.5 MB | 0 MB/s | 0 Mbps |
| Service Host: Local System (Net... | 0% | 5.6 MB | 0 MB/s | 0 Mbps |

Locate the process you want to monitor, right click and use the name as you see below (VPinball_9_9_1) without the EXE. Nb, your name is likely to be different, this is just how I have VP version 9.9.1 named.



VPinball_9_9_1.exe Properties

VPinball_9_9_1.exe

| | |
|---|---|
| Type of file: | Application (.exe) |
| Description: | Visual Pinball 9.9.1 |
| Location: | C:\VP |
| Size: | 1.75 MB (1,838,592 bytes) |
| Size on disk: | 1.75 MB (1,839,104 bytes) |
| Created: | Sunday, 6 March 2016, 10:56:50 AM |
| Modified: | Saturday, 13 February 2016, 2:55:30 PM |
| Accessed: | Sunday, 6 March 2016, 10:56:50 AM |
| Attributes: | ☐ Read-only ☐ Hidden |

## *Add Flipper Force Feedback to Selected VP Tables – not needed for most installations*

Ensure you have your VP process as one of the processes in your PROCESSES= parameter, ie "PROCESSES=Pinball FX2,VPinball_9_9_1". In fact, it is more likely to be the process name for your old versions of VP since almost everything VPX is DOF enabled (yay!)

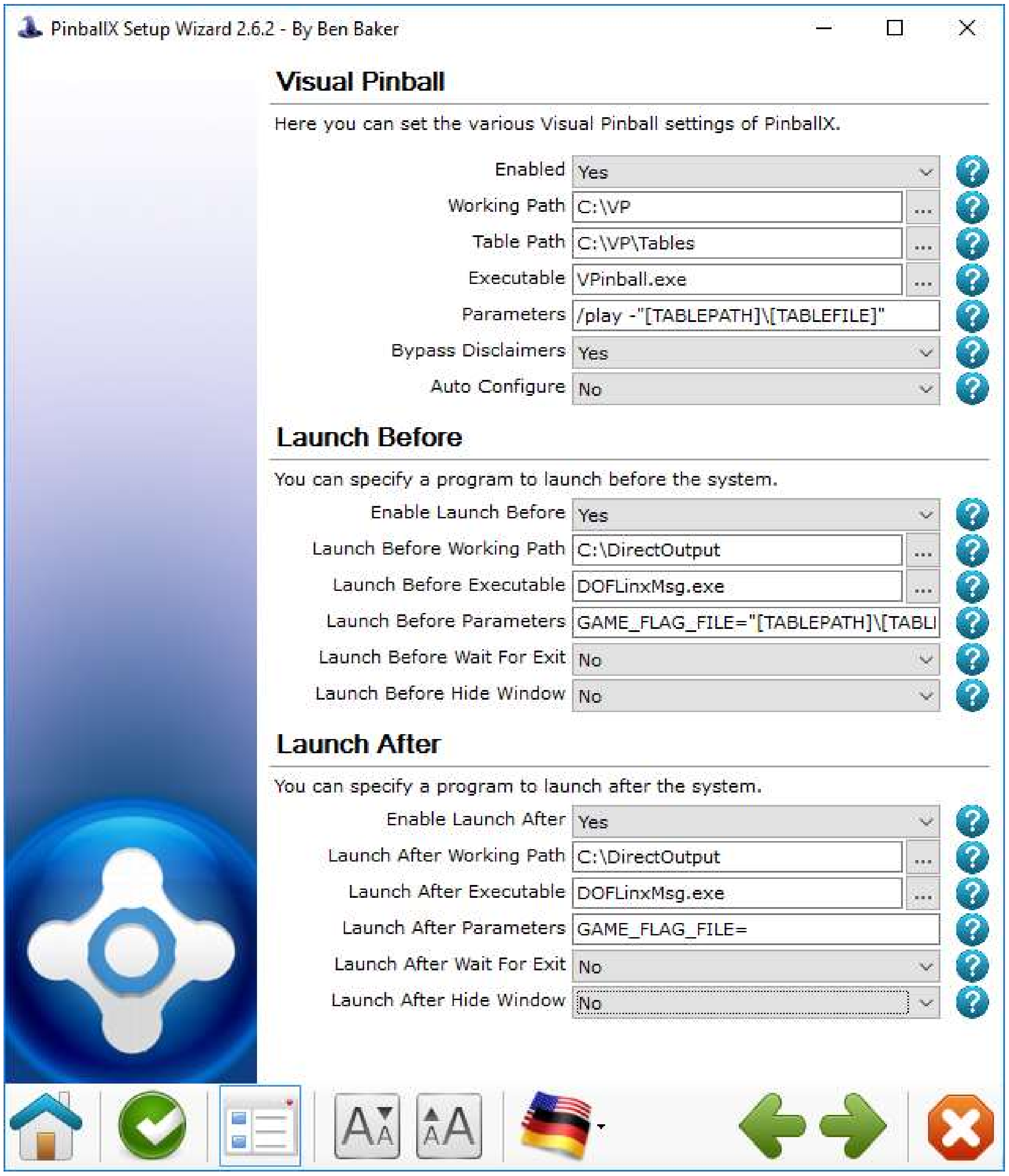
Setup your 'Launch Before' and 'Launch After' in your favourite front end like the sample below. You can send messages to DOFLinx in multiple ways, below shows it done using the small utility provided.

Create a dummy file named "TABLEFILE.DOFLINX" in your VP tables folder. The "TABLEFILE" name is the same name as you VPT or VPX, and B2S file for that table. This can be seen in the example below.

The logic is:

a) DOFLinx is running in Set and Forget mode
b) The 'Launch Before' program tells DOFLinx the current GAME_FLAG_FILE to look for
c) When the VP process starts DOFLinx checks if the flag file exists
d) If a flag file is set to something, then one of two things can happen;
    a. The file name provided exists, in which case DOFLinx 'wakes up'
    b. The file name provided does not exist, in which case DOFLinx stays 'sleeping'

The end result is that DOFLinx runs in the background at all times monitoring for VP, but only selectively 'wakes up' when a flag file is present.  This allows VP to be monitored at all times and wake as required.


PinballX Setup Wizard 2.6.2 - By Ben Baker

## Visual Pinball

Here you can set the various Visual Pinball settings of PinballX.

| | |
|---|---|
| Enabled | Yes |
| Working Path | C:\VP |
| Table Path | C:\VP\Tables |
| Executable | VPinball.exe |
| Parameters | /play -"[TABLEPATH]\[TABLEFILE]" |
| Bypass Disclaimers | Yes |
| Auto Configure | No |

## Launch Before

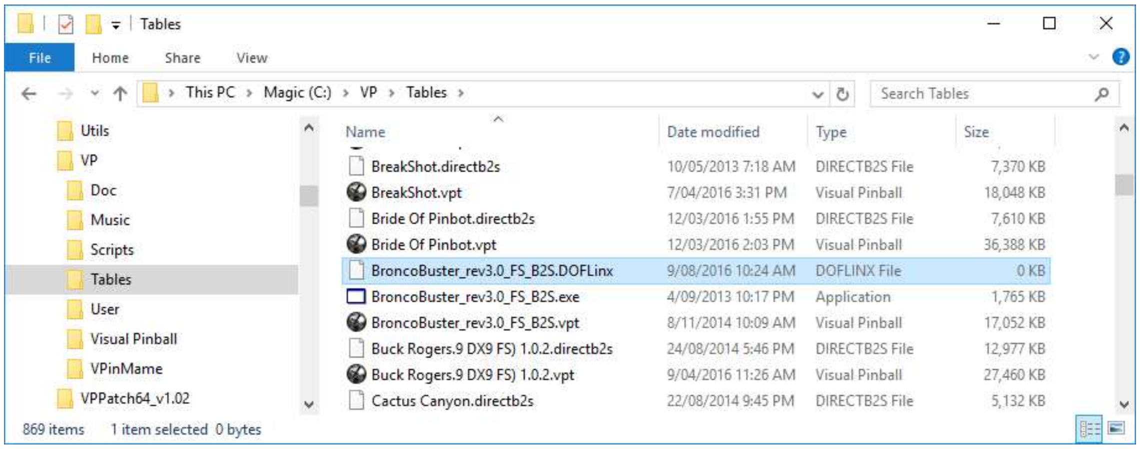
You can specify a program to launch before the system.

| | |
|---|---|
| Enable Launch Before | Yes |
| Launch Before Working Path | C:\DirectOutput |
| Launch Before Executable | DOFLinxMsg.exe |
| Launch Before Parameters | GAME_FLAG_FILE="[TABLEPATH]\[TABLI |
| Launch Before Wait For Exit | No |
| Launch Before Hide Window | No |

## Launch After

You can specify a program to launch after the system.

| | |
|---|---|
| Enable Launch After | Yes |
| Launch After Working Path | C:\DirectOutput |
| Launch After Executable | DOFLinxMsg.exe |
| Launch After Parameters | GAME_FLAG_FILE= |
| Launch After Wait For Exit | No |
| Launch After Hide Window | No |



Utils
VP
Doc
Music
Scripts
Tables
User
Visual Pinball
VPinMame
VPPatch64_v1.02

| Name | Date modified | Type | Size |
|---|---|---|---|
| BreakShot.directb2s | 10/05/2013 7:18 AM | DIRECTB2S File | 7,370 KB |
| BreakShot.vpt | 7/04/2016 3:31 PM | Visual Pinball | 18,048 KB |
| Bride Of Pinbot.directb2s | 12/03/2016 1:55 PM | DIRECTB2S File | 7,610 KB |
| Bride Of Pinbot.vpt | 12/03/2016 2:03 PM | Visual Pinball | 36,388 KB |
| BroncoBuster_rev3.0_FS_B2S.DOFLinx | 9/08/2016 10:24 AM | DOFLINX File | 0 KB |
| BroncoBuster_rev3.0_FS_B2S.exe | 4/09/2013 10:17 PM | Application | 1,765 KB |
| BroncoBuster_rev3.0_FS_B2S.vpt | 8/11/2014 10:09 AM | Visual Pinball | 17,052 KB |
| Buck Rogers.9 DX9 FS) 1.0.2.directb2s | 24/08/2014 5:46 PM | DIRECTB2S File | 12,977 KB |
| Buck Rogers.9 DX9 FS) 1.0.2.vpt | 9/04/2016 11:26 AM | Visual Pinball | 27,460 KB |
| Cactus Canyon.directb2s | 22/08/2014 9:45 PM | DIRECTB2S File | 5,132 KB |

869 items    1 item selected 0 bytes

## Future Pinball Link

With the addition of some code Future Pinball (FP) can be set to message DOFLinx directly.  This allows FP tables to be DOF'ed.

### *Assumptions*

a) DOFLinx is installed and working

*b)* Future Pinball is installed and working

### *How does this work?*

DOFLinx detects the FP process, then it waits for the FP table to initialise.  If the FP table has DOFLinx code setup within it then a flag is set in memory that DOFLinx reads.  DOFLinx then sets its own flag for FP that details the capability, ie shaker, gear, knocker.  FP then sends messages by setting memory values to DOFLinx.  DOFLinx continuously monitors for these messages while the FP process remains active and processes them as appropriate.

DOFLinx has parameters to identify the various solenoids, contactors and toys.  FP messages DOFLinx by function, not device / port.  In other words, FP would send a message like Left Flipper on for some time.  Its then up to DOFLinx to interpret and process that message.  As such DOFLinx must know what port, etc to use for each device.

Multiple versions of FP have been tested.  Any version should work.  No changes have been made to FP to allow this to work.

In a slightly more technical sense, when enabled via the FP_ATTEMPT_LINK=1 and with FP listed in the PROCESSES=  parameter, ie "PROCESSES=Pinball FX2,Future Pinball" DOFLinx will look for a link to FP when the FP process starts.  If FP_ATEMPT_LINK=0 then DOFLinx will operate as normal and listen to the keyboard keystrokes.

If the link is set to run within DOFLinx but not set up within the FP table you are starting, then the link attempt will time out and DOFLinx will handle Future Pinball as per its other configuration.

To get a link to FP, the FP table must have the DOFLinx code inserted into it and activated.  Examples of how to do this are below.

## *How do I set up for this?*

To set yourself up do the following:

a) Ensure you have "Future Pinball" as one of your processes, ie "PROCESSES=Pinball FX2,Future Pinball" in your DOFLinx.INI file.
b) Add the parameter FP_ATTEMPT_LINK=1 to your DOFLinx.INI file
c) Setup all of your LINK_xx= parameters where xx=LF,RF,LS,RS,ML,MC,MR,BL,BC,BR,SH,GR,KN,ST,EB,CN,EX,LB,FN.SR,BK
d) Add DOFLinx.vbs to your FP scripts folder. If your folder structure was C:\FP\Tables then this folder would be C:\FP\Scripts.
e) Run your DOF'ed FP table.

Below is a sample section of the DOFLinx.INI file to configure the various devices. This is my setup, so be sure to edit it for your device / port numbers, best default times for devices and desired maximum on times.

```
FP_ATTEMPT_LINK=1
LINK_LF=117,50,10000
LINK_RF=125,50,10000
LINK_LS=118,50,500
LINK_RS=126,50,500
LINK_ML=127,50,500
LINK_MC=128,50,500
LINK_MR=129,50,500
LINK_BL=130,50,500
LINK_BC=131,50,500
LINK_BR=132,50,500
LINK_SH=115,1000,5000
LINK_GR=116,750,10000
LINK_KN=124,120,500
LINK_ST=102
LINK_EB=101
LINK_CN=106,105,103
LINK_EX=104
```

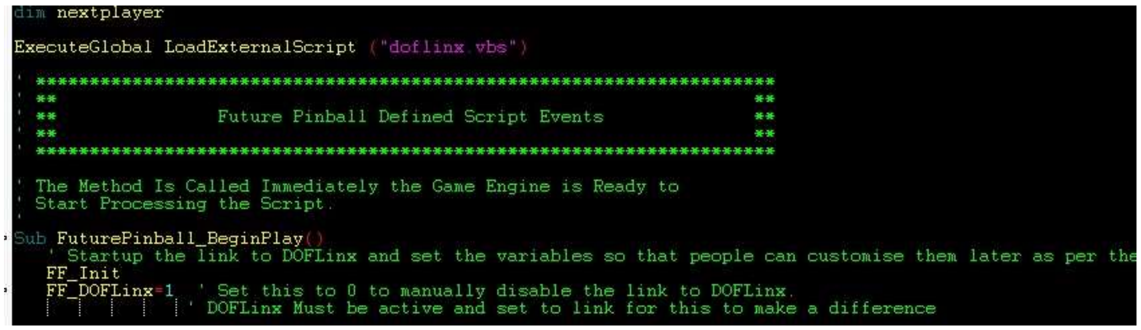
## If I want to DOF a table, how do I do it?

The FP code for DOF'ing a FP table is designed to be added in and left in regardless of DOFLinx being used. Ideally table authors will add this code to their tables in the same way it has been done for VP tables.

The mechanics of the process are:

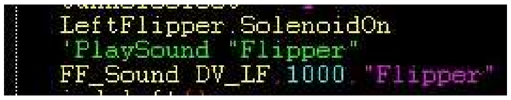
1. Directly below your "Dim" statements add the line "ExecuteGlobal LoadExternalScript ("DOFLinx.vbs")
2. As the first items in the standard FuturePinBall_BeginPlay() section add the lines "FF_Init" and "FF_DOFLinx=1"
3. Now do your coding.

The image below shows the above steps implemented.



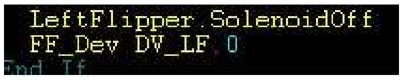
```
dim nextplayer

ExecuteGlobal LoadExternalScript ("doflinx.vbs")

' ***********************************************************************
' **                                                                   **
' **                Future Pinball Defined Script Events               **
' **                                                                   **
' ***********************************************************************

' The Method Is Called Immediately the Game Engine is Ready to
' Start Processing the Script.
'
Sub FuturePinball_BeginPlay()
   ' Startup the link to DOFLinx and set the variables so that people can customise them later as per the
   FF_Init
   FF_DOFLinx=1   ' Set this to 0 to manually disable the link to DOFLinx.
                 ' DOFLinx Must be active and set to link for this to make a difference
```

At its simplest replace “PlaySound xx” statements for flippers, slingshots, drop targets, bumpers and jets with “FF_Sound LF,-1,xx”. The format is “FF_Sound DD,HHHH,SSSSS” , SSSSS is the original sound. The original sound is played when the DOFLinx link is not active. DD is the device signified by the device code (constant in the DOFLinx.vbs file). HHHH is the time to turn the device on for in hundredths of seconds, the special value of -1 will use the default on time setup by the user in the DOFLinx.INI file.



```
LeftFlipper.SolenoidOn
'PlaySound "Flipper"
FF_Sound DV_LF,1000,"Flipper"
```

The image above shows the line that was, and the new line that replaces it. The left flipper is turned on for 10 seconds in this command. This will keep the left flipper on for a reasonable time or until the key is released and another “FF” command turns it off. If the user has set their maximum on time for this device to less than 10 seconds to protect their solenoids, then the left flipper solenoid will release sooner. It would be quite reasonable to set to time to 999999 upon flipper press.



```
LeftFlipper.SolenoidOff
FF_Dev DV_LF,0
End If
```

In the key released section, you then turn off the left flipper off.

For a simple flipper “bang” you could use time “-1” when turning the flipper on, and not turn the left flipper back off. This would use the users default on time for this device. Best practice is to turn it on then back off.

More complex programming can be performed using a few more supplied procedures and functions:

***Functions***

## FF_Active – Boolean

Used to test if the link from FP to DOFLinx is active. True when the link is active, false when it is not.

## FF_ISDevice(DD) – Boolean

Allows you to test if a specific device is configured for the cabinet this table is being played on. The function will return true if the device is configured.

Generally testing for a device is not required. If a message is sent for a device that does not exist, then the message will simply be ignored. Testing for devices is used by the supplied procedures FF_SOUND and FF_SOUNDVOL to determine if the device exists when determining if the original sound should be played or not.

So, for example, there is no need to test if the Coin button exists before setting it to flash. Just set it, and if it exists, it will flash, if not, nothing will happen.

Valid devices to test are:

DV_LF = Left flipper
DV_RF = Right flipper
DV_LS = Left slingshot
DV_RS= Right slingshot
DV_ML = Mid field left solenoid
DV_MC = Mid field centre solenoid
DV_MR = Mid field right solenoid
DV_BL = Back left solenoid
DV_BC = Back centre solenoid
DV_BR = Back right solenoid
DV_SH = Shaker motor
DV_GR = Gear motor
DV_KN = Knocker
DV_FN = Fan
DV_SR = Strobe
DV_BK = Beacon
DV_BE = Bell
DV_C1 = Chime 1 (high)
DV_C2 = Chime 2
DV_C3 = Chime 3 (low)
DV_RGB = RGB lighting
DV_FLOL = Flasher, outside left
DV_FLIL = Flasher, inside left
DV_FLCN = Flasher, centre
DV_FLOR = Flasher outside right
DV_FLIR = Flasher, inside right
BUT_ST = Start button
BUT_EB = Extra Ball button
BUT_EX = Exit button
BUT_CN = Coin button
BUT_LB = Launch ball button
BUT_FR = Fire button
BUT_P1 = Player 1
BUT_P2 = Player 2
BUT_PS = Pause

***Procedures***

### FF_Button DD,AA,IIIII,HHHHH

DD – Button device to activate

BUT_ST = Start button
BUT_EB = Extra Ball button

BUT_EX = Exit button
BUT_CN = Coin button
BUT_LB = Launch ball button
BUT_FR = Fire button
BUT_P1 = Player 1
BUT_P2 = Player 2
BUT_PS = Pause

AA - The action to perform

BA_ON =Turn the button LED on
BA_OFF = Turn the button LED off
BA_TT = Turn on the button LED for the supplied hundredths of a second
BA_FL = Flash the button LED for the number of iterations with the supplied time being the duration between iterations
BA_FD = Fade the button LED up and down for the number of iterations with the supplied time being the duration to fade up / down

IIIII – The number of iterations to perform the action. For flash and fade cycles on is one iteration and off is another. So, if you want the button to flash on then off its 2 iterations. This way you can flash or fade a button LED leaving it on or off when the cycle is finished.
HHHHH – Hundredths of seconds per iteration

## FF_Colour CCCCC,AA,HHHHHH

CCCCCC – a valid DOF colour name, ie Red, Light_cyan. "Random" is a valid colour name and will simply display a random colour from the currently active colour palette.
AA – the action to perform

RGB_DF = Start the default RGB colour cycle as set by the user in DOFLinx. This will give automatic colour changes based on a fixed user set colour, flipper initiated changes, or timed changes using either the full colour palette or a restricted palette, as per standard DOFLinx parameter and GAME_COLOUR settings. The colour name supplied (CCCCC) is irrelevant for this action. Once started for a game the default colour run cannot be stopped. It is an easy way to add colour with one command for the entire table.
RGB_TT = Change the RGB's to the supplied colour for the supplied time then revert to the colour displayed before the command.
RGB_CH = Change the colour to the new colour and leave it on until over written by further commands.

HHHHHH – hundredths of seconds to turn on the colour for. Maximum is 999999 or 9999 seconds.

## FF_Dev DD,HHHHHH – Activate a device for the specified time

Turn on the device code for the specified time. The special time of minus one (-1) will use the default run time set by the user in their DOFLinx.INI parameters. An on time of zero (0) will turn the device off from a previous on command if the time period has not yet elapsed.

This command is used when there is no sound required when the DOF link is not operational. It is the same as FF_Sound DD,HHHHHH, " "

## FF_DOF T,EEE,HHHHHH

This command triggers Direct Output Framework events as defined in the various DirectOutputxx.ini files. It is initially designed to trigger addressable LED commands, but there is no reason that any other “E” type command can’t be triggered.

T=Type, normally “E”
EEE = The event trigger number
HHHHHH = The duration to leave the trigger on in hundredths of seconds. Two special values exist, zero (0) will turn the trigger off, and minus one (-1) will turn the event on with no off time, normally for cases where the event has its own timer within DOF.

FF_DOF “E”,123,100 will turn on DOF event E123 for 1 second

Note you must have first opened a ROM via the FF_ROM command to use any FF_DOF commands.

## FF_Flasher DD,AA,IIII,HHHHH,PPP,CCCCC

DD- The device to activate.

DV_FLOL = Flasher, outside left
DV_FLIL = Flasher, inside left
DV_FLCN = Flasher, centre
DV_FLOR = Flasher outside right
DV_FLIR = Flasher, inside right

AA - The action to perform

FL_ON =Turn the flasher on
FL_OFF = Turn the flasher off
FL_TT = Turn on the flasher for the supplied hundredths of a second
FL_FL = Flash the flasher for the number of iterations with the supplied time being the duration between iterations
FL_FD = Fade the flasher up and down for the number of iterations with the supplied time being the duration to fade up / down

IIIII – The number of iterations to perform the action. For flash and fade cycles on is one iteration and off is another. So if you want the flasher to flash on then off its 2 iterations. This way you can flash or fade a flasher leaving it on when the cycle is finished.
HHHHH – Hundredths of seconds per iteration
PPP – The maximum percentage intensity 0 – 100. Ie, if set to 50 and fade (FL_FD) the flasher will fade up to 50% of the supplied colour and back to nothing.
CCCCC – The colour to use during the action. “Random” is a valid colour name and will simply display a random colour from the currently active colour palette.

## FF_Init – initialise the force feedback (FF) link from the FP side.

This command must be executed as early as possible in table start up to enable the link to function correctly.

## FF_PROC CCCCCC,AAAA

Used to run an external command, CCCCC, with argument, AAAA. Not used by core DOFLinx but available for use if you want to have a table trigger some sort of special external event via program or batch file.

## FF_PUPlayer CCCCC,SSSSSS,AAA

Used to run PUPlayer from FP. There are two CCCCC commands, PUP_INIT (1) and PUP_EVENT (2). You must issue a valid PUP_INIT in a table before you can issue PUP_EVENT commands. The PUP_INIT command should be placed in your timer to test once the DOFLinx link is established.

The SSSSSS string will be the ROM name for a PUP_INIT command and the Event code for a PUP_EVENT. Argument AAA is zero for PUP_INIT, and must be a valid argument for a PUP_EVENT from 0 – 255.

Sample commands are:

```
PUPlayer PUP_INIT,”trn_174h”,0
PUPlayer PUP_EVENT,”W11”,255
PUPlayer PUP_EVENT,”W44”,255
```

## FF_ROM RRRRRR

If you plan to use FF_DOF commands to trigger DOF events, ie addressable LEDs, then you must start a ROM in the Direct Output Framework (DOF) first. Normally the ROM started is not a real ROM like in VP, but a fake ROM setup in the DOF Directoutput30.ini file to support things like addressable LED commands.

The command is used as early in the FP scripts, but after FF_Init, and after the link is established. The best place is in the DOFLinx timer once the link is detected by the FP script as being in place. Sample syntax is;

FF_ROM “fp_motu”

See FF_DOF for  details on how to send DOF commands from FP scripts.

## FF_Sound DD,HHHHHH,SSSSS (replaces PlaySound SSSSS)

## FF_SoundVol DD,HHHHHH,SSSSS,VV (replaces PlaySound SSSSS,VV)

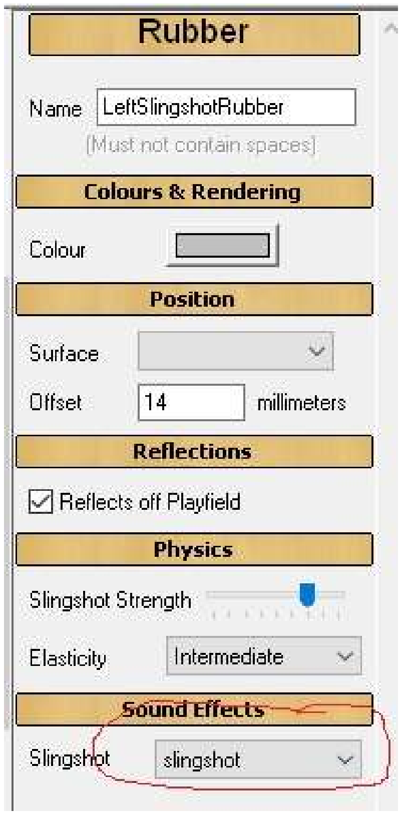
SSSSS – the sound to play when the FP to DOFLinx link is not active.
VV – The volume modifier for the standard PlaySound command
DD – the device to activate.

DV_LF = Left flipper
DV_RF = Right flipper
DV_LS = Left slingshot
DV_RS= Right slingshot
DV_ML = Mid field left solenoid
DV_MC = Mid field centre solenoid
DV_MR = Mid field right solenoid
DV_BL = Back left solenoid
DV_BC = Back centre solenoid

DV_BR = Back right solenoid
DV_SH = Shaker motor
DV_GR = Gear motor
DV_KN = Knocker
DV_FN = Fan
DV_SR = Strobe
DV_BK = Beacon
*Note: SH,GR and KN will play the supplied sound unless DOFLinx indicates the required devices are present. With an active link presence is determined by DOFLinx having been supplied valid devices LINK_SH, LINK_GR and LINK_KN via the DOFLinx.INI configuration file.*

HHHHHH – hundredths of seconds to turn on the device for. Maximum is 999999 or 9999 seconds.

Automatic sounds set on the object should not be used in this design. Doing so will cause the sound to play regardless of force feedback being active or not. For objects with sounds set you will need to remove the default sound and add the FF_Sound line, ie there is no PlaySound line to replace.


Rubber

Name: LeftSlingshotRubber
(Must not contain spaces)

Colours & Rendering

Colour

Position

Surface

Offset: 14 millimeters

Reflections

[✓] Reflects off Playfield

Physics

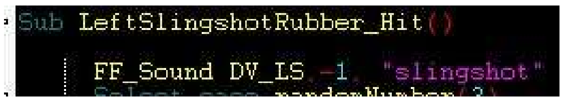
Slingshot Strength

Elasticity: Intermediate

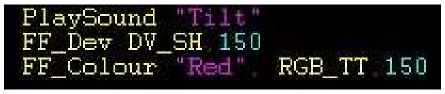
Sound Effects

Slingshot: slingshot

In this example the sound effect was “slingshot”, it should be set it to “none” and then add the line below in the “_hit” code.

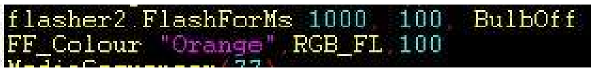

```
Sub LeftSlingshotRubber_Hit()
    FF_Sound DV_LS -1, "slingshot"
```

Actions can be added to other events on the table. For example, when the user tilts the following will run the shaker motor and flash the RGB devices red. The PlaySound "Tilt" was the original action. The new actions simply add to that.



```
PlaySound "Tilt"
FF_Dev DV_SH 150
FF_Colour "Red", RGB_TT 150
```

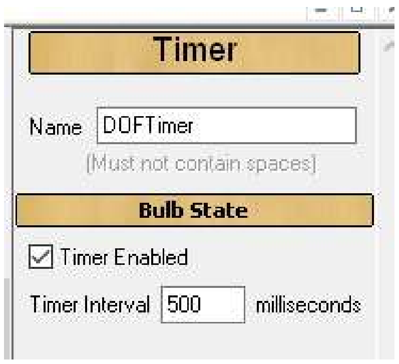
An example of complementing a playfield flasher with the RGB flashing the same colour for the same time is given below. Note the FF_Colour is measured in hundredths of seconds (value of 100 below) whereas the FlashForMs is in milliseconds (value of 1000 below).



```
flasher2.FlashForMs 1000, 100, BulbOff
FF_Colour "Orange",RGB_FL 100
```

If you want to set some force feedback items when a table starts (ie start colours, set button LEDs) you will need to set up a timer. The reason is that any force feedback control code, other than initialisation stuff, placed in the FuturePinBall_BeginPlay() section is likely to execute before the link has been properly established. Force feedback control code executed before the link is properly established will be effectively be ignored.

Doing this is simple, add a timer like this to the side of the playfield.



Timer

Name: DOFTimer

(Must not contain spaces)

Bulb State

☑ Timer Enabled

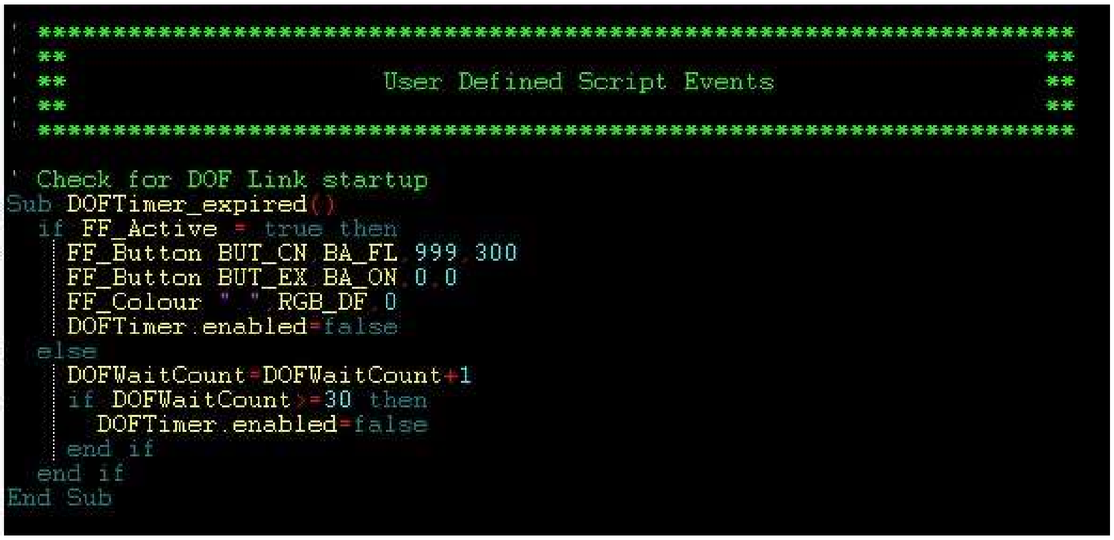
Timer Interval: 500 milliseconds

Then add some code like that that below. With the timer set to 500ms the table will test for an active link up to 30 times at ½ second intervals, meaning it will test for 15 seconds after the table starts. The link is normally up within 1 second, so a 15 second wait will be fine. Once a link is detected some actions can be performed and the timer is disabled for the rest of the game.

The lines to customise in the sample below are:

```
FF_Button BUT_CN,BA_FL,999,300
FF_Button BUT_EX,BA_ON,0,0
FF_Colour " ",RGB_DF,0
```

Of course you can add more lines of code for initialisation. Be careful to not remove the DOFTimer.enabled=false statement as you want the timer to stop once it has been executed once.



```
' *********************************************************************
' **                                                                 **
' **                     User Defined Script Events                  **
' **                                                                 **
' *********************************************************************

' Check for DOF Link startup
Sub DOFTimer_expired()
  if FF_Active = true then
    FF_Button BUT_CN,BA_FL,999,300
    FF_Button BUT_EX,BA_ON,0,0
    FF_Colour " ",RGB_DF,0
    DOFTimer.enabled=false
  else
    DOFWaitCount=DOFWaitCount+1
    if DOFWaitCount>=30 then
      DOFTimer.enabled=false
    end if
  end if
End Sub
```

## Pinball FX2 Memory Triggers – Full Force Feedback

If a specific FX2 table has been memory mapped, then full force feedback is possible. DOFLinx looks for a trigger configuration file named "GameName.FX2" first, then "default.fx2", if neither are found then basic keyboard monitoring will be used.

DOFLinx will first look in the PATH_FX2= path if configured, if not found there it will look in the DOFLinx.exe folder. It is recommended that you do set the PATH_FX2= parameter to a separate folder just for FX2 trigger files to keep things neat and tidy.

The trigger configuration file will look a bit like this:

```
##################
#
# Bob's Burgers Pinball
# DOFLinx force feedback configuration file v6
#
##################

[STARTUP]
FF_ROM=fx2_bb

[MEMORY TRIGGERS - CORE]
GAMEMODE=8879B4,FF
SCOREMODE=887919,FF
ANIMEMODE=88791B,FF
LAUNCHMODE=88791A,FF
LAUNCHBALL=883CF4,FF
LeftFlipper=88781C,FF
RightFlipper=88781C,FF00
Nudge=88B8F4,FF000000

[MEMORY TRIGGERS - TABLE BASED]
Left_Slingshot=882FC0,FF,2E4,0C,20,08
Right_Slingshot=882FC0,FF,2E4,14,20,08
Bumper1=882FC0,FF,2E4,00,20,08
Bumper2=882FC0,FF,2E4,04,20,08
Bumper3=882FC0,FF,2E4,08,20,08
Bumper1_Flasher=882FC0,FF,2B0,0C,00,04,A4
Bumper2_Flasher=882FC0,FF,2B0,0C,04,04,A4
Bumper3_Flasher=882FC0,FF,2B0,0C,08,04,A4
Left_SlingShot_Flasher=882FC0,FF,2B0,0C,0C,04,A4
Right_SlingShot_Flasher=882FC0,FF,2B0,0C,14,04,A4
Crane_Flasher=882FC0,FF,2B0,0C,1C,04,A4
FerrisWheel_Flasher1=882FC0,FF,2B0,0C,20,04,A4
FerrisWheel_Flasher2=882FC0,FF,2B0,0C,24,04,A4
Lamp1_Flasher=882FC0,FF,2B0,0C,28,04,A4
Lamp2_Flasher=882FC0,FF,2B0,0C,2C,04,A4
Lamp3_Flasher=882FC0,FF,2B0,0C,30,04,A4
Lamp4_Flasher=882FC0,FF,2B0,0C,34,04,A4
SinkHole_Flasher=882FC0,FF,2B0,0C,38,04,A4
WonderWharf_Left_Flasher=882FC0,FF,2B0,0C,3C,04,A4
WonderWharf_Right_Flasher=882FC0,FF,2B0,0C,40,04,A4

[COMMANDS]
GAMEMODE|ON|FF_Colour Black,RGB_DF,0|FF_Flasher DV_FLOL,FL_FD,6,750,50,Orange|FF_Flasher
DV_FLOR,FL_FD,6,750,50,Orange|FF_Flasher DV_FLIL,FL_FD,6,750,50,Yellow|FF_Flasher
DV_FLIR,FL_FD,6,750,50,Yellow|FF_Flasher DV_FLCN,FL_FD,6,750,50,Orange|FF_B2S B2SStartAnimation
Sign
LAUNCHMODE|ON|FF_Button BUT_LB,BA_FL,10000,500|FF_DOF E310,5000
LAUNCHMODE|OFF|FF_Button BUT_LB,BA_OFF,0,0
LAUNCHBALL|OFF|FF_Dev DV_MR,-1|FF_B2S B2SStartAnimation Sign|FF_DOF E310,0
SCOREMODE|ON|FF_Flasher DV_FLOL,FL_OFF,1,1,100,Random|FF_Flasher
DV_FLIL,FL_OFF,1,1,100,Random|FF_Flasher DV_FLCN,FL_OFF,1,1,100,Random|FF_Flasher
```

```
DV_FLIR,FL_OFF,1,1,100,Random|FF_Flasher DV_FLOR,FL_OFF,1,1,100,Random|FF_Dev DV_SR,0|FF_Dev
DV_BK,0|FF_B2S B2SStopAllAnimations|FF_DOF E415,-1
LeftFlipper|ON|FF_Dev DV_LF,10000
LeftFlipper|OFF|FF_Dev DV_LF,0
RightFlipper|ON|FF_Dev DV_RF,10000
RightFlipper|OFF|FF_Dev DV_RF,0
Nudge|ON|FF_Dev DV_SH,10000|FF_Colour Red,RGB_TT,1500|FF_Flasher
DV_FLCN,FL_ON,1,1,100,Red|FF_Flasher DV_FLOL,FL_ON,1,1,100,Red|FF_Flasher
DV_FLOR,FL_ON,1,1,100,Red|FF_B2S B2SStartAnimation,BurgerBall|FF_DOF E402,-1
Nudge|OFF|FF_Dev DV_SH,250|FF_Flasher DV_FLCN,FL_OFF,1,1,100,Red|FF_Flasher
DV_FLOL,FL_OFF,1,1,100,Red|FF_Flasher DV_FLOR,FL_OFF,1,1,100,Red

Left_Slingshot|ON|FF_Dev DV_LS,-1|FF_DOF E110,-1
Right_Slingshot|ON|FF_Dev DV_RS,-1|FF_DOF E111,-1
Bumper1|ON|FF_Dev DV_BL,-1|FF_DOF E120,-1
Bumper2|ON|FF_Dev DV_BC,-1|FF_DOF E121,-1
Bumper3|ON|FF_Dev DV_BR,-1|FF_DOF E122,-1
Bumper1_Flasher|ON|FF_Flasher DV_FLIL,FL_ON,1,1,100,Random|FF_B2S B2SStartAnimation,LouiseFace
Bumper1_Flasher|OFF|FF_Flasher DV_FLIL,FL_OFF,1,1,100,Random
Bumper2_Flasher|ON|FF_Flasher DV_FLCN,FL_ON,1,1,100,Random|FF_B2S B2SStartAnimation,GeneFace
Bumper2_Flasher|OFF|FF_Flasher DV_FLCN,FL_OFF,1,1,100,Random
Bumper3_Flasher|ON|FF_Flasher DV_FLIR,FL_ON,1,1,100,Random|FF_B2S B2SStartAnimation,TinaFace
Bumper3_Flasher|OFF|FF_Flasher DV_FLIR,FL_OFF,1,1,100,Random
Left_SlingShot_Flasher|ON|FF_Colour Yellow,RGB_TT,125|FF_B2S B2SStartAnimation,BobFace
Right_SlingShot_Flasher|ON|FF_Colour Yellow,RGB_TT,125|FF_B2S B2SStartAnimation,LindaFace
Crane_Flasher|ON|FF_Flasher DV_FLOR,FL_ON,1,1,100,Yellow|FF_B2S B2SStartAnimation,Windows|FF_DOF
E413,-1
Crane_Flasher|OFF|FF_Flasher DV_FLOR,FL_OFF,1,1,100,Yellow
FerrisWheel_Flasher1|ON|FF_Flasher DV_FLOL,FL_ON,1,1,100,Silver|FF_B2S
B2SStartAnimation,Windows|FF_DOF E410,-1
FerrisWheel_Flasher1|OFF|FF_Flasher DV_FLOL,FL_OFF,1,1,100,Silver
FerrisWheel_Flasher2|ON|FF_Flasher DV_FLOL,FL_ON,1,1,100,Silver|FF_B2S
B2SStartAnimationReverse,Windows|FF_DOF E412,-1
FerrisWheel_Flasher2|OFF|FF_Flasher DV_FLOL,FL_OFF,1,1,100,Silver
Lamp1_Flasher|ON|FF_Dev DV_SR,2000|FF_B2S B2SSetData 51,1|FF_B2S B2SSetData 58,1|FF_DOF E140,-1
Lamp2_Flasher|ON|FF_Dev DV_SR,2000|FF_B2S B2SSetData 52,1|FF_B2S B2SSetData 57,1|FF_DOF E141,-1
Lamp3_Flasher|ON|FF_Dev DV_SR,2000|FF_B2S B2SSetData 53,1|FF_B2S B2SSetData 56,1|FF_DOF E301,-1
Lamp4_Flasher|ON|FF_Dev DV_SR,2000|FF_B2S B2SSetData 54,1|FF_B2S B2SSetData 55,1|FF_DOF E308,-1
SinkHole_Flasher|ON|FF_Flasher DV_FLOL,FL_ON,1,1,100,Lime|FF_B2S B2SStopAnimation
Windows|FF_DOF E407,-1
SinkHole_Flasher|OFF|FF_Flasher DV_FLOL,FL_OFF,1,1,100,Lime|FF_B2S B2SSetData 51,0|FF_B2S
B2SSetData 52,0|FF_B2S B2SSetData 53,0|FF_B2S B2SSetData 54,0|FF_B2S B2SSetData 55,0|FF_B2S
B2SSetData 56,0|FF_B2S B2SSetData 57,0|FF_B2S B2SSetData 58,0
WonderWharf_Left_Flasher|ON|FF_Flasher DV_FLOL,FL_ON,1,1,100,Yellow|FF_B2S B2SSetData 1,1|FF_DOF
E411,-1
WonderWharf_Left_Flasher|OFF|FF_Flasher DV_FLOL,FL_OFF,1,1,100,Yellow
WonderWharf_Right_Flasher|ON|FF_Flasher DV_FLOR,FL_ON,1,1,100,Yelow|FF_B2S B2SSetData
2,1|FF_DOF E414,-1
WonderWharf_Right_Flasher|OFF|FF_Flasher DV_FLOR,FL_OFF,1,1,100,Yelow
```

There are three sections, but only two types of section. They are TRIGGERS and COMMANDS. TRIGGERS are the detected events that initiate the COMMANDS. All of the triggers need to be discovered by research and memory mapping of FX2 tables. Never change a provided trigger unless you have really good reason to, it will likely screw things up considerably.

The two different TRIGGER sections are CORE and TABLE BASED. ***DO NOT*** rename the CORE triggers. The Table based triggers can have any name you like, so long as it matches the name used for its commands in the COMMAND section.

The COMMANDS are the instructions to carry out when the trigger event occurs. Every trigger event can have an ON and OFF command, ie LeftFlipper ON action would be to turn your left flipper solenoid on, the LeftFlipper OFF command would be to turn it off. Not all triggers need both ON and

OFF commands. If a trigger does not need any commands its command line can be left with no commands or simply omitted altogether.

The commands are a subset of those available for Future Pinball and documented in the previous section. The available commands are

- FF_Dev
- FF_Colour
- FF_Button
- FF_Flasher
- FF_ROM
- FF_DOF

Please note, all times used are in milliseconds. For Future Pinball they are in hundredths of a second. So for a FX2 command, 1000 = 1 second.

You can have as many commands on a line as you like separated with a pipe (|) character.

There is one additional command for Pinball FX2 for use with B2S back glasses, FF_B2S

## FF_B2S CCC,PPPP – Send a command with parameter(s) to the B2S back glass

The command is subset of the standard B2S commands. The valid commands are;

- B2SSetData
- B2SStartAnimation
- B2SStartAnimationReverse
- B2SStopAnimation
- B2SStapAllAnimations

For example

FF_B2S B2SStartAnimation,MyAnimation – start the animation you called MYANIMATION in the B2S

FF_B2S B2SStopAllAnimation – cease all animations

FF_B2S B2SSetData 15,1 – turn on Lamp 15

FF_B2S B2SSetData 4,0 – turn off Lamp 4

Details of these commands and their parameters can be found in the B2S documentation.

One special note, when using DOFLinx to communicate with a B2S back glass all animation names must be IN CAPITAL LETTERS ONLY when you make / edit your back glass (directb2s) file.

B2S back glass files are placed in the path as set by PATH_FX2_B2S and should have the same name as the FX2 file for that same game.

For B2S back glasses to be visible you will need to turn off the relocation of the back glass within Pinball FX2 cabinet settings (Back Glass Repositioning = Off). This does turn it off for all games, so we’d better get directB2S files for every game happening soon!

FX2 trigger configuration files have been setup with a default set of commands, you don’t need to edit them if you don’t want to.

## Pinball FX3 Memory Triggers – Full Force Feedback

Similar to FX2 …… but just not written yet.  In fact, unlikely to ever get written.  I’m sure if you have the need to know that we’ll be in touch.

## Pinball FX Events and Triggers – Full Force Feedback

Not written yet.

## MAME Commands, Triggers and Actions – Full Force Feedback

You will need to use a version of MAME with the functionality added to talk to DOFLinx. Install a version of MAME that matches the MAME64.exe included with DOFLinx then replace MAME64.exe. In addition, you will need to make sure the following is setup in your MAME.INI.

```
# OSD OUTPUT OPTIONS
#
output              network
```

MAME files contain commands, triggers and actions that are run when MAME games start and stop. In the same way as FX3 there are four XX.MAME files;

1) All_Pre.MAME – this is executed first for every MAME game when it starts
2) Default.MAME – this runs for any game that does not have a game specific file
3) XXXX.MAME – where XXXX is the specific MAME game that is running
4) All_Post.MAME – this is executed last for every MAME game when it starts

Only three will ever be run. Only default or the game specific file (2 or 3) can run, not both.

The first three of these has the sections;

[STARTUP]
[SHUTDOWN]
[COMMANDS]
[CLEAR COMMANDS]

The last has;

[COMMANDS]
[CLEAR COMMANDS]

The actual game MAME file also has a [SCORE] section.

The [STARTUP] and [SHUTDOWN] sections contain pretty much any parameters that can be contained in either the DOFLinx.INI or supplementary INI files. For example, BUTTONS_ON= or FF_ROM= .

Both the [COMMANDS] and [CLEAR COMMANDS] sections contain the triggers and actions associated with the trigger from the game. For example, a trigger of “led0” can have both on and off actions such as;

```
led0|ON|FF_Button BUT_P1,BA_ON,0,0
led0|OFF|FF_Button BUT_P1,BA_OFF,0,0
```

The above will turn on the Player 1 button when led0 is “1” and turn it off when led0 is “0”. In many MAME games this effectively flashes the Player 1 button when credits are inserted.

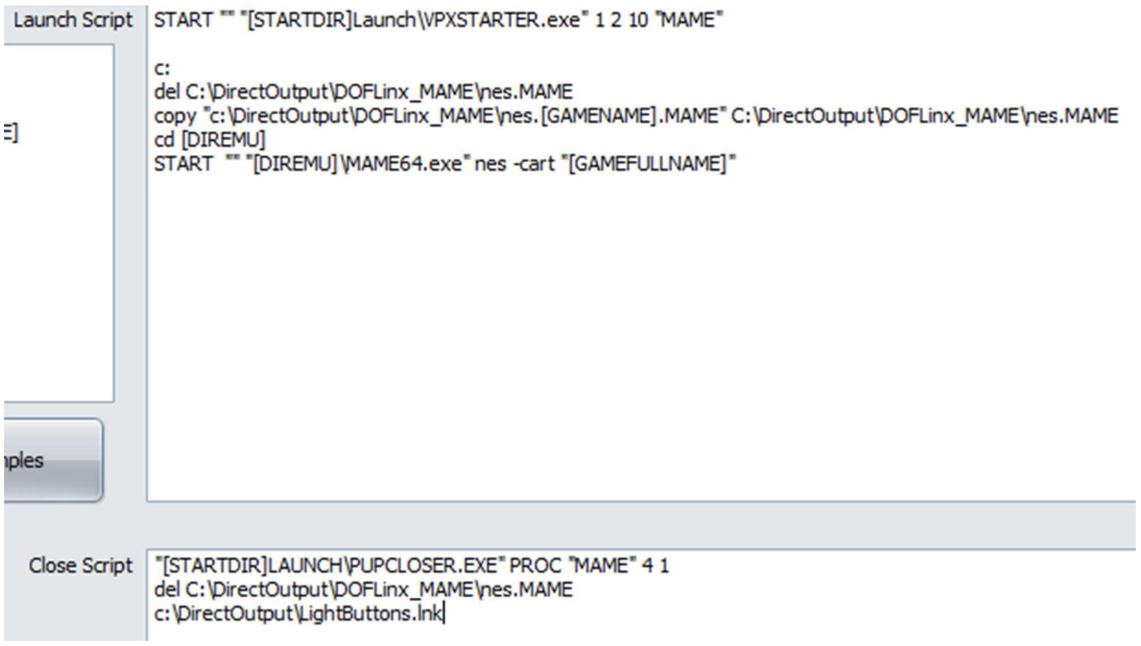
The [CLEAR COMMANDS] section exists to remove any commands that you do not want processed. It is most often used in the All_Post.MAME file to remove commands that are not relevant to your setup and may appear in a number of other MAME files. It saves editing all of those game files individually.

The [SCORE] section allows details for the polling of the score in the MAME game. Monitoring of score changes and reaching particular scores can be setup as triggers for any sort of actions.

One of the machines included with MAME is the Nintendo Entertainment System (NES). Running a NES game in MAME is achieved using a command line like

MAME64 nes -cart Tetris.zip

When MAME starts it reports “nes” as the game not the individual cartridge. As such, to get frce feedback via the correct MAME file for DOFLinx you will need to have your front end (or some other method) copy in the correct nes.XXXXXXXXX.MAME file to the nes.MAME file. Below is my setup from PinUp Popper.



```
START "" "[STARTDIR]Launch\VPXSTARTER.exe" 1 2 10 "MAME"

c:
del C:\DirectOutput\DOFLinx_MAME\nes.MAME
copy "c:\DirectOutput\DOFLinx_MAME\nes.[GAMENAME].MAME" C:\DirectOutput\DOFLinx_MAME\nes.MAME
cd [DIREMU]
START  "" "[DIREMU]\MAME64.exe" nes -cart "[GAMEFULLNAME]"
```

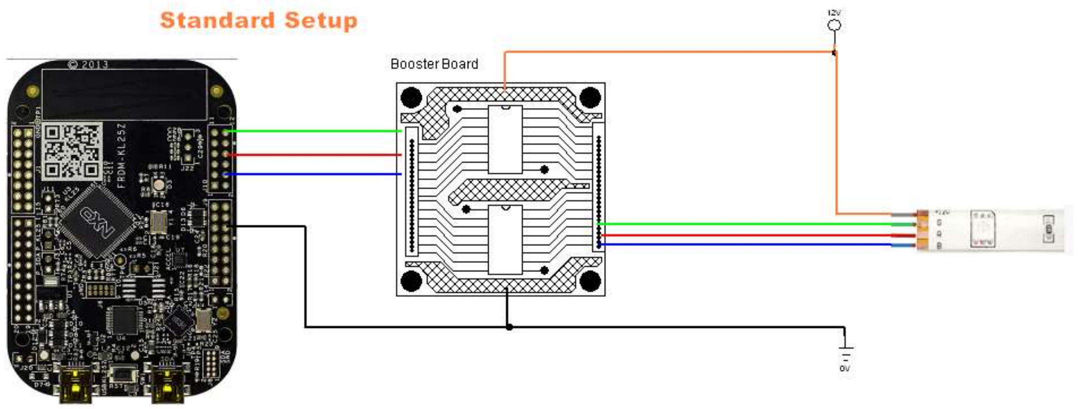
```
"[STARTDIR]LAUNCH\PUPCLOSER.EXE" PROC "MAME" 4 1
del C:\DirectOutput\DOFLinx_MAME\nes.MAME
c:\DirectOutput\LightButtons.lnk
```

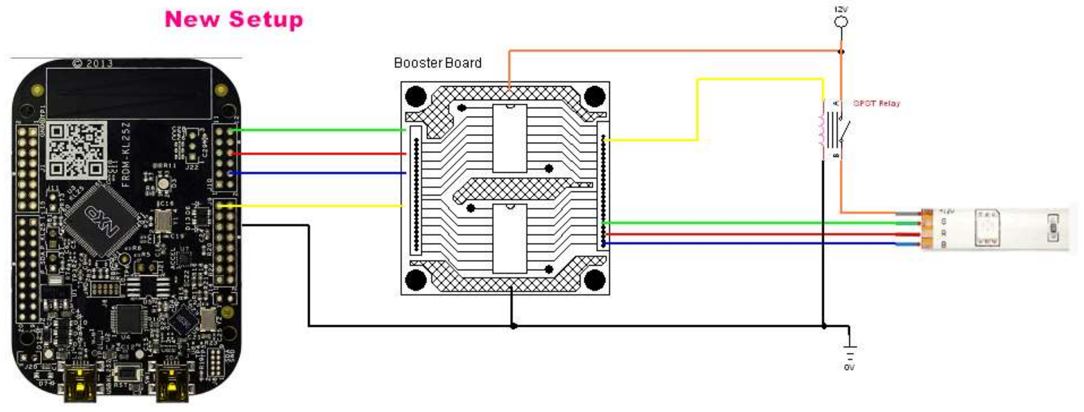
## Settings Pop Up Menu

The setting pop up menu is designed to allow you to control items within your cabinet. It is not about configuring DOFLinx. There are two types of settings, items that turn device ports on / off and hence control cabinet items electrically, plus one special setting for NIGHT mode.

To use settings that control electrical items, you must have those electrical items wired up appropriately. In my case I wanted to be able to turn on / off some of my lighting effects easily so that different people in my house like / don't like. I started off by having a physical switch inside the cabinet to switch the positive rail of the under cabinet lighting, I got sick of that. So then I installed a relay instead of that switch, the control for that relay is wired to another LEDWiz output. This means by switching the output to the relay I can electrically switch on / off my under cabinet lighting. The reason I didn't want to do this via configuration and disable it is because there are multiple configurations and they have a habit of going wrong at some stage, this is a quick and simple way.

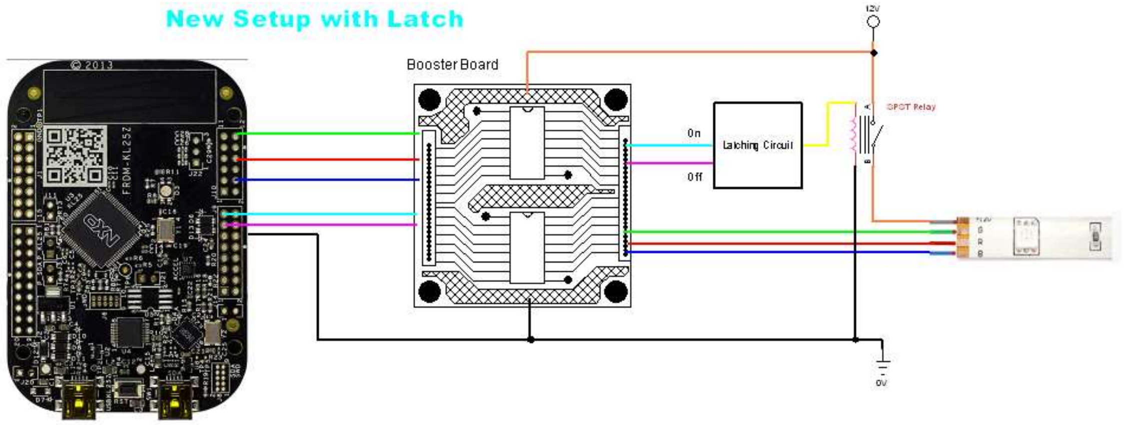
Below is the standard connection for a toy from your device.



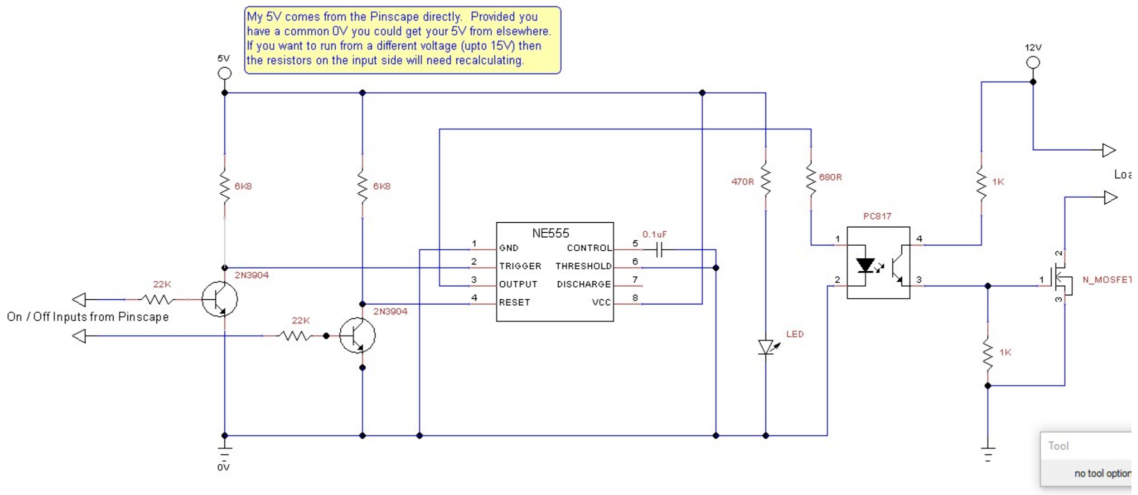


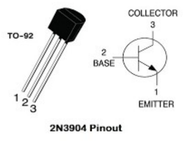
Above is the modified connection to allow control via the Settings option. It differs by the addition of a relay on the switch power line to the toy. The relay is controlled via another output device output port.

Each time an application opens the DOF framework it resets the outputs on each device. DOFLinx opens and closes DOF as required when tables / emulators are started and stopped. To avoid the relay resetting for your output it is possible (and desirable) to add a latching circuit. That means one signal to turn on a relay / MOSFET and another to turn it off. Doing this ensures that the device you are controlling remains on / off until you specifically change that setting. A logical diagram of this is below.

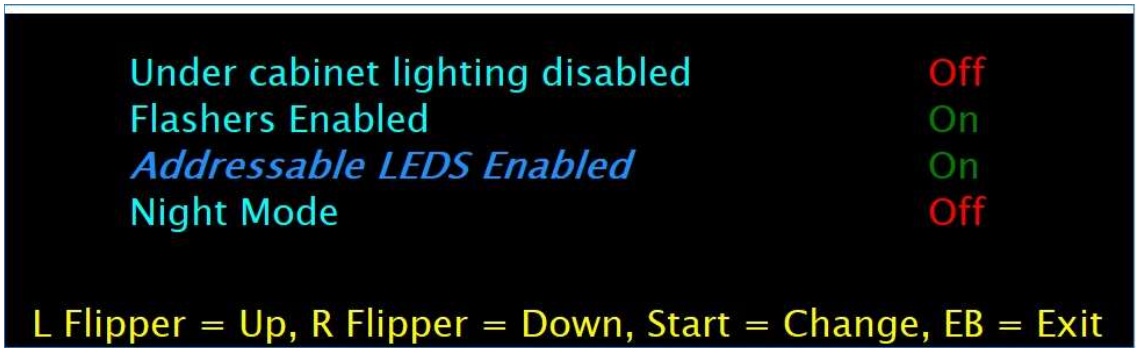


A cheap and easy latching circuit that I have used with a Pinscape is shown below.


My 5V comes from the Pinscape directly. Provided you have a common 0V you could get your 5V from elsewhere. If you want to run from a different voltage (upto 15V) then the resistors on the input side will need recalculating.



The Settings pop up menu is a bit reminiscent of a traditional pinball configuration on the DMD. As such I've made it so you can drag the pop up menu to your DMD (or what ever screen you like) and size it appropriately. When you exit the location and size will be saved in the registry for next time. (HKEY_LOCAL_USER\Software\DOFLinx\XX. If anything goes wrong you can safely delete the entire DOFLinx tree from the registry, it will recreate next time.


| | |
|---|---|
| Under cabinet lighting disabled | Off |
| Flashers Enabled | On |
| *Addressable LEDS Enabled* | On |
| Night Mode | Off |

L Flipper = Up, R Flipper = Down, Start = Change, EB = Exit

There is one “special”, and hence reserved, ID code that can be used, that is “NIGHT”, so “SETTING_NIGHT= “. This special setting allows the internal DOFLinx flag for NIGHT_MODE to be toggled.

An example of the configuration that would produce the settings pop up above and use Left Shift + Right Shift + Left Control + Right Control (all flipper and magna save buttons in my case) at the same time, along with Left Shift for UP, Right Shift for DOWN, Escape for EXIT and 0 for CHANGE is:

```
SETTING_ACTIVATE=A0,A1,A2,A3
SETTING_UP=A0
SETTING_DOWN=A1
SETTING_CHANGE=31
SETTING_EXIT=1B
SETTING_MESSAGE=L Flipper = Up, R Flipper = Down, Start = Change, EB = Exit
SETTING=UnderCab,101,Under cabinet lighting disabled,Off,Y
SETTING=Flashers,102,Flashers Enabled,On,Y
SETTING=Addressable,103,Addressable LEDS Enabled,On,N
SETTING=NIGHT,,Night Mode,Off,Y
```