

ZX Touch PC Tool

Designing ZTG packages with FX on a PC

Version 0.5

Companion to the ZX Touch User Manual and the ZX Touch FX Guide

1. Before you start

This guide assumes two things: that you have read the ZX Touch FX Guide, and that you have already built ZTG files with FX on the console itself. It does not re-explain what a marker is, what a shader slot does, or how the FX Manager decides to change a background. Those belong to the FX Guide and nothing about them changes here.

What this guide covers is the PC tool: where each of those things lives on screen, what the tool does differently from the console, and what it can do that the console cannot.

What the tool is for

The ZX Touch PC Tool is a Windows program for two related jobs. The first is designing and editing .ztg packages: it has a ZX Spectrum emulator built into it so that you can set up FX and see the result at once, instead of copying files to an SD card and trying again. The second is arranging finished packages into the dashboards the console shows when it starts, and exporting them ready to copy onto the card. Chapter 12 covers that.

It is a design tool. The emulator exists to show you what your package will look like, and everything in the program is arranged around getting a package right rather than around playing.

How faithful the emulator is

The emulator in this tool is not the console's emulator. It is a separate implementation of the same machine, written for the PC, and it behaves closely enough for preparing packages - colours, the screen, the FX pipeline and the way markers are recognised all match - but it is not identical in every detail.

The most visible difference is timing. On the console the emulator is synchronised with the display refresh; here it is not. Scrolling is therefore less smooth and fine motion is less exact than on the real device. There are also small rough edges here and there.

Please do not judge the console by this

If you do not own a ZX Touch, do not take anything you see here as a statement about the console's own emulator. That one is a different, more polished piece of work running under conditions this program cannot reproduce. Anything that looks imperfect here is a limitation of this design tool, not of the device.

None of this gets in the way of the job. Judging a background, a palette, a shader or a marker does not depend on scrolling being perfectly smooth. Playing does - which is why the tool is not meant for that.

Installing

There is nothing to install. The tool is a single executable that runs from wherever you put it, and it writes nothing to your system unless you ask it to.

- Keep ZXT_PC_TOOL.exe in a folder of its own.
- Put the ROMs folder next to it, holding 48.rom, 128-0.rom, 128-1.rom and trdos.rom. Having them there is not enough on its own: the tool starts with its built-in SE ROM, and uses the folder only once you set Settings > Misc > ROM to "Custom ROMs from ROMS". Without trdos.rom, TRD and SCL disks will not boot.
- The tool creates a SETTINGS folder next to itself for its own saved settings blocks.

Antivirus

A freshly built, unsigned executable that Windows has never seen before is sometimes flagged by antivirus heuristics. Nothing in the tool does anything unusual, but if Defender removes it, restore it from Protection History and add its folder to your exclusions.

Opening files by double-clicking

By default nothing is associated with the tool. If you want .ztg, .xztproj and .xzt dash files to open in it when you double-click them, use Tools > Associate .ztg, .xztproj and .xzt dash with this tool. It affects only your own Windows account, needs no administrator rights, and installs nothing. It is never done automatically.

The association points at the exact executable you ran it from. If you move the exe later, run that command again.

2. Projects and packages

This is the one idea in the tool that has no equivalent on the console, and it is worth getting straight before anything else.

A package is the output

A .ztg is what the console reads: one file holding the game, the settings, the artwork, the poke file and the FX data, all baked together. It is what you ship.

A project is the workshop

A .xztproj is a small text file that the console never sees. It does not contain artwork - it remembers where your artwork is, which game file to use, what you have named each background, how large each image is allowed to be, your own notes, and everything you have configured. You work in a project and export a package from it.

The advantage is that your source images stay where they are, at whatever resolution and quality you made them. Change a background on disk and the next export picks it up. A .ztg, by contrast, is finished goods.



The main window with a project open: tree on the left, emulator on the right.

Running a project

A project is not exported in order to be tried. There is a separate **Start Game** button on the toolbar, and it runs the project directly - no .ztg is created, nothing is written to disk.

What you get is the same thing the exported package would give you. The game, the settings, the FX data and the images the tool has prepared are all put into the emulator exactly as the export would write them, so running the project and running the .ztg you export from it amount to the same thing.

Press it again at any time to restart the game from the beginning. That does not disturb the settings or FX you have been editing - only the game restarts, and the At Start action runs again as it would on a real start.

Several projects at once

You can keep several projects open. They all appear in the tree, but only one drives the emulator at a time - that one is marked (active). Double-click another project's root to switch to it. Only the active project shows its full contents, because the On/Off states shown in the tree describe what is actually loaded.

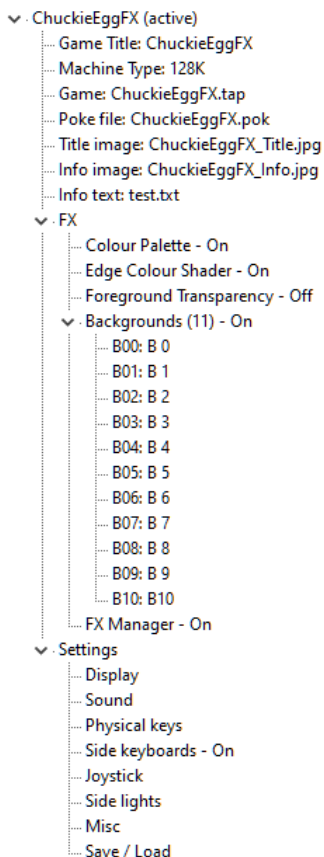
Closing the tool

Anything unsaved is offered for saving when you exit, project by project. If you cancel that save dialog, the tool stays open rather than discarding your work.

3. The main window

The tree

The tree is the table of contents for the active project. Every entry is clickable and opens the thing it names. Entries that are a switch show their state, so you can see at a glance whether FX, backgrounds or the side keyboards are on.



The project tree with a project expanded.

A newly opened project shows one level - the top-level entries are visible, FX and Settings stay closed. Whatever you open by hand stays open as you work.

The toolbar

The toolbar has three groups: transport (Start Game, play/pause, reset), FX (the five FX windows plus the FX master switch), and tools (the eight settings windows, a Tape player button that is not built yet and says so when pressed, snapshot and the POKE tool).

The tools group ends with the two export buttons - one writes the package to a file on this PC, the other sends it straight to the console. Both need a project open and are greyed out otherwise. Chapter 10 covers them.



The toolbar.

The emulator view and why 1:1 matters

The emulator draws the console as it really looks, at the console's own 1024x600: the screen in the middle, the side keyboards in the margins when they are enabled, and the six round buttons down each side. The round buttons work - Reset, mute, play/pause and the file browser do what they say.

The **1:1** button sizes the window so that one console pixel is drawn as exactly one screen pixel. Use it whenever you are judging how something looks.

At any other size the picture is being stretched by a fraction, and Windows has to invent pixels that are not there. The Spectrum's own character set suffers most: text that is crisp on the console turns slightly blurred or uneven, thin lines thicken or disappear, and a marker you are about to capture no longer looks like the thing the console will be matching. The overall impression of the design becomes untrustworthy - you end up correcting for a fault that only exists in the window.

The 1:1 button turns reddish whenever the window is not at an exact scale, so you can see at a glance that what you are looking at is approximate.

The status bar

Along the bottom: the machine type, the FX state, and a small floppy disk that appears only while a TR-DOS disk is in use. It lights up while the disk is being read, exactly as the indicator does on the console.

4. Controlling the machine

The PC keyboard

The PC keyboard drives the Spectrum keyboard directly, which is enough for loading, typing and most testing.

The gamepad

For anything to do with key mapping, a gamepad is not optional - it is the only way to exercise what you have configured.

Plug in an XInput gamepad and it stands in for the console's own controls. Its two D-pads and its buttons act as the console's eight physical keys, in the same arrangement, and its analogue stick acts as the console's joystick. That is what lets you check that your physical key mapping, your side keyboards and your joystick settings actually do what you intended - without one, those three settings windows can be filled in but not tried.

The Fn key is not available

The console's Function key and its combinations do not work here. Anything you would normally do by holding Fn - adjusting the volume, stepping through palette presets, taking a screenshot, cycling backgrounds, toggling FX - is done in the application instead: the toolbar buttons, the settings windows and the FX windows.

This is worth remembering when you come to the FX master switch. On the console you toggle all FX by holding the Function key together with its nearest adjacent key and pressing Left on the opposite D-pad. Here you press the FX button on the toolbar.

5. Loading games

File > Open Game or ZTG accepts .ztg, .tap, .z80, .sna, .trd and .scl.

Opening a plain game file is not the same as working in a project. It resets the FX settings to their defaults first, so a game you open casually always looks the same regardless of what was loaded before it. A project keeps its own FX state.

TRD and SCL disks

These boot through the Beta Disk interface with the TR-DOS ROM, the same way the console does, and need trdos.rom in the ROMs folder. The disk stays in the drive across a machine reset, and is ejected when you load something else.

6. Settings

The eight settings windows hold exactly the parameters that travel inside a .ztg, and nothing else. Device-wide settings that the console overwrites from its own configuration when a package loads - screen brightness, volume, Wi-Fi, the dashboard - are deliberately absent, because setting them here would achieve nothing.

Everything you change is live. There is no apply step and no need to restart the game.

- Display - screen size, turbo, and the display options
- Sound - AY engine and mixer levels
- Physical keys, Side keyboards, Joystick - the three key-mapping screens, laid out like the console's own. These are the ones a gamepad lets you test.
- Side lights - mode, colours, and behaviour during tape loading
- Misc, Save / Load - the remainder, and the settings blocks

Turbo

Double emulation speed works here as it does on the console. The FX Guide suggests it for speed-running to a level you need a marker from; that works in the tool too.

7. FX

The five FX windows correspond one to one with the console's FX screens, and the concepts behind them are the FX Guide's subject. What follows is only where they are and what differs.

The FX master switch

The FX button on the toolbar switches the whole FX pipeline off and on. Turning it off leaves all your FX data intact and simply stops it being rendered, which is the quickest way to see what your work is actually contributing.

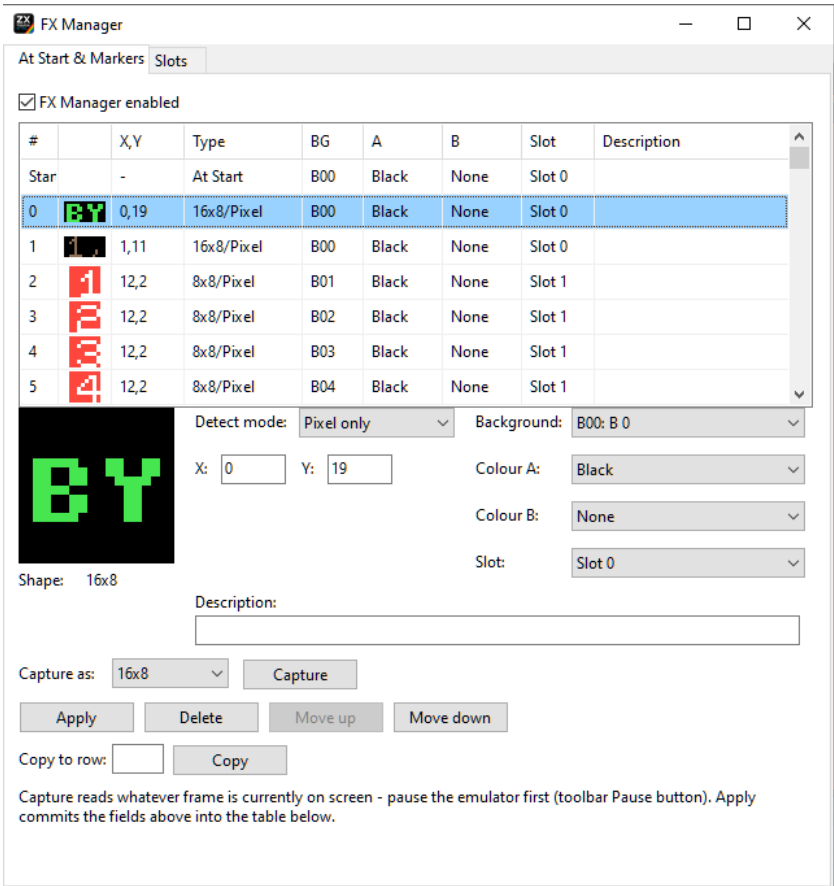
Loading a game or a project that carries FX data turns the switch back on, so FX never stays invisible because of a state you left behind.

Colour palette, Shader, Foreground transparency

Three windows, each editing live. Because the emulator keeps running while you work, you can leave a window open and watch the picture change as you drag. The Shader window carries the console's three quick tools next to Edit - Swap L1 / L2, L1 <- Base and L2 <- L1 - which are the fastest way to fix a shader style whose colours are right but on the wrong levels.

FX Manager

Where the console shows one marker at a time, the tool shows all fifty in a single table, plus the At Start row. Everything is visible at once and you can compare rows instead of paging through them.



The FX Manager table: At Start plus fifty marker slots, with the edit strip below.

Click a row and its values appear in the strip beneath the table; change what you need and press Apply. The marker's captured image is shown enlarged beside the table so you can see what you actually captured.

Descriptions

Every marker, every shader slot and the At Start row can carry a description of your own. This is the single most useful thing in the window once a package grows: instead of a table of anonymous rows you get "level 2 title", "boss appears", "slot 3 - cave, dark blue".

Descriptions live in the project only. They are never written into the .ztg, so they cost nothing in the package and the console never sees them. They are notes to yourself, and to whoever picks up the project after you.

Reordering markers

Markers can be moved up and down the table, which the console cannot do at all. This is not cosmetic. The FX Manager resolves markers by priority, and priority follows position in the table - so moving a marker changes which one wins when two could match. Being able to reorder means you can tune that without recapturing anything.

A marker can also be copied to another row, which is what the level-progression trick in the FX Guide needs: several actions in sequence sharing one marker.

Capturing a marker

Capture reads the frame that is currently on screen, so the game has to be standing still. The capture cursor only appears when all three of these are true:

- the emulator is paused
- the FX Manager window is open
- a marker row is selected - not the At Start row

With those in place, a thin rectangle appears on the emulator screen. The sequence is:

1. Run the game to the point where the marker you want is on screen.
2. Pause the emulator with the toolbar's Pause button.
3. Open FX Manager and select the marker row you want to fill.
4. Click on the emulator screen to move the rectangle onto the marker. It snaps to the character grid. You can also type the coordinates directly.
5. Press Capture, then Apply.

This is also where 1:1 earns its keep: at any other scale you are positioning the rectangle over a picture that is not quite the picture the console will be matching.

8. Backgrounds

Backgrounds are where the PC tool differs most from the console, because it can prepare the images for you.

Resolutions

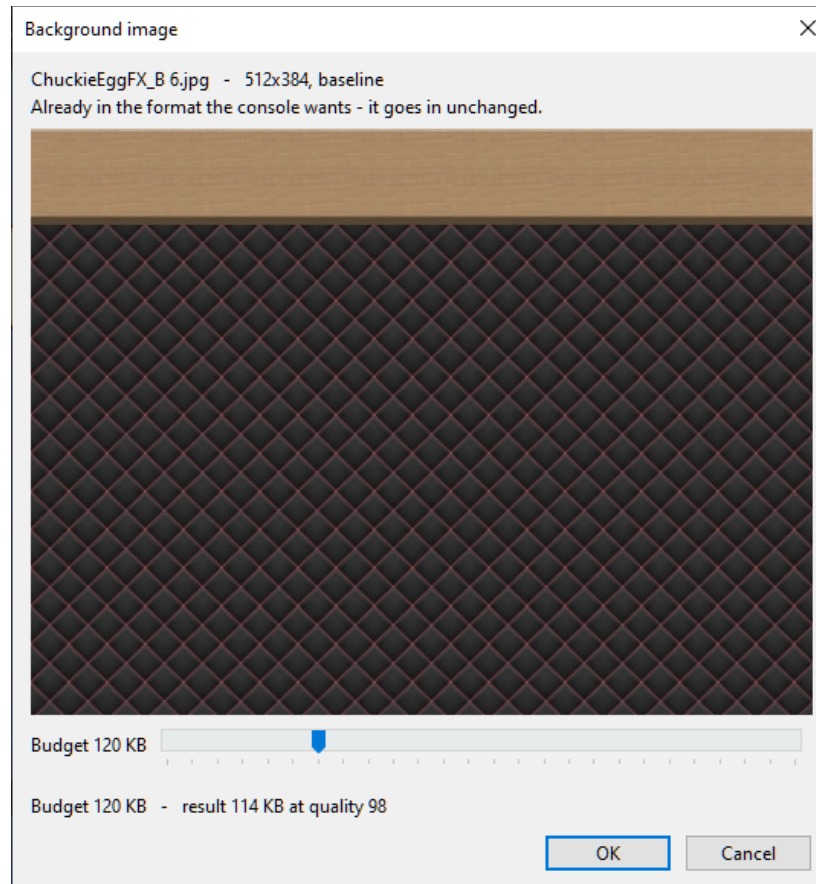
The console accepts many resolutions and scales them, but multiples of the Spectrum's own 256x192 avoid scaling artefacts and render faster. The tool therefore settles every background on one of three sizes:

- 256x192 - passed through untouched
- 512x384 - anything smaller than 768x576
- 768x576 - anything that reaches 768 wide or 576 tall

You do not have to prepare anything. Add a photograph straight from a camera and the tool will scale it, convert it to baseline JPEG, and tell you it is doing so.

The size budget

Rather than a quality percentage, each background has a size budget in kilobytes. The tool then finds the highest quality that fits inside it. This is the useful way round: the same quality number produces wildly different sizes on different pictures, whereas a budget is a promise about the finished package.



The background dialog: the image at the resolution it will actually go in at, with the budget slider below.

The preview is not an approximation. The image really is scaled, really encoded at that budget and decoded back, so what you are looking at includes the compression artefacts the console will show.

Images that are already right

A file that is already the correct size, already baseline, and already inside its budget is copied into the package untouched. Re-encoding a JPEG that needs no work only loses quality, and can even make the file bigger.

What you see is what ships

The emulator is given exactly the bytes the export would write - not your original files. A background that will be scaled and re-compressed on the way into the package looks that way while you are testing, so there are no surprises after export.

Names

A background's name has nothing to do with its filename. It is yours to write, and it is worth writing: "cave, level 3" tells you where the image belongs, which B07_something.jpg never will.

Like the FX Manager descriptions, these names stay in the project and are not written into the .ztg.

Trying backgrounds while the game runs

On the console, manual background switching is locked out while the FX Manager is enabled. Here it is not. With the game running you can open the Backgrounds window and step through your images to see how each one sits behind the actual graphics.

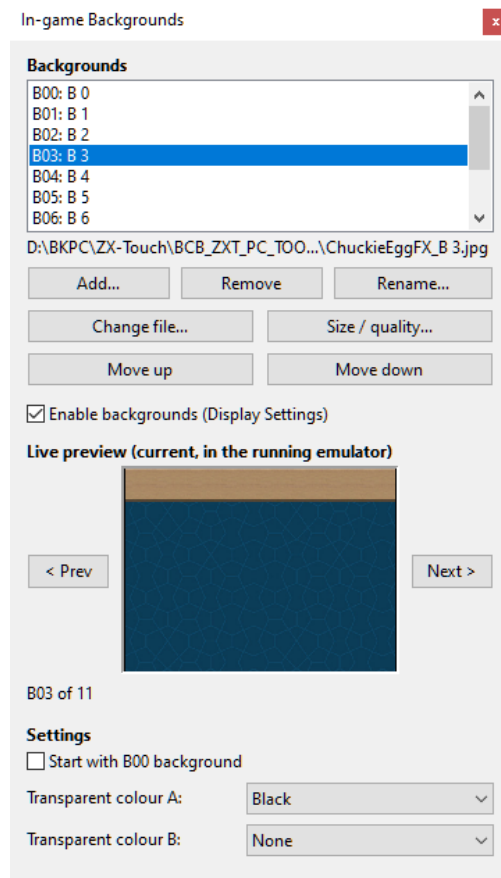
It is only a view

Changing the background this way changes nothing in the project and nothing in the FX Manager. It is a temporary look, no more. If the FX Manager recognises a marker while you are doing it, it will set the background its own entry calls for and your manual choice is replaced. Nothing you do here alters what the manager will do.

Managing the list

Change file replaces the image behind a background without moving it. B03 stays B03. This matters because the Bxx numbers are positions, so adding and deleting to swap one image renumbers everything after it.

Size / quality reopens the budget dialog for a background you have already added. Clicking a background in the project tree goes straight there.



The In-game Backgrounds window.

9. Artwork and extras

Title image

144x208, and the tool will scale whatever you give it. Progressive JPEGs are fine here - the title image is stored in the package as raw pixels rather than as a JPEG, so its encoding never matters. There is no quality setting for the same reason: the block is a fixed size whatever you put in.

Info image

796x600, stored in the package as a real JPEG, so it has a size budget exactly like a background. The default is 200 KB. Full quality at that resolution can reach 400-500 KB, which is more than the picture usually deserves.

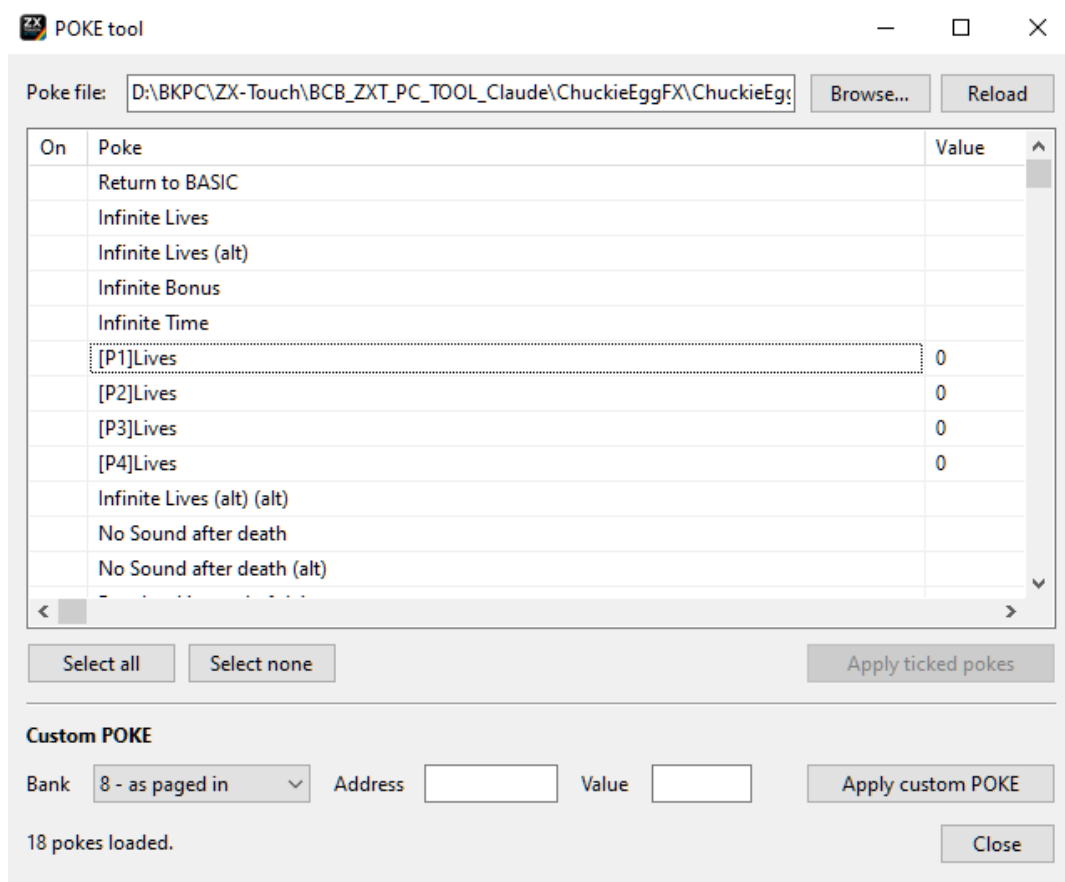
The dialog shows the image at its true size. Judging compression on a shrunken preview hides precisely the artefacts you are trying to judge.

Info text and POKE file

Both are attached as files. The tool carries them into the package unchanged; use the console to preview how the info text will be laid out.

POKE tool

The same two ways in as the console: a custom poke typed as bank, address and value, or a .pok trainer file whose entries you tick and apply.



The POKE tool, with a trainer file loaded.

The tool shows the whole list at once instead of paging through it. Pokes whose value the file leaves open get an editable Value cell, as they get an edit box on the console. Applying writes into the running machine and reports how many bytes actually took.

The poke file is picked up from the active project, or from inside a .ztg you opened directly. Browse loads any other .pok.

Snapshots and screenshots

One button, and the format you choose in the save dialog decides what happens. Pick .z80 or .sna and you save a snapshot; pick .jpg or .png and you save a picture.

- JPG captures the whole 1024x600 display, exactly what you see.
- PNG captures only the foreground, at 512x384, with the background pixels transparent - the same PNG the console produces, from the same encoder. That size is deliberate: it is one of the background resolutions, so a foreground cut out this way drops straight onto a background you are preparing.

The PNG is the one to use when preparing background artwork, because the foreground arrives already cut out and can be laid straight over a candidate background.

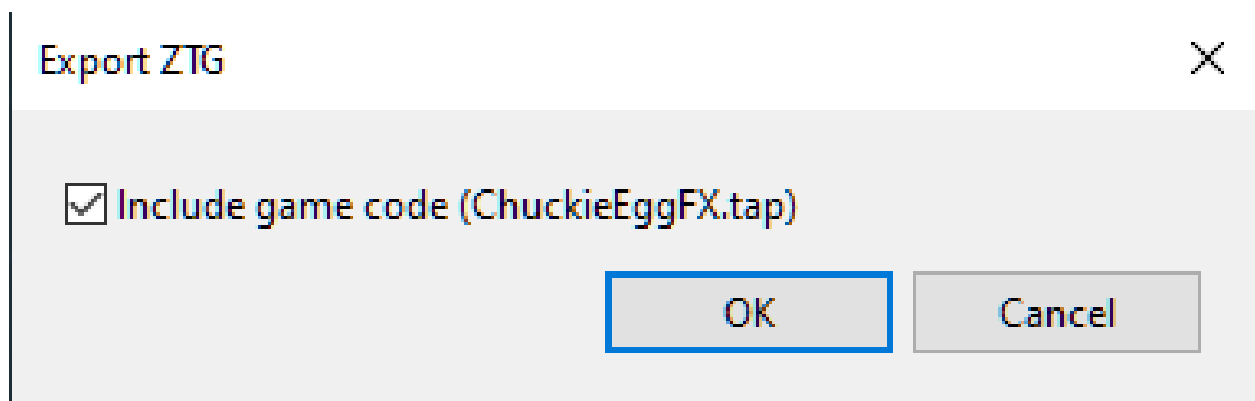
Snapshots are worth remembering for a different reason. Reaching a level you need a marker from is often the hardest part of the job, and the FX Guide's advice applies here: use the POKE tool, load a snapshot someone else made, or record your own along the way with double speed switched on.

The suggested filename is built from the project or the loaded game plus a number, and the dialog opens in that file's folder.

10. Producing the package

Export

File > ZTG > Export Project as ZTG writes the package. Images that need converting are converted at this point, from your originals - which is why the originals are worth keeping at full quality.



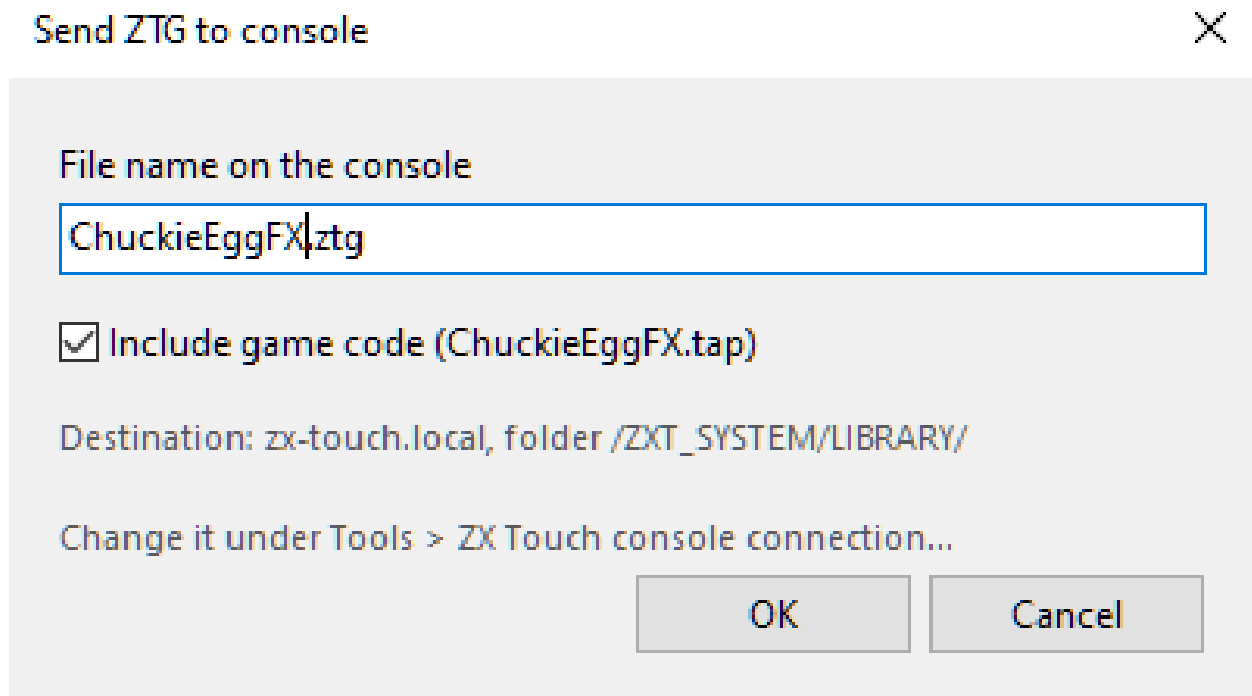
The export dialog.

The **Include game code** option decides whether the game itself goes in. Clear it and the package carries everything except the game, which is the sharing arrangement the User Manual describes: the recipient supplies their own copy of the game.

Sending it straight to the console

File > ZTG > Send Project to Console, or the last button on the toolbar, builds the package and sends it to the console over the network. Nothing is left behind on this PC: the package is built into a temporary file, sent, and deleted. When you want a copy on disk as well, use **Export Project as ZTG** instead - or as well.

The dialog asks for the name the file should have on the console, and whether the game code goes in - the same choice as a normal export. Underneath, it reminds you which console and which folder it is about to send to, because that is set once in another window and is easy to forget.



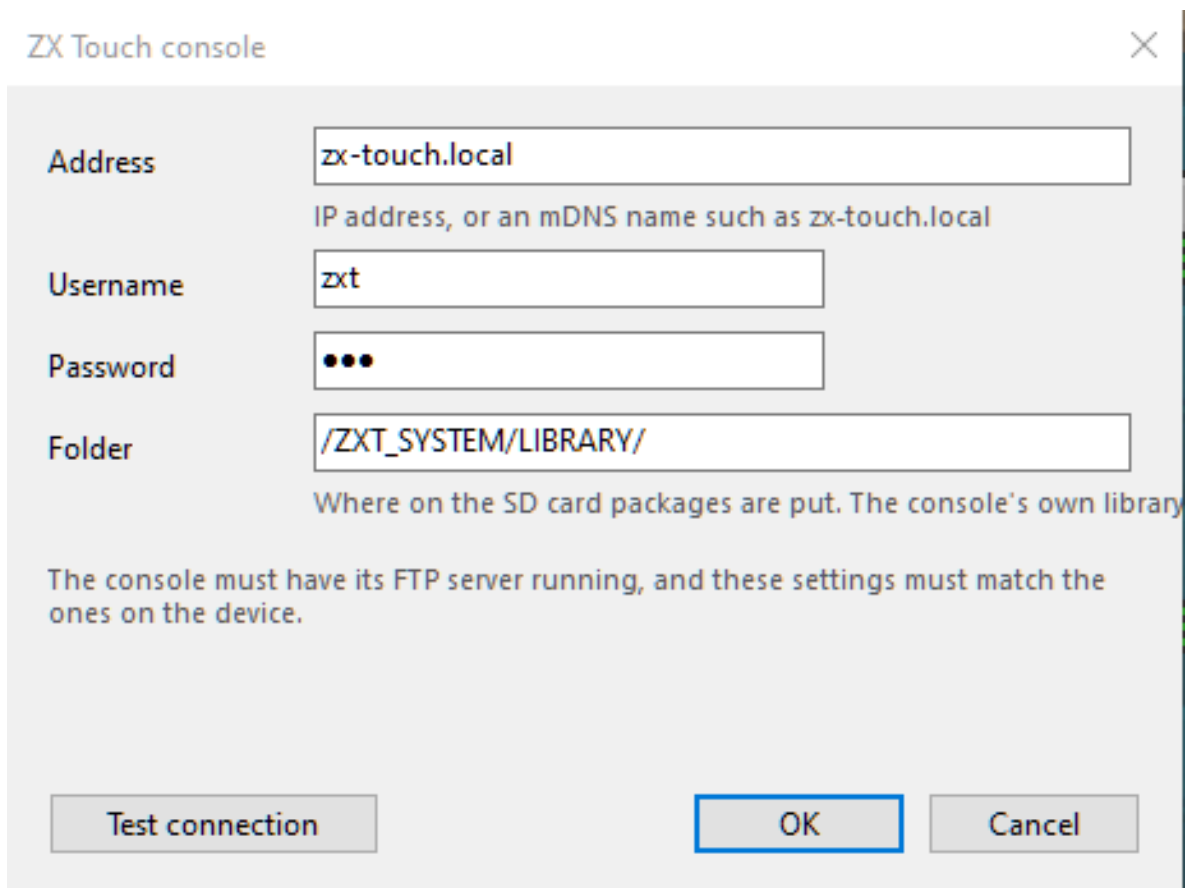
The send dialog, with the destination shown underneath.

If a file of that name is already in that folder, you are asked before anything is replaced. Say no and the transfer stops there - nothing on the console is changed.

Setting up the connection

The console details are entered once, under **Tools > ZX Touch console connection**.

- Address - the console's IP address, or its mDNS name (zx-touch.local by default). The mDNS name is worth using: the router can hand the console a different IP after a reconnection, the name does not change.
- Username and password - as set on the console. Both are zxt out of the box.
- Folder - where on the SD card the package should land. /ZXT_SYSTEM/LIBRARY/ is the console's own library, which is what you usually want.



The screenshot shows a dialog box titled "ZX Touch console" with a close button (X) in the top right corner. It contains four input fields: "Address" with the value "zx-touch.local", "Username" with the value "zxt", "Password" with three dots, and "Folder" with the value "/ZXT_SYSTEM/LIBRARY/". Below the "Folder" field is a descriptive text: "Where on the SD card packages are put. The console's own library". At the bottom, there is a note: "The console must have its FTP server running, and these settings must match the ones on the device." and three buttons: "Test connection", "OK", and "Cancel".

Address:
IP address, or an mDNS name such as zx-touch.local

Username:

Password:

Folder:
Where on the SD card packages are put. The console's own library

The console must have its FTP server running, and these settings must match the ones on the device.

Test connection OK Cancel

The connection settings, with the Test button.

Test connection logs in, checks that the folder is there and hangs up again, and says what happened. It is worth doing once after typing the details in, rather than discovering a typo in the middle of a transfer.

The settings are kept in an .ini file next to the program, so they travel with the folder if you move it. The password is stored in plain text there. It is a device password on your own network, but it is worth knowing.

The FTP server has to be running on the console

Network settings on the console offer the Web File Manager and the FTP Server, and only one of the two can be active at a time. Sending from here needs the FTP server. The console also accepts only one connection at a time, so close any other FTP client (FileZilla, Total Commander) before sending.

If it will not connect

The server supports only active (PORT) mode, which means the console opens a connection back to this PC rather than the other way round. Windows Firewall may ask about that the first time you send; allow it for private networks.

That also explains the most confusing failure: if the login clearly worked and it then sits there until it times out, it is almost always the firewall blocking the connection coming back. The tool says as much when it gives up.

Everything else is usually simpler - the wrong address, the FTP server not switched on, or the Web File Manager running instead of it.

Starting from an existing package

File > ZTG > Create Project from ZTG unpacks a package into a project - artwork, text, poke file and backgrounds are written out as files you can edit, and everything else is carried across. It is the way to pick up a package someone else made.

11. Working on a package without a project

Adding or removing the game code does not require a project at all. A finished .ztg can be handled directly, which is convenient when all you want to do is prepare a package for sharing, or complete one you have been sent.

Taking the game out

With a .ztg open, **File > ZTG > Save ZTG without Game As** writes a copy with the game removed and everything else - settings, FX, artwork, poke file - byte for byte intact. It always asks for a new name and never overwrites the package you have open.

Putting a game in

Open a package that has no game in it and the tool offers to complete it. Choose a game file and it is written into that same .ztg, which then opens ready to run.

Decline the offer and the package opens as it is. You get the console's own "The game code is missing!" message on screen, and the status bar says so for as long as the package is loaded.

For everything about FX itself - what to capture, when a shader helps, how to plan backgrounds across levels - the ZX Touch FX Guide remains the reference.

12. Dashboards and the games library

Everything up to here has been about one package at a time. This chapter is about the other half of the tool: taking packages you have already made and arranging them into the dashboards the console shows, then exporting the result ready to copy onto the SD card.

It is a second kind of project, with its own file. An FX project (.xxtproj) builds one package; a dashboard project (.xxtdash) arranges many. Both can be open at the same time, and the tree shows both - but only one of them drives the emulator, and that one is marked (active).

What a dashboard project holds

Only paths. The games stay wherever you keep them on disk, at whatever names they already have, and the project remembers where they are and what order they appear in. Nothing is copied until you export.

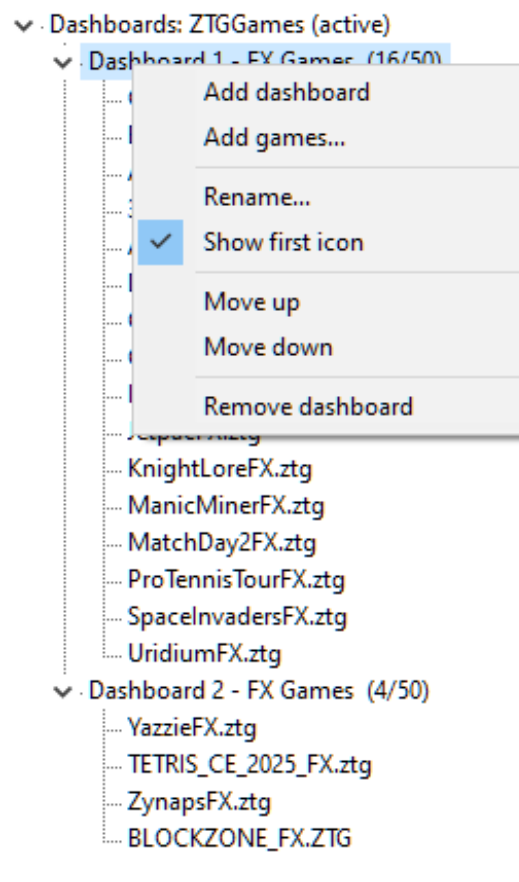
That means you can keep working on a package in an FX project and the dashboard picks up the new version at the next export. It also means moving or renaming a .ztg on disk breaks the reference - the tool marks such a

game (missing) in the tree and refuses to export until it is put right, rather than quietly leaving a hole in the dashboard.

Making one

File > Dashboards > New Dashboard Project asks for a name and then, straight away, where to save it - the same way a new FX project does. The name matters: it becomes the name of the export folder.

A new project starts with one empty dashboard. Right-click it in the tree for everything you can do to it.



The dashboard project in the tree, with the right-click menu open.

Adding and arranging games

- Add dashboard adds another one to the project. They are numbered in the order they appear in the tree, which is the order the console will show them in.
- Add games... opens a file dialog. You can pick several .ztg files at once, and they are added in the order the dialog lists them.
- Rename... names the dashboard. The name appears along the top of it on the console. It is limited to 49 characters, because that is the size of the field in the dashboard file.
- Show first icon turns the console's SD-card entry on or off for that dashboard. It is on by default and takes the first cell, which is why turning it off shifts everything up one place.
- Move up / Move down shift a game, or a whole dashboard, one place.

- Remove takes a game off the dashboard, or deletes the dashboard. Neither touches the .ztg files themselves.

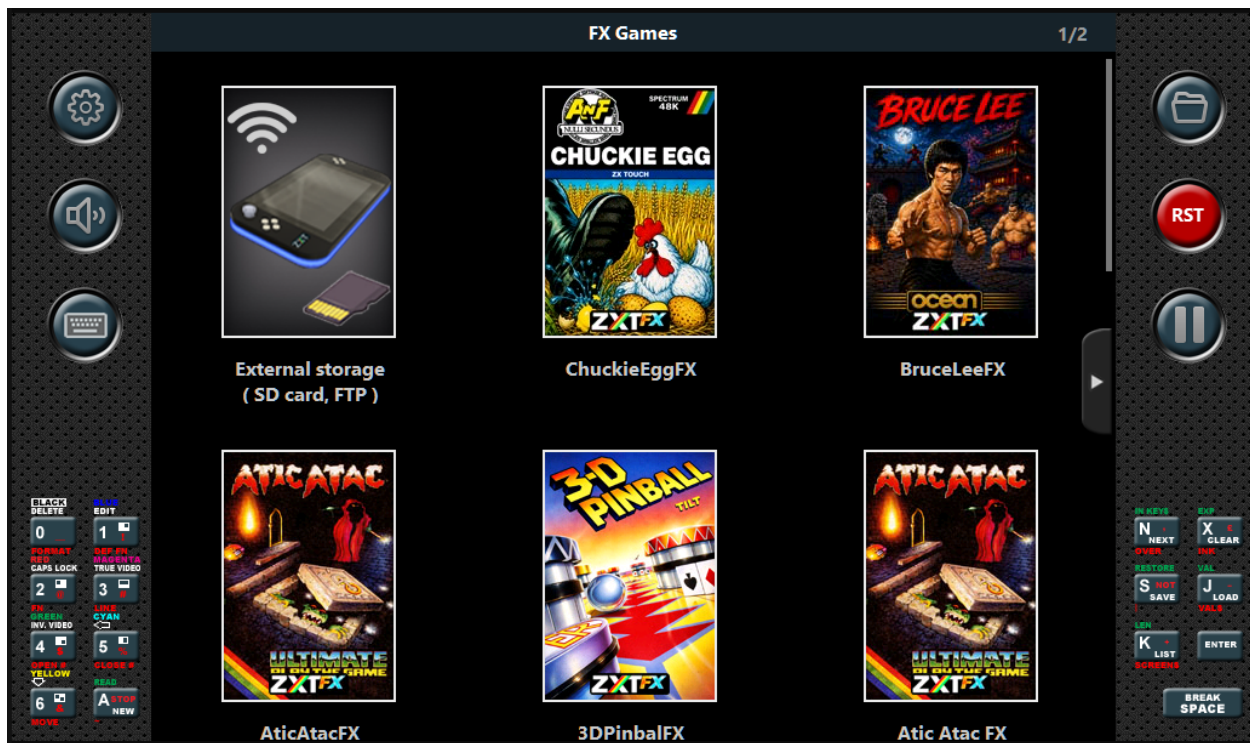
Games can also be rearranged by dragging - either in the tree, or by dragging the icons themselves on the emulator screen. Dragging onto another game puts it in that place; dragging onto a different dashboard in the tree moves it there.

Whole dashboards can be dragged in the tree the same way: drop one onto another and it takes that place, which is what Move up and Move down do a step at a time. Whichever dashboard you were looking at stays on screen wherever it ends up in the order.

A dashboard holds at most **50 games**, which is the console's own limit. The tree shows the count next to each one.

Seeing it as the console will

Double-click the dashboard project - or any dashboard in it - and it appears on the emulator screen, drawn the way the device draws it: three icons across, the package's Title image on each, the name underneath in up to two lines, and the dashboard's own name along the top with a counter in the corner.



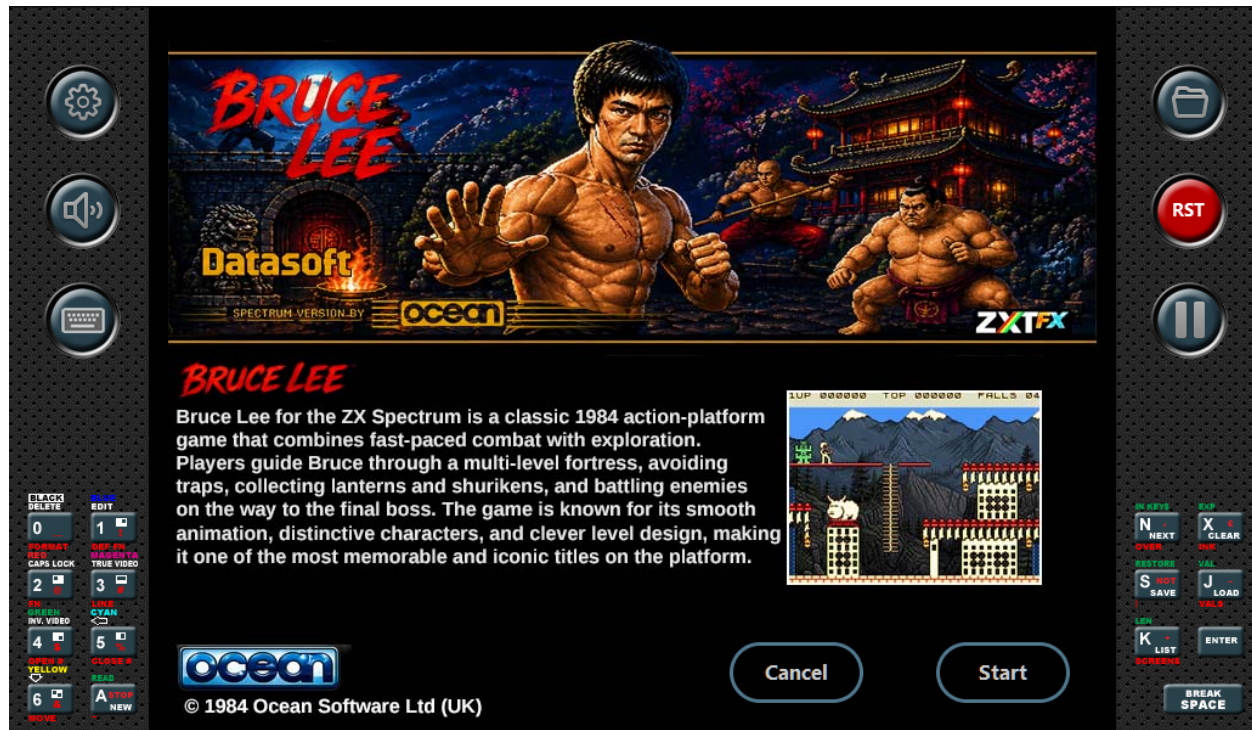
A dashboard on the emulator screen, drawn as the console draws it.

- The arrows at the sides move between dashboards, and appear only when there is one to move to. The counter in the top right corner says which of how many.
- The mouse wheel scrolls a dashboard with more than six entries, and coasts to a stop the way the console does.
- Dragging an icon near the top or bottom edge scrolls the list, so a game can be moved to a row that is off screen.

Activating a dashboard project takes the emulator away from the FX project, which becomes inactive. Its two export buttons grey out, because they export that project and it is no longer what is running. The FX windows stay available: they work on whatever the emulator is holding, so a game started from a dashboard can be examined and adjusted exactly like one opened through File > Open.

Running a game from it

Click an icon and the game starts, exactly as if you had opened that .ztg through File > Open. If the package carries an Info image you get the Info screen first, with the same buttons as the console: Start, Cancel, and - when the package also has info text - Story.



The Info screen, with Start, Story and Cancel.

- Start runs the game.
- Cancel goes back to the dashboard.
- Story shows the text. Its first line, if followed by a blank line, is treated as a heading.

The **folder button** at the top right of the console panel switches between the two views, exactly as it does on the console. One is the browser - the dashboards, the Info screen and the Story text - and the other is the machine. Neither is a step in a sequence: each simply carries on from wherever you left it.

So you might start a game from the first dashboard, press the button, move to the third dashboard and scroll down it, press the button again to go back to the game, and press it once more - and you are on the third dashboard, still scrolled where you were. The game picks up where it stopped, too. With no game started yet, the other view is simply the machine sitting at its boot ROM.

The two views are otherwise independent. Browsing dashboards, switching between them or editing the project does not disturb the game, and the game does not disturb the browser. The one thing that connects them is that a dashboard can start a game - and starting another one is what replaces the first.

Cancel does not end the game either. It takes you from the Info screen back to the dashboard, but the game is still there, and the folder button goes straight back into it, still where you left it. Only starting another game replaces the one that was running.

The first icon - the console's SD-card entry - opens a file dialog here and runs whatever you pick. It is a way to try a file, not to edit the project: nothing is added to the dashboard.

The machine stops while a dashboard is on screen

Whenever the dashboard, an Info screen or the Story text is showing, the emulated machine is held stopped, and the Pause button is greyed out because there is nothing behind it to start. This is not the Pause button being pressed for you: step back into a game and it resumes in whatever state you left it, paused or running.

Exporting

File > Dashboards > Export Dashboards and Library asks where to put the result and writes a single folder named after the project:

```
<Project name>_ZXT_SYSTEM
├─ DASHBOARDS  dashboard1.dsh, dashboard2.dsh, ...
└─ LIBRARY     every .ztg those dashboards use
```

Copy DASHBOARDS and LIBRARY into the ZXT_SYSTEM folder on the SD card and the console shows what you arranged here.

- The export is the whole project, not an addition to it. Dashboards are numbered from 1 every time, and files left over from a previous export of the same project are cleared out. Nothing outside those two folders is touched.
- Two games with the same filename cannot both keep it in one library, so the second is renamed and you are told which.
- A game the project can no longer find stops the export, with the list of what is missing. It is better to be told than to hand the console a dashboard with a gap in it.

The .ztg files themselves are copied untouched. Nothing is re-encoded, so a package that works on the console will work exactly the same way after this.

Why dashboards must be numbered without gaps

The console counts dashboards by looking for dashboard1.dsh, dashboard2.dsh and so on, and stops at the first one that is missing or the wrong size. That is why the export always renumbers from 1, and why it is safer to replace both folders than to merge them by hand.