Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN204388

Getting Started For FM4 S6E2DH (GAT-D)

Associated Part Family: S6E2DH Series

The purpose of this guide is to provide a brief introduction of the graphics resolution of S6E2DH designated by Cypress.

Contents

1 Preface 1
1.1 About This Guide 1
2 Overview 1
3 Utilization of GAT-D Tool Step by Step 2
3.1 Create a New Project 3
3.2 Import Images 4
3.3 Page Creation 6
3.4 Flow Design Table Creation 17
3.5 Source Generation 21
3.6 Add Generated Source to IAR project 22
3.7 Download and Run after Build 24
4 Application of GAT-D 26
4.1 Event Management 26
4.2 User's Source Code 27
4.3 Display Simulation 29
4.4 Sprite Special Effect 30
4.5 Touch Panel Resolution 31
4.6 Source Generate Option 32
5 GAT-D API 34
5.1 Interface Description 35
5.2 Join with Source 37
6 Program 38
6.1 Debugging Mode 38
6.2 Product Mode 39
7 Sample Project Instruction 39
7.1 Construction 39
7.2 Function Overview 41
Document History 42

1 Preface

1.1 About This Guide

The purpose of this guide is to provide a brief introduction of the graphics resolution of S6E2DH designated by Cypress. This guide explains:

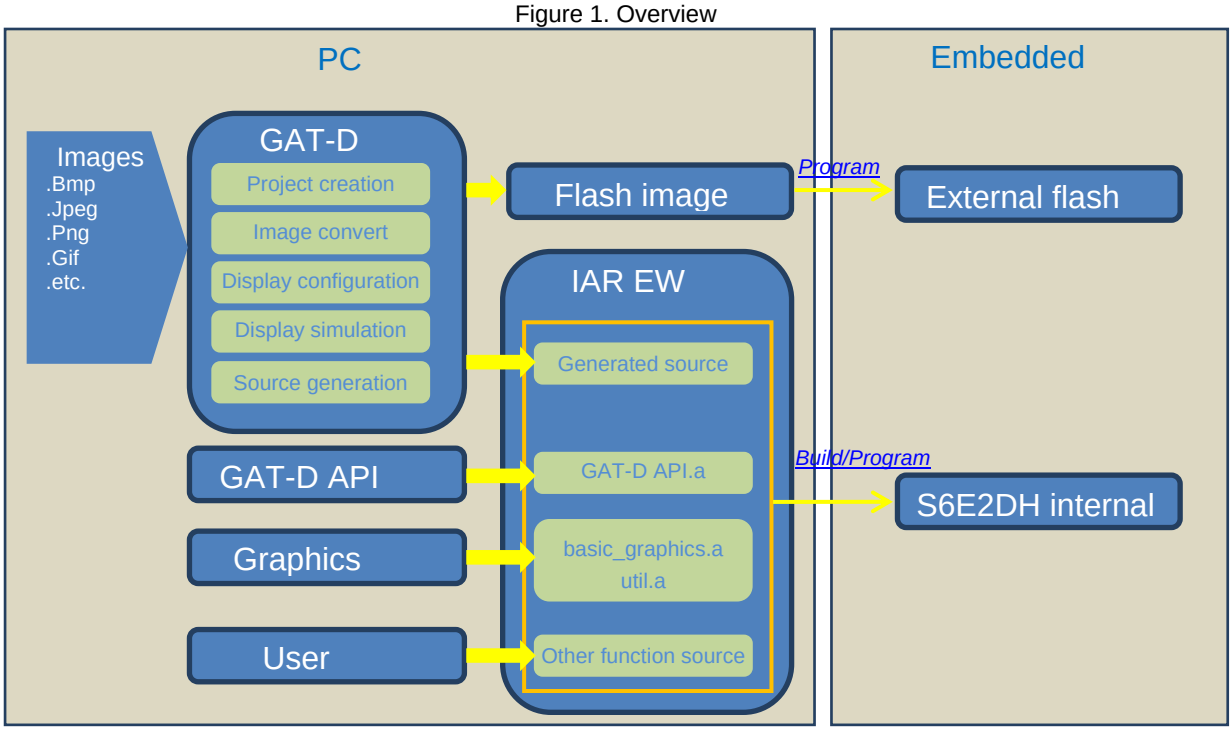
1. How to work with Graphics Authoring Tool for S6E2DH (GAT-D).
2. Advanced application of GAT-D.
3. How to integrate the generated source and Graphics Driver to embedded software.
4. Program to internal Flash and external Flash.
5. Construction of sample.

Note: This guide is described with example while using EWARM of IAR Systems AB. For other IDE please refer it properly.

2 Overview

Follow figure shows the exact relationship of each component.

Figure 1. Overview



GAT-D Tool and GAT-D API Library could be found in S6E2DH-Graphics resolution-Ver1.0.zip.

GAT-D Tool: S6E2DH-Graphics resolution-Ver1.0\01. GAT-D Tool\01. Installer\GAT-D_xxx.exe

GAT-D API Library: S6E2DH-Graphics resolution-Ver1.0\02. GAT-D API Library\01. IAR\01. bin\GAT-D API.a

Graphics Driver: In the package of S6E2D-2DGraphicsDriver-v1.00.798-IAR.zip

S6E2D-2DGraphicsDriver-v1.00.798-IAR \01_bin\arm_none_iar\release\basic_graphics.a

S6E2D-2DGraphicsDriver-v1.00.798-IAR \01_bin\arm_none_iar\release\util.a

3 Utilization of GAT-D Tool Step by Step

The follow steps explain how to use GAT-D Tool to arrange a video (a series of frame) to display on S6E2DH Series based LCD device.

1. Create a new project
2. Import images
3. Page creation
4. Flow Design Table creation
5. Source generation
6. Add generated source to IAR project
7. Download and run after build

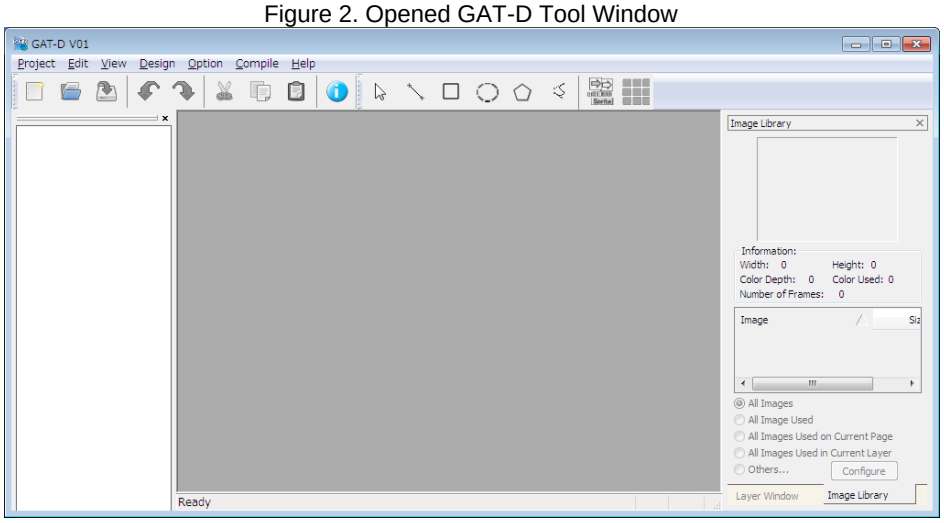
Also assumed conditions:

1. GAT-D Tool installed
2. IAR Embedded Workbench installed (ver.7.10 or upper)
3. Image material prepared
4. A S6E2DH series based LCD device and ICE (I-Jet recommend) prepared

3.1 Create a New Project

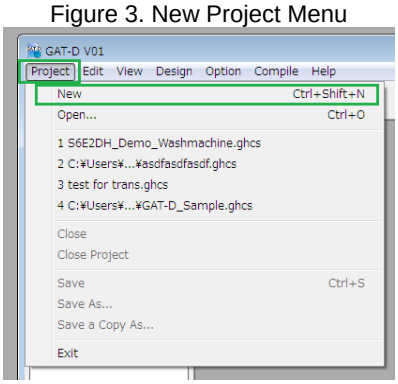
Step1: Open GAT-D Tool, the tool will open as follow.

Figure 2. Opened GAT-D Tool Window



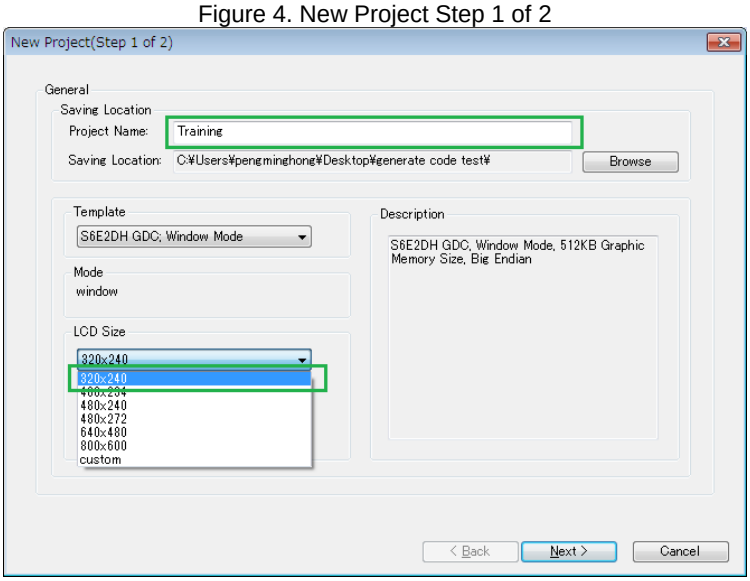
Step2: Click “Project” menu and click “new” as follows

Figure 3. New Project Menu



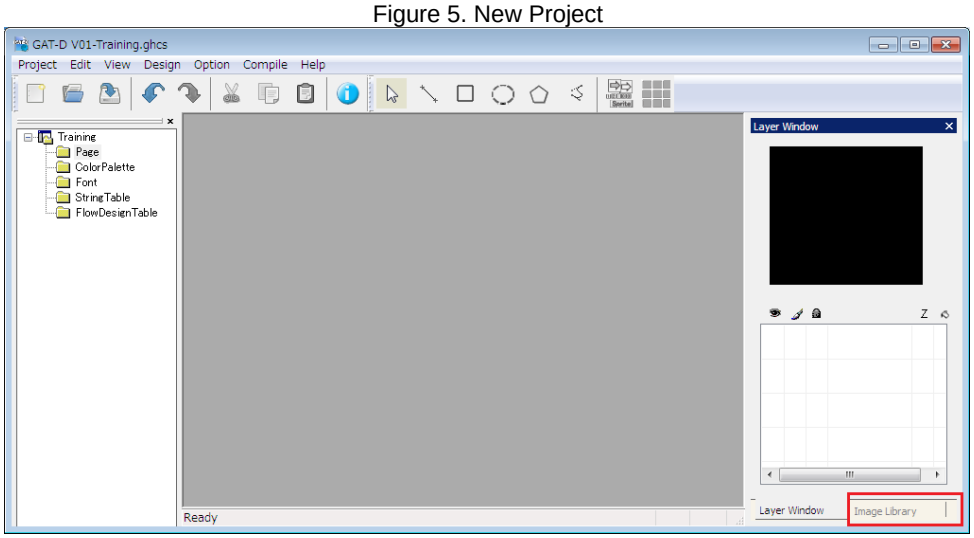
Step3: Input the name and select LCD Size.

Figure 4. New Project Step 1 of 2



Step4: Click “Next” and then click “Finish” a new project will create as follow.

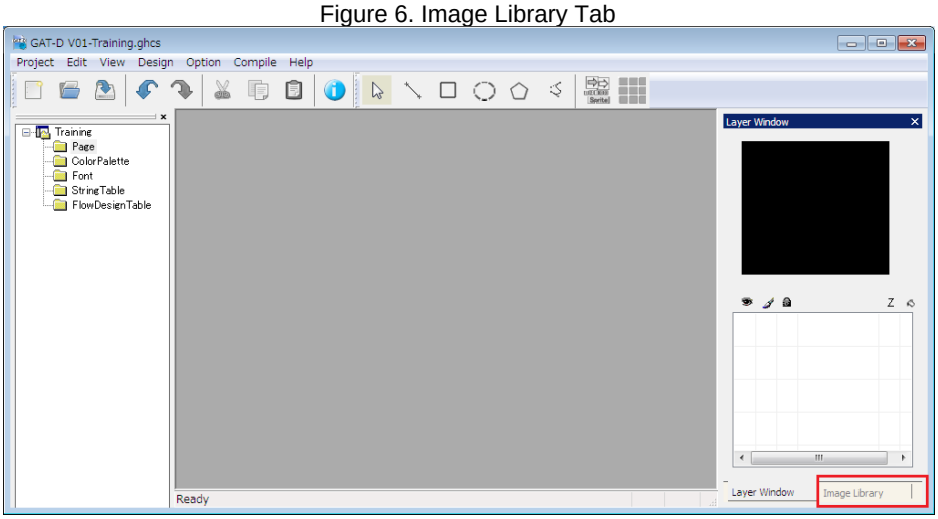
Figure 5. New Project



3.2 Import Images

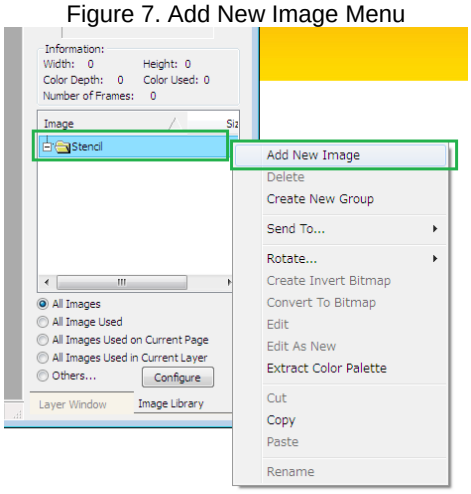
Step1: Click “Image Library”

Figure 6. Image Library Tab



Step2: Right click “Stencil” in “Image Library”, and Click “Add New Image” in the menu.

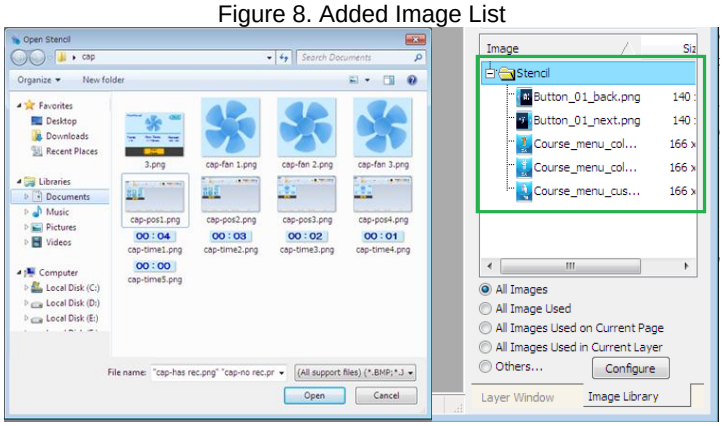
Figure 7. Add New Image Menu



Step3: Go to the path and select images, image will list as follow.

Note: Image’s file type support bmp, png, jpeg, gif, psd.

Figure 8. Added Image List

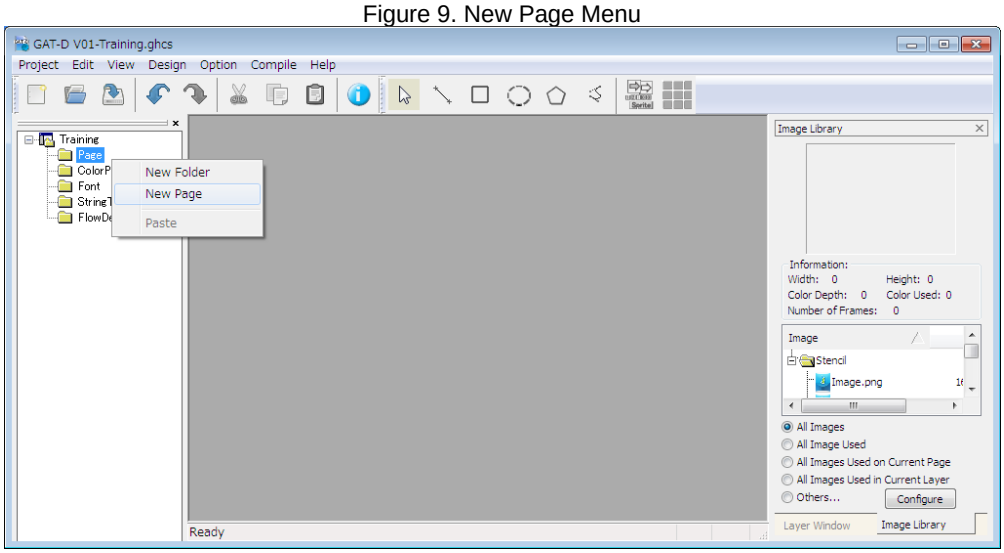


3.3 Page Creation

3.3.1 Create a new Page

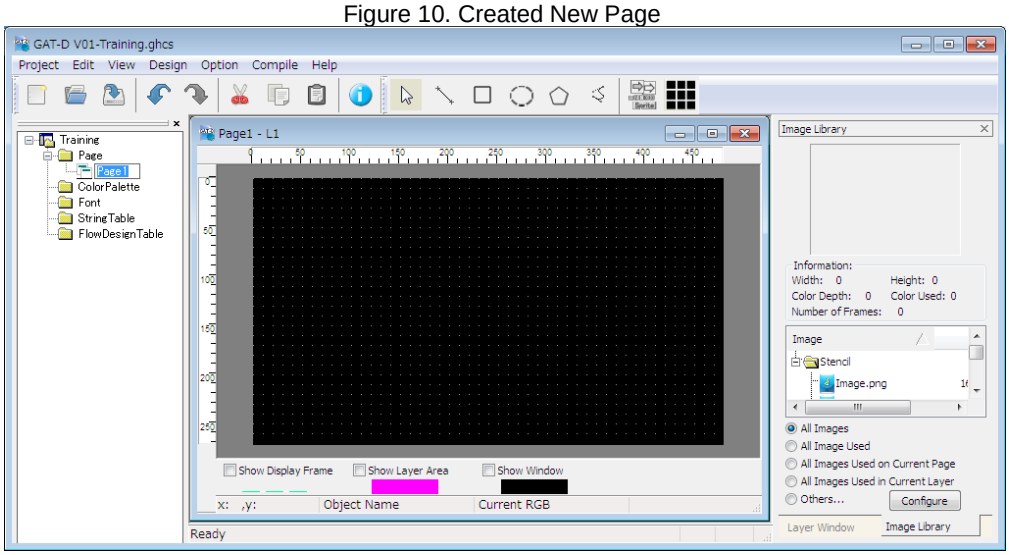
Step1: Right click “Page” folder, and click “New Page” in the menu.

Figure 9. New Page Menu



Then a new page is created as follow.

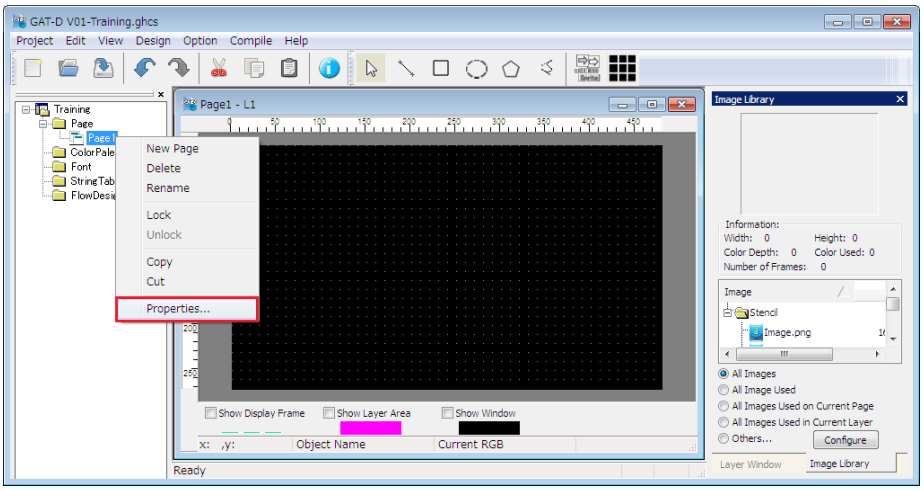
Figure 10. Created New Page



3.3.2 Page Configuration

Step1: Right Click “Page1” and click “Properties…” in menu

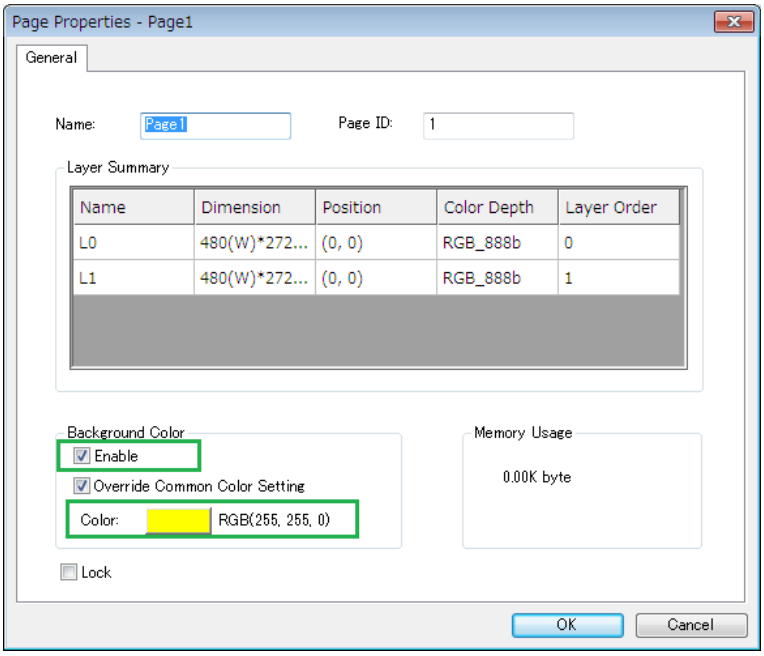
Figure 11. Page Property Menu



Step2: Check “Enable”, the background color of page could be changed.

Select RGB Color, if there is nothing on the page, LCD will display background color.

Figure 12. Page Property Window

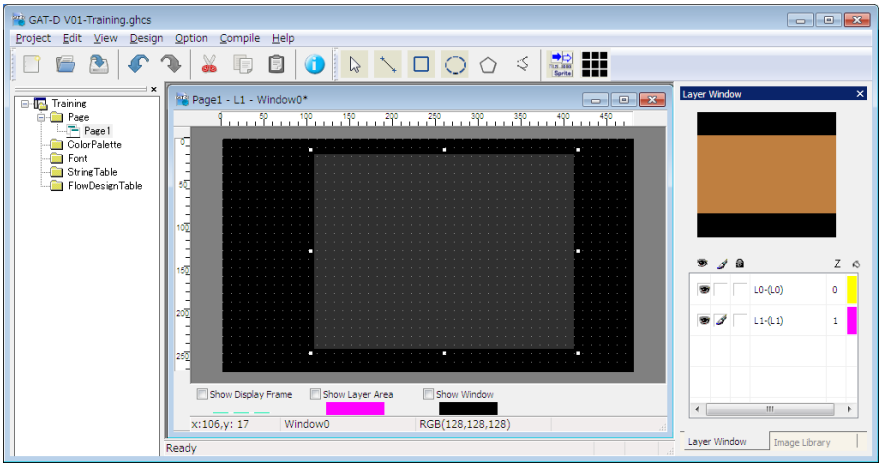


3.3.3 Add Window to Page

Step1: Select window from drawn tool bar, left click to work area, window is added as follow.

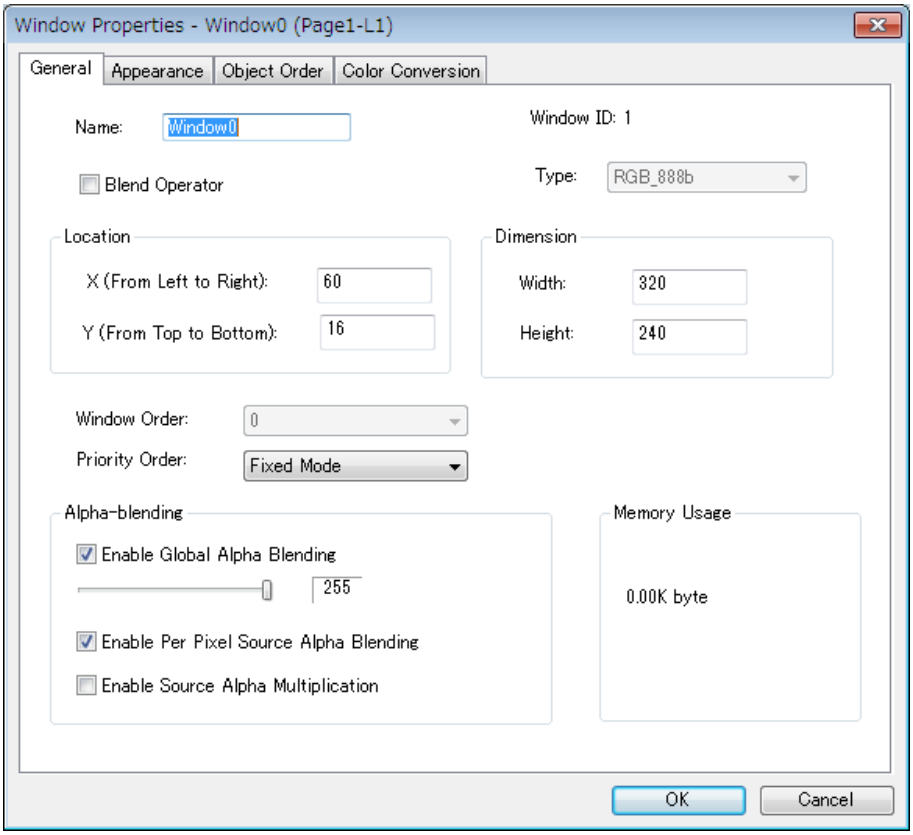
If want to add several window to a layer, you have to set the layer option to multi-window layer, otherwise could only add one window to a layer

Figure 13. Add Window



Step2: Double click window in work area, window configuration window will displays. Configure them.

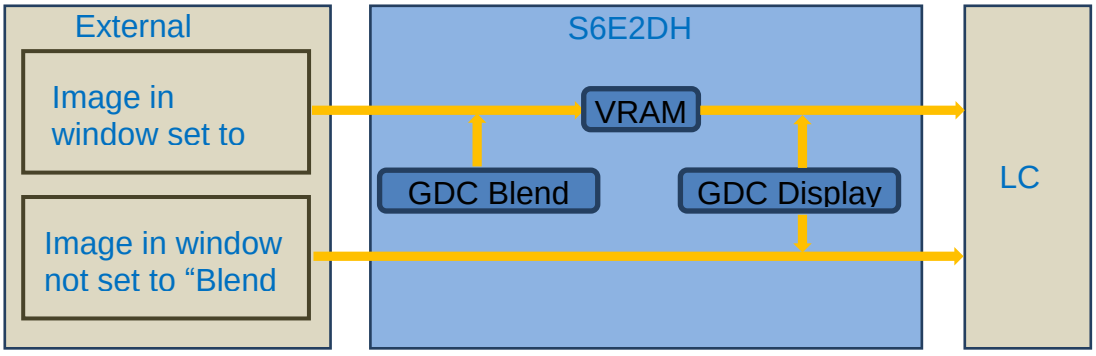
Figure 14. Window Property



If want add several image to one window, you have to check “Blend Operator”

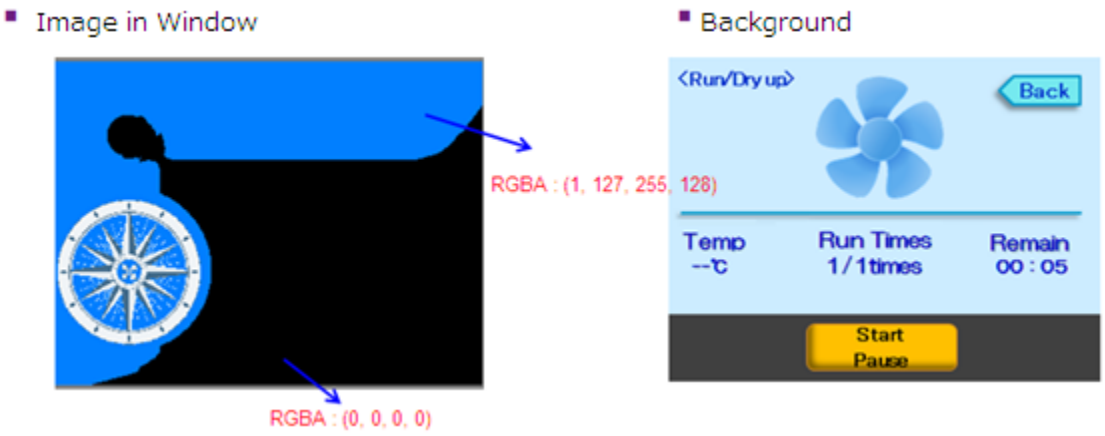
Figure 3-14 shows difference of image in windows that are set to “Blend Operator” or not

Figure 15. Blend Operator



The setting of “Alpha-blending”, will effects blend result of image on the window with background.

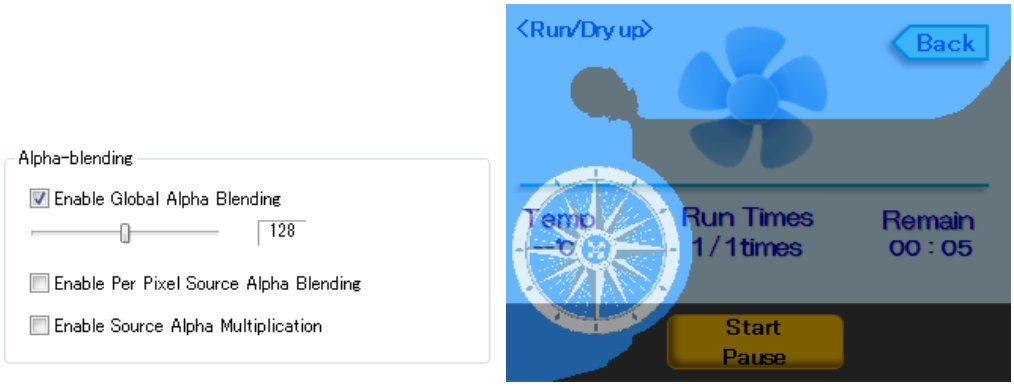
Figure 16. Window Alpha Blend Illustrations



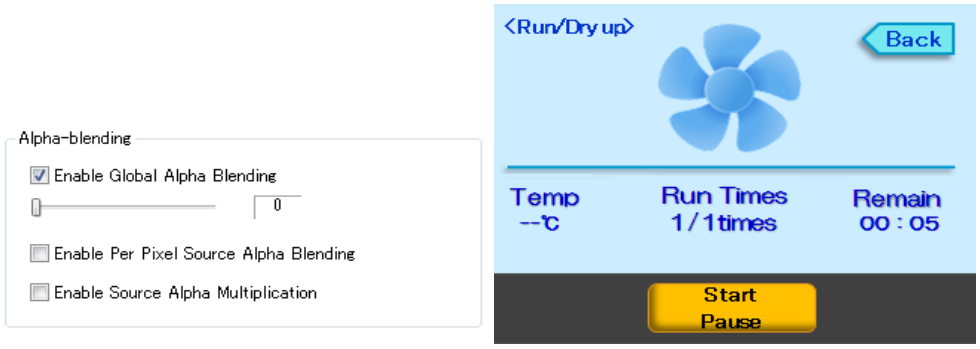
Illustrations A:



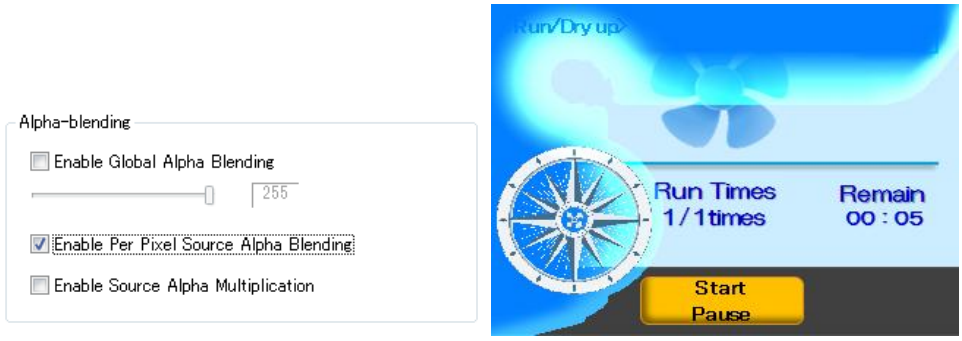
Illustrations B:



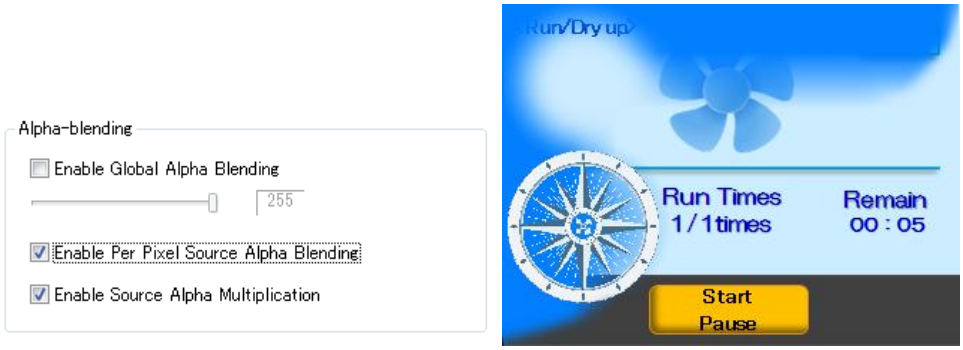
Illustrations C:



Illustrations D:



Illustrations E:



Illustrations F:

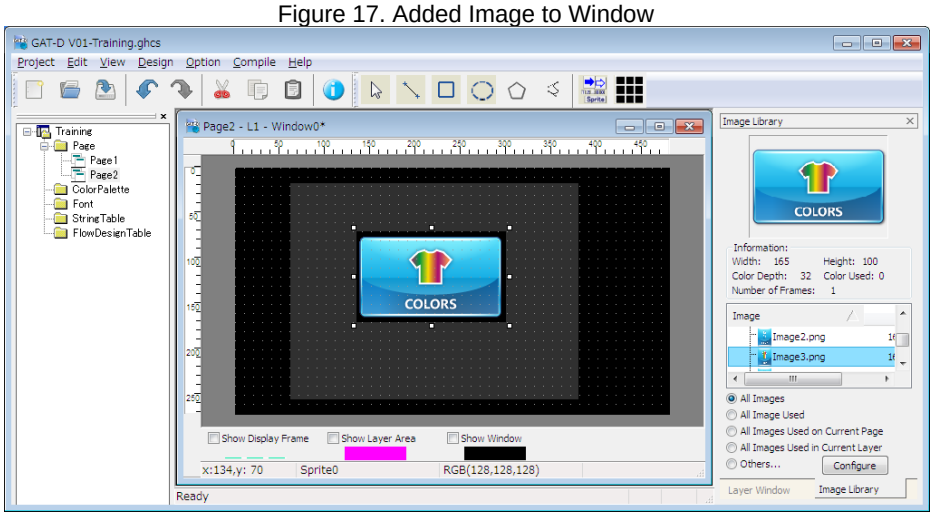


Above illustration are from page1 of sample in S6E2DH-Graphics resolution-Ver1.0\03. Sample\01. GAT-D_Tool\01. Training\Training.zip.

3.3.4 Add Images to Window

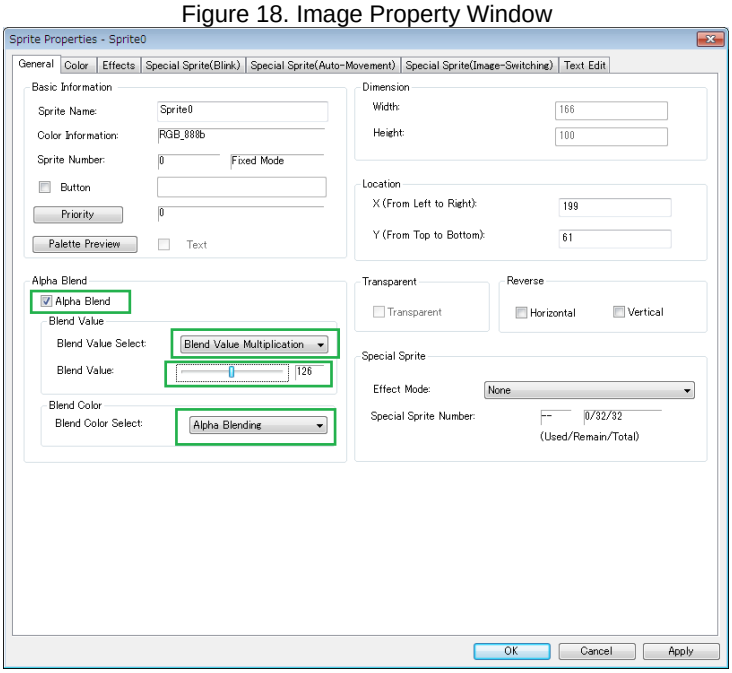
Step1: Select image from “Image Library”, drag it to a selected window and drop, image is added as follow.

Figure 17. Added Image to Window



Step2: Double click image, image configuration window will displays. Configure them

Figure 18. Image Property Window



For behavior of Alpha Blend configuration, refer to following illustrations.

Figure 19. Alpha Blend Illustrations

Image used:

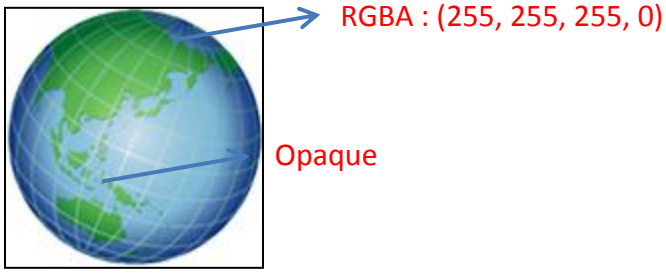


Illustration A:

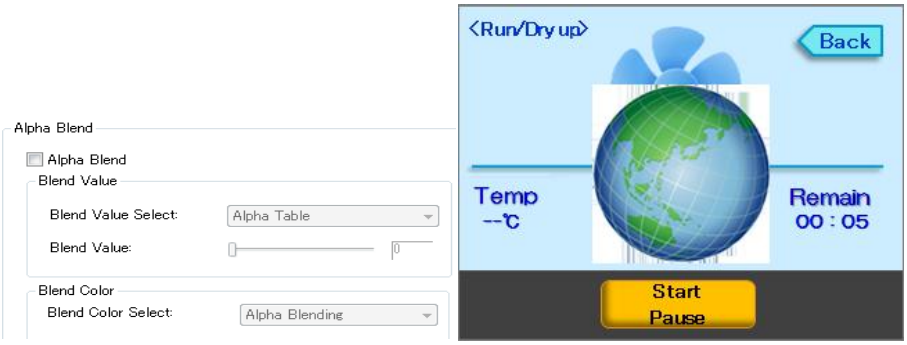


Illustration B:

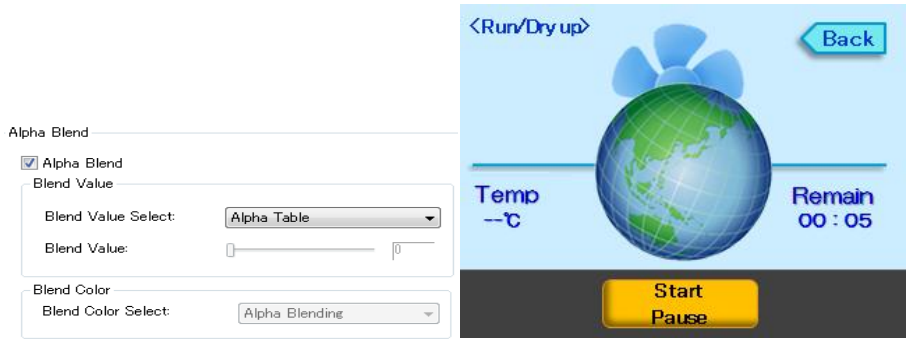


Illustration C:

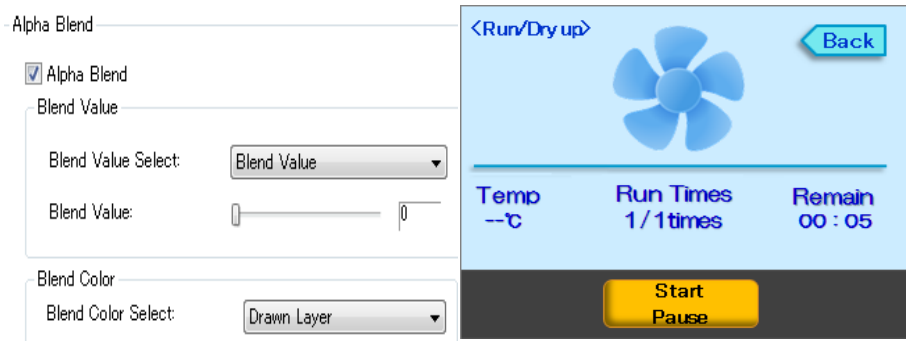


Illustration D:

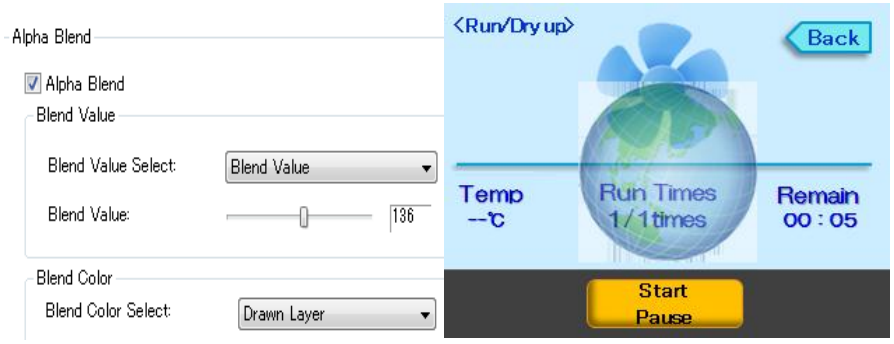


Illustration E:

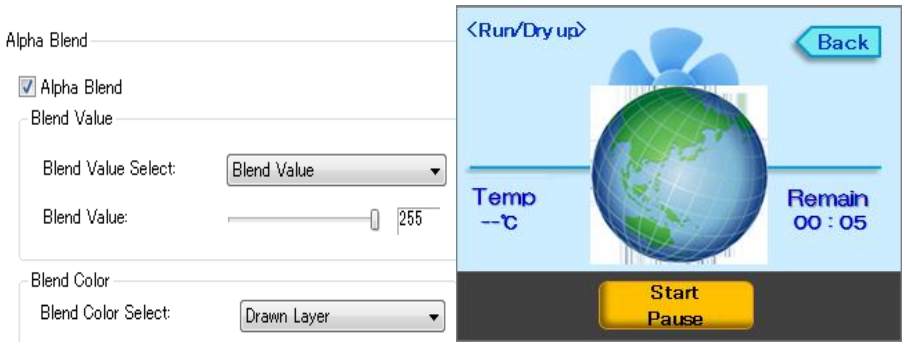


Illustration F:

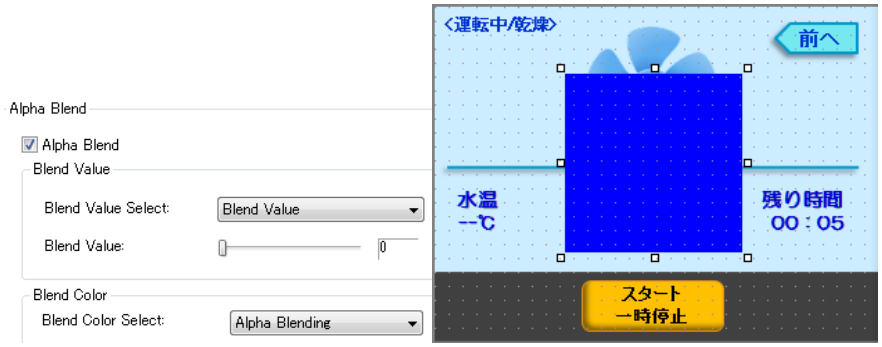


Illustration G:

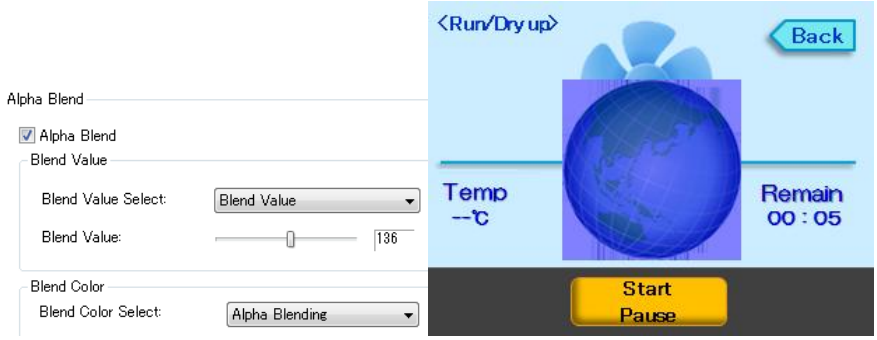


Illustration H:

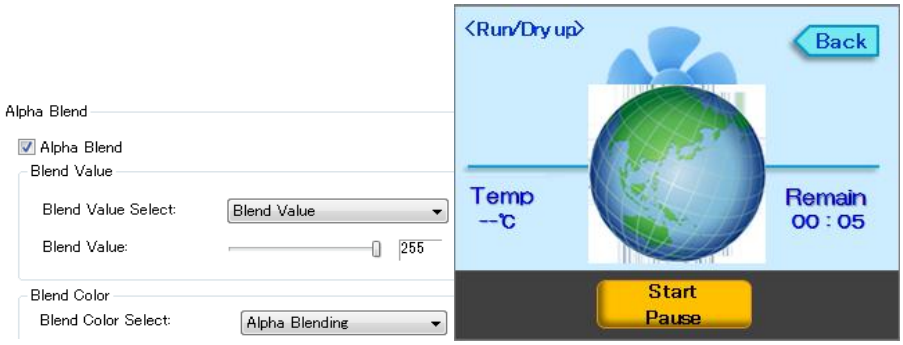


Illustration I:

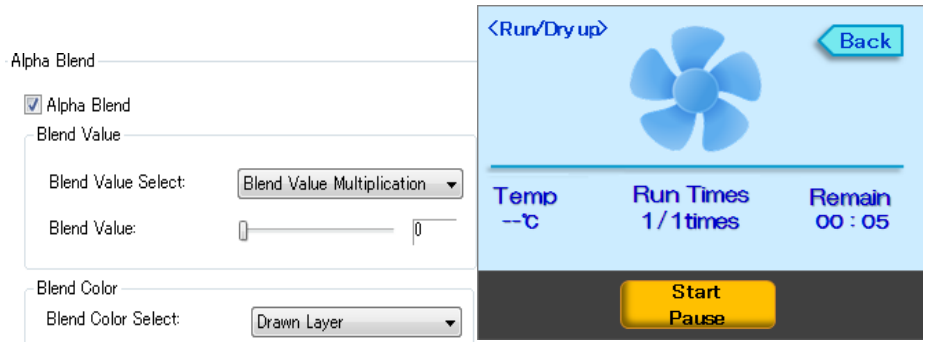


Illustration J:

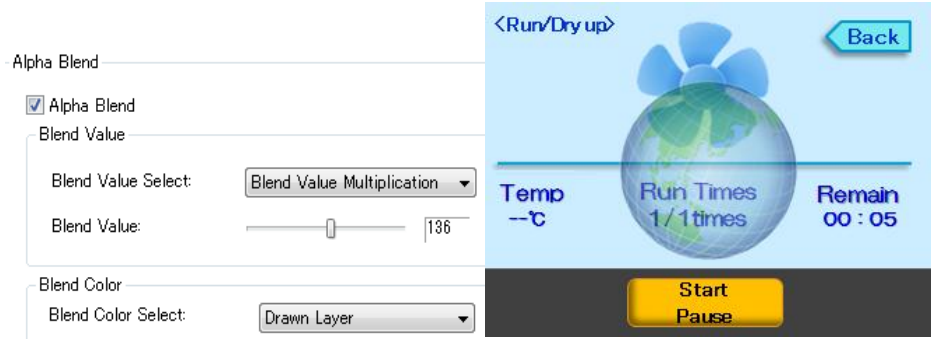


Illustration K:

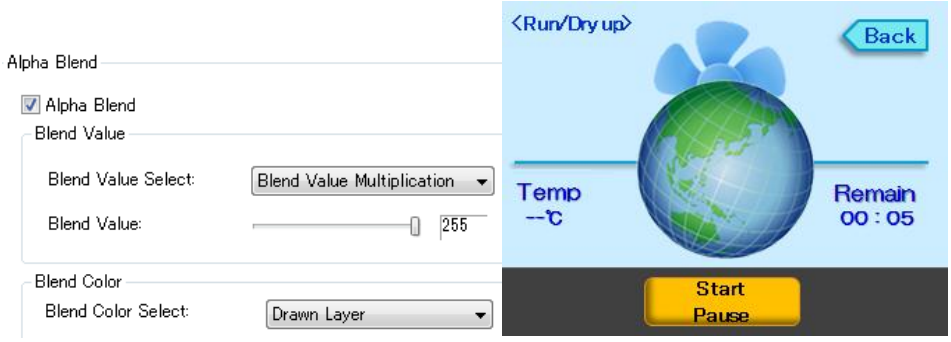


Illustration L:

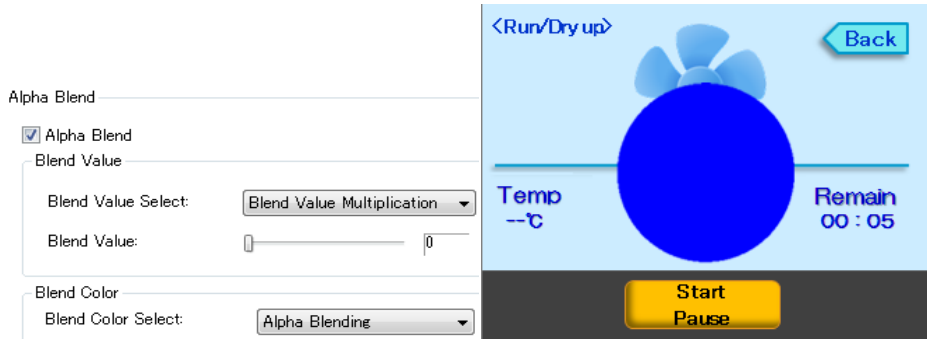


Illustration M:

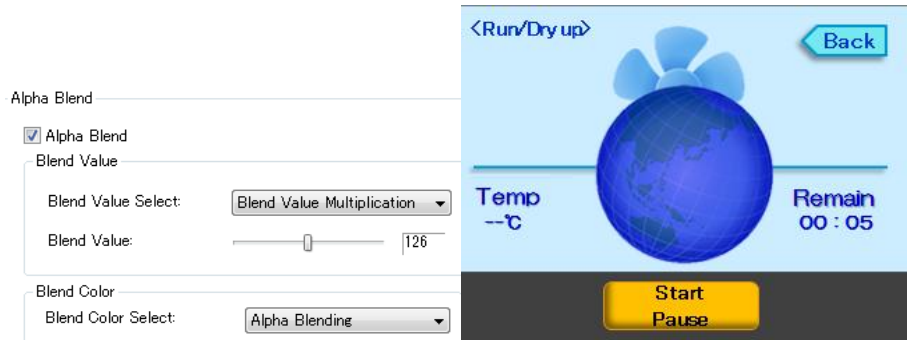
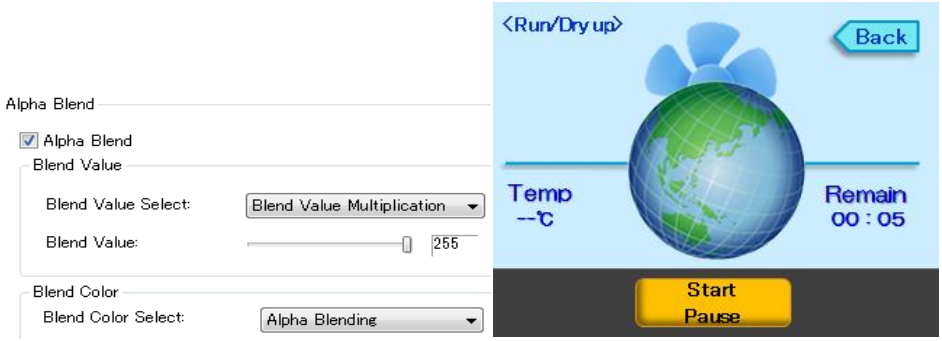


Illustration N:

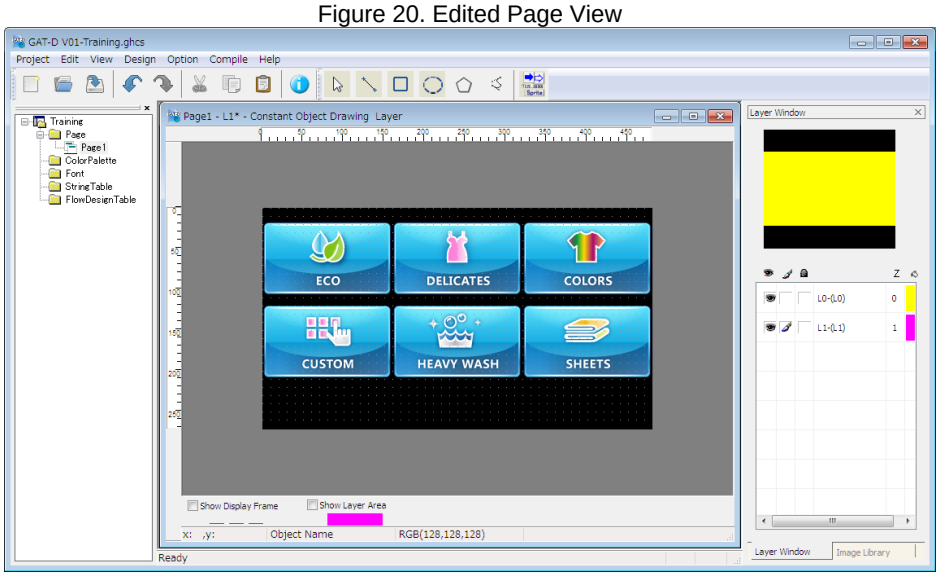


Above illustration are from page4 of sample in S6E2DH-Graphics resolution-Ver1.0\03. Sample\01. GAT-D_Tool\01. Training\Training.zip.

Step3: Add needed images and configure them.

Note: After configuration of a page, could preview it from “View” -> “Page Preview Window”.

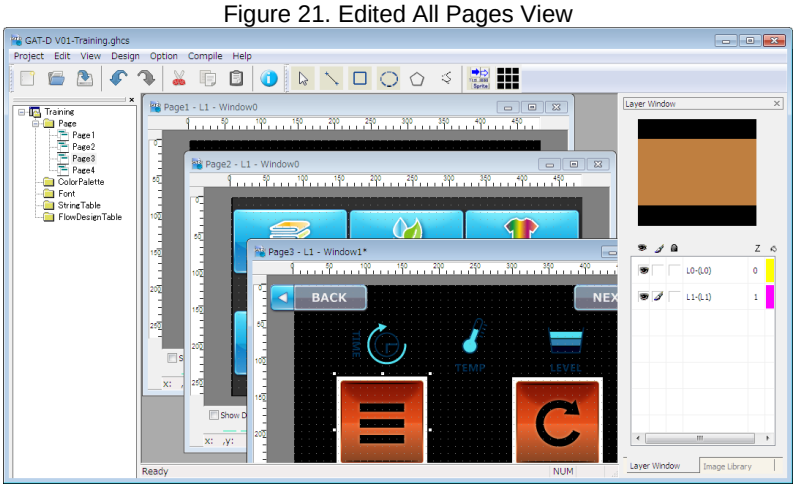
Figure 20. Edited Page View



3.3.5 Create All Pages

Create more pages if needs.

Figure 21. Edited All Pages View

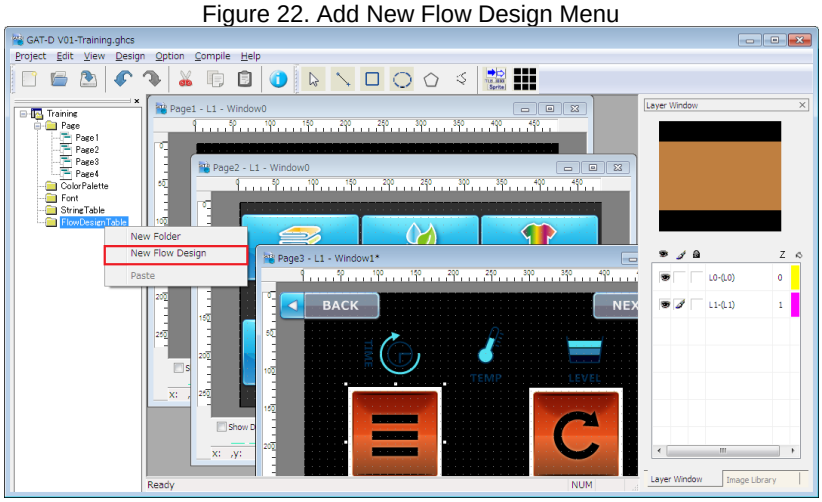


3.4 Flow Design Table Creation

3.4.1 Add a Flow Design

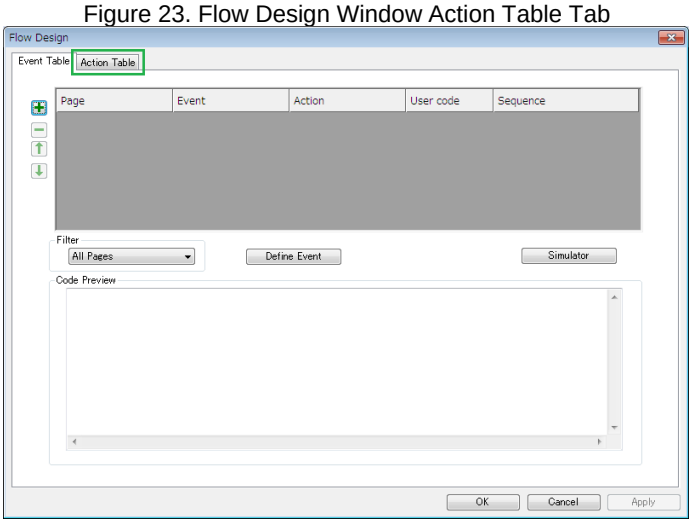
Step1: Right click “FlowDesignTable” folder, and click “New Flow Design” in the menu.

Figure 22. Add New Flow Design Menu



Step2: Double click the new Flow Design, window displays as follow, click “Action Table” directly because don’t use event if display startup only.

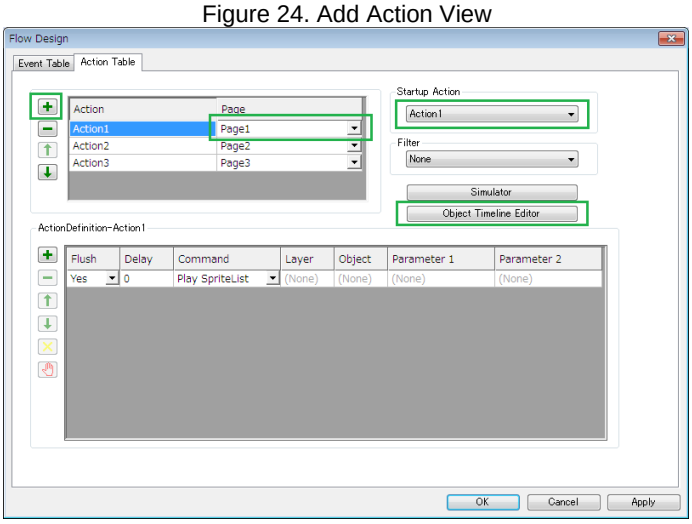
Figure 23. Flow Design Window Action Table Tab



Step3: Click “+” to add actions, and select page for action to display.

Note: Action is something like a small video (a series of frame),

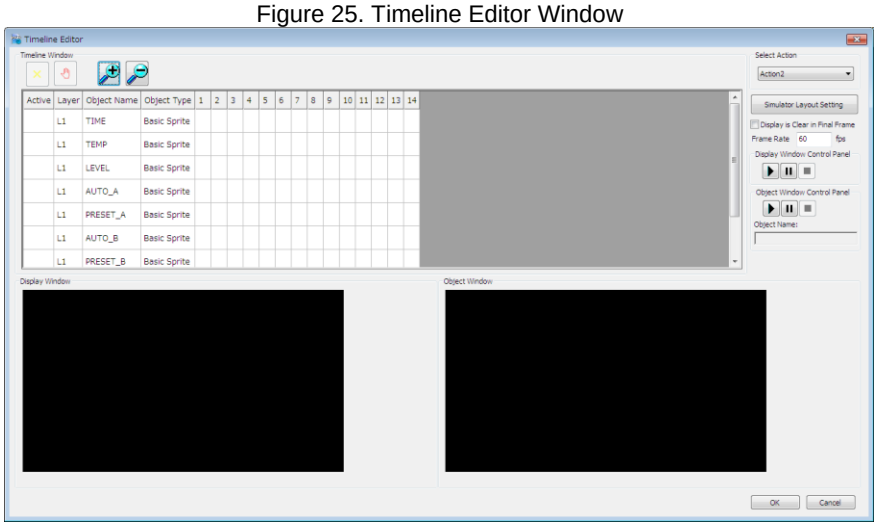
Figure 24. Add Action View



Step4: Select “Startup Action”, which to decide the action to be displayed after power on.

Step5: Click “Object Timeline Editor”, “Timeline Editor” window will display.

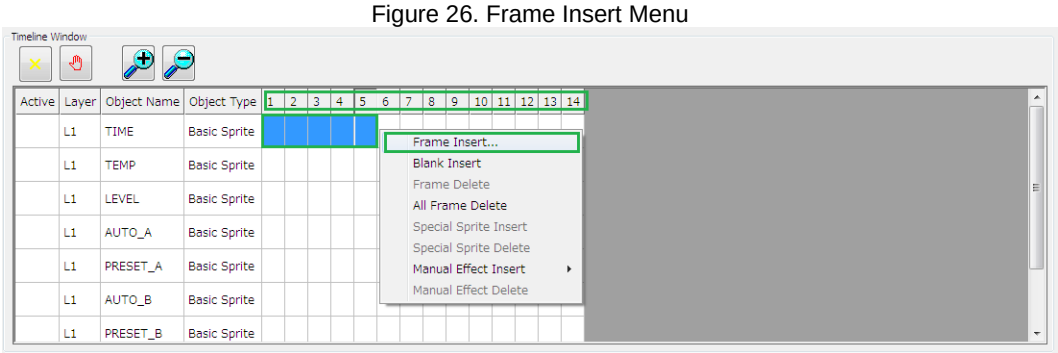
Figure 25. Timeline Editor Window



Step6: Select the frames where want to display the image, double click then select “Frame Insert.”

Note: The head number is frames.

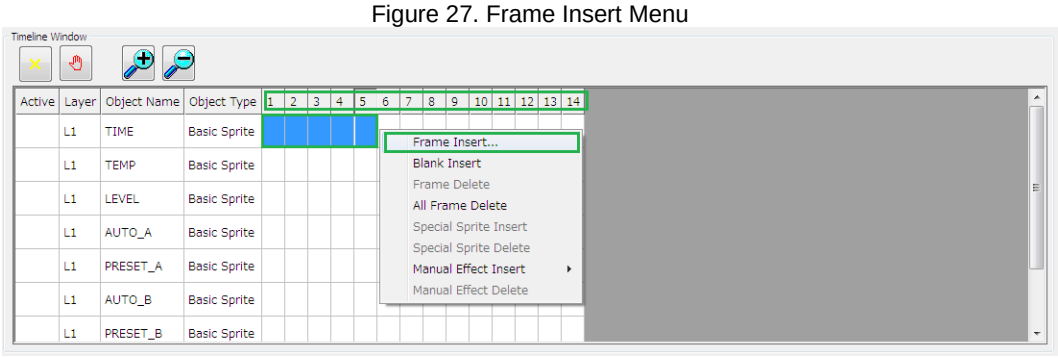
Figure 26. Frame Insert Menu



Step6: Select the frames where want to display the image, double click then select “Frame Insert…”

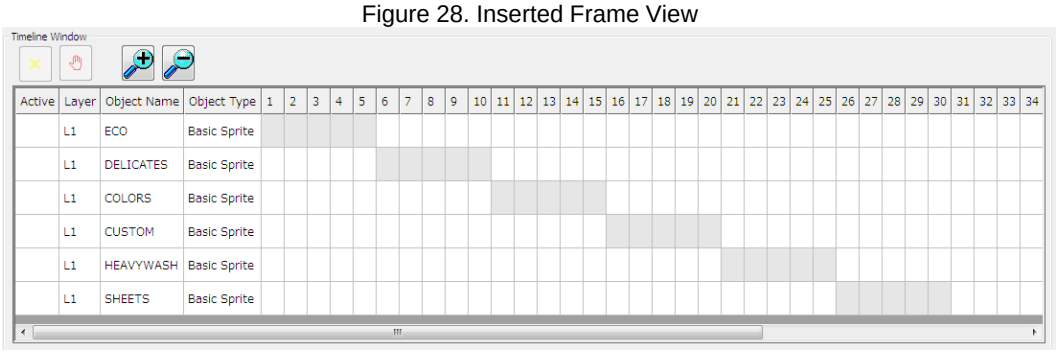
Note: The head number is frames.

Figure 27. Frame Insert Menu



Step7: Insert frame for each image. If display each image 5 frames and then next, the time line is as follow.

Figure 28. Inserted Frame View

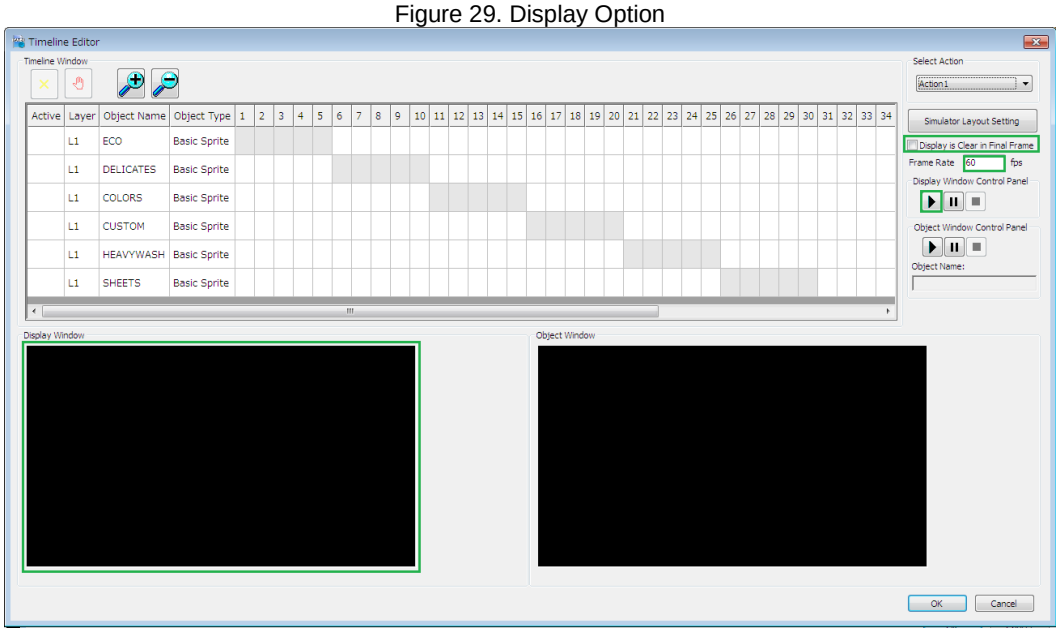


Step8: Check “Display is Clear in Final Frame” if display nothing after action play over.

Change “Frame Rate” to 60fps because frame rate of S6E2DH series is 60fps.

Click “▸” can simulate action in Display window.

Figure 29. Display Option



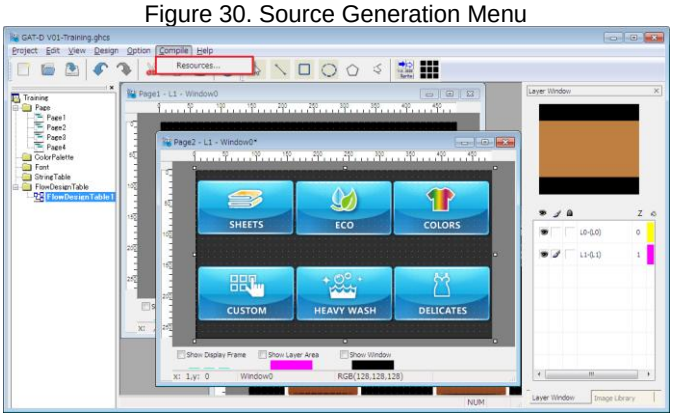
3.4.2 Add All Flow Design

More Flow Design also could be added, but one Flow Design is enough for one program.

3.5 Source Generation

Step1: Click “Compile” and then click “Resource…” to open “Operation Code and Data Library Generation” window.

Figure 30. Source Generation Menu

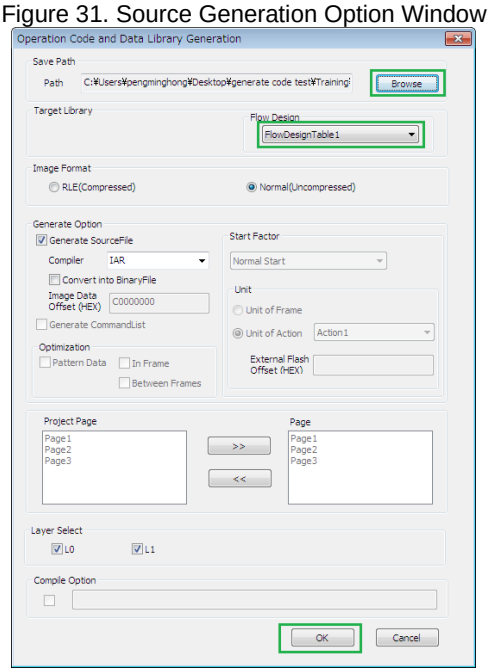


Step2: Click “Browse” to select path where source generated to.

Select Flow Design if there are any.

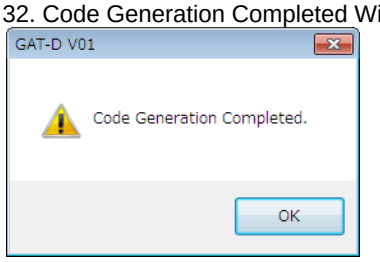
Click “OK” to generate source with other options default.

Figure 31. Source Generation Option Window



Step3: Follow window means source generation success, and generated source could be found in selected path in step2.

Figure 32. Code Generation Completed Window

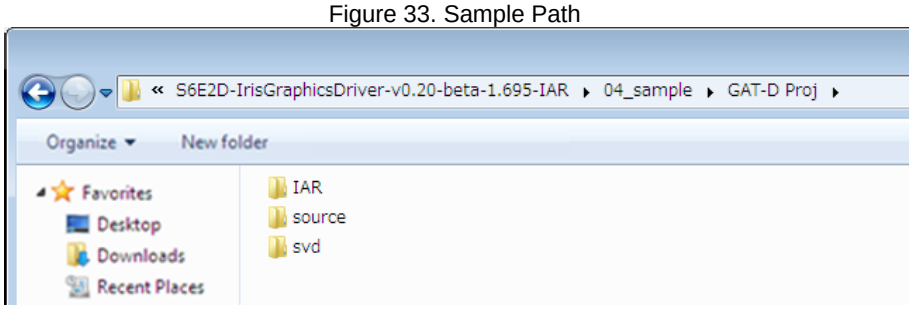


3.6 Add Generated Source to IAR project

Step1: Get Graphics Driver package S6E2D-2DGraphicsDriver-v1.00.798-IAR.zip, extract it to any path.

Find the S6E2DH-Graphics resolution-Ver1.0-IAR\03. Sample\02. IAR\GAT-D Proj.zip and extract it to 04_sample of Graphics Driver's extracted path.

Figure 33. Sample Path



Step2: Open IAR project.

Figure 34. Sample Project File

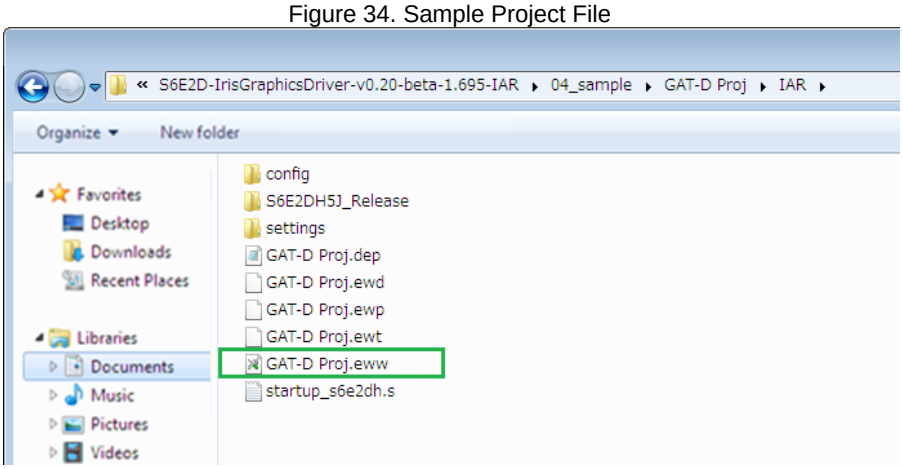
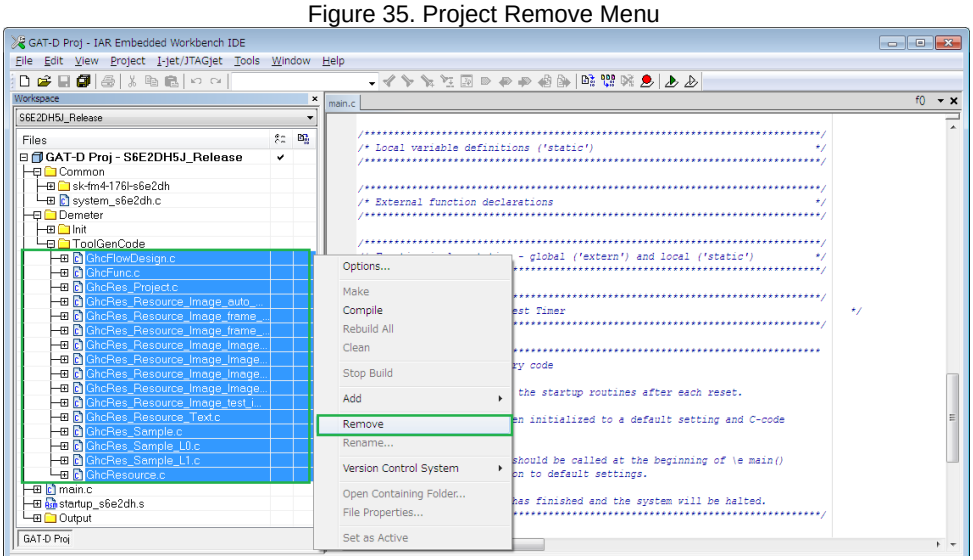
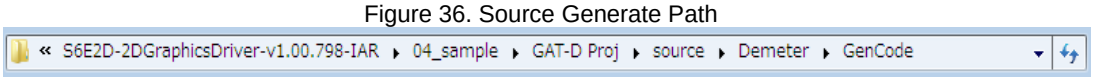


Figure 35. Project Remove Menu



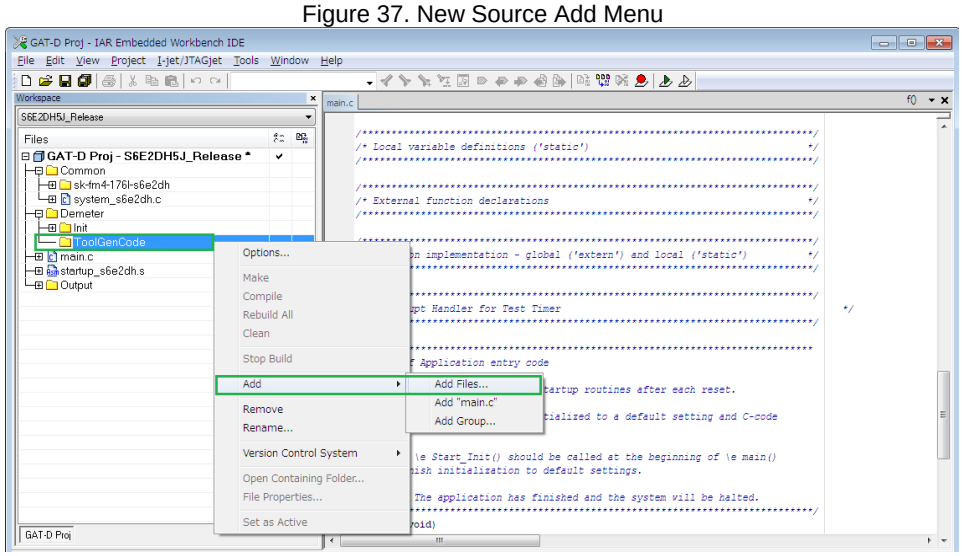
Step4: Delete all source in path: S6E2D-2DGraphicsDriver-v1.00.798-IAR.zip\04_sample\GAT-D Proj\source\Demeter\GenCode , then copy all new generated source the path.

Figure 36. Source Generate Path



Step5: Add all new generated sources (*.c files only are OK) from S6E2D-2DGraphicsDriver-v1.00.798-IAR \04_sample\GAT-D Proj\source\Demeter\GenCode to IAR project.

Figure 37. New Source Add Menu



Note: The minimum call of API is implemented in main.c.

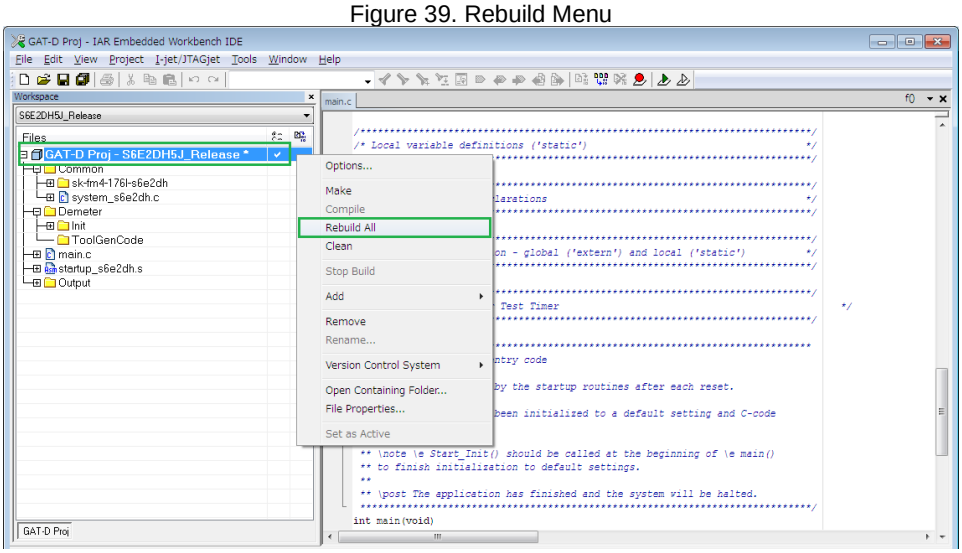
Figure 38. Minimum Call of API

```
if (GhcTrue == ret) {
    ret = GhcProject_load(0, 0);
}
if (GhcTrue == ret) {
    ret = GhcIrisInitialize();
}
if (GhcTrue == ret) {
    ret = GhcProject_setStartupAction();
}
if (GhcTrue == ret) {
    ret = GhcProject_RunAction();
}
```

3.7 Download and Run after Build

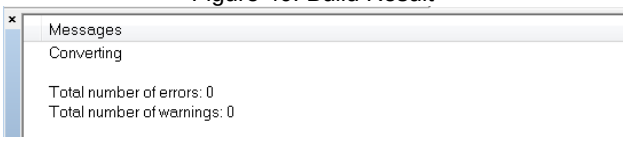
Step1: Right click IAR project group, select “Rebuild All” in the menu to build project.

Figure 39. Rebuild Menu



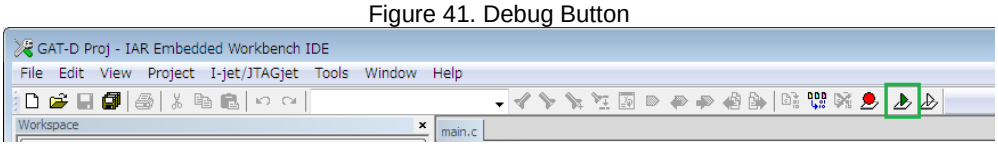
Step2: Confirm no errors.

Figure 40. Build Result



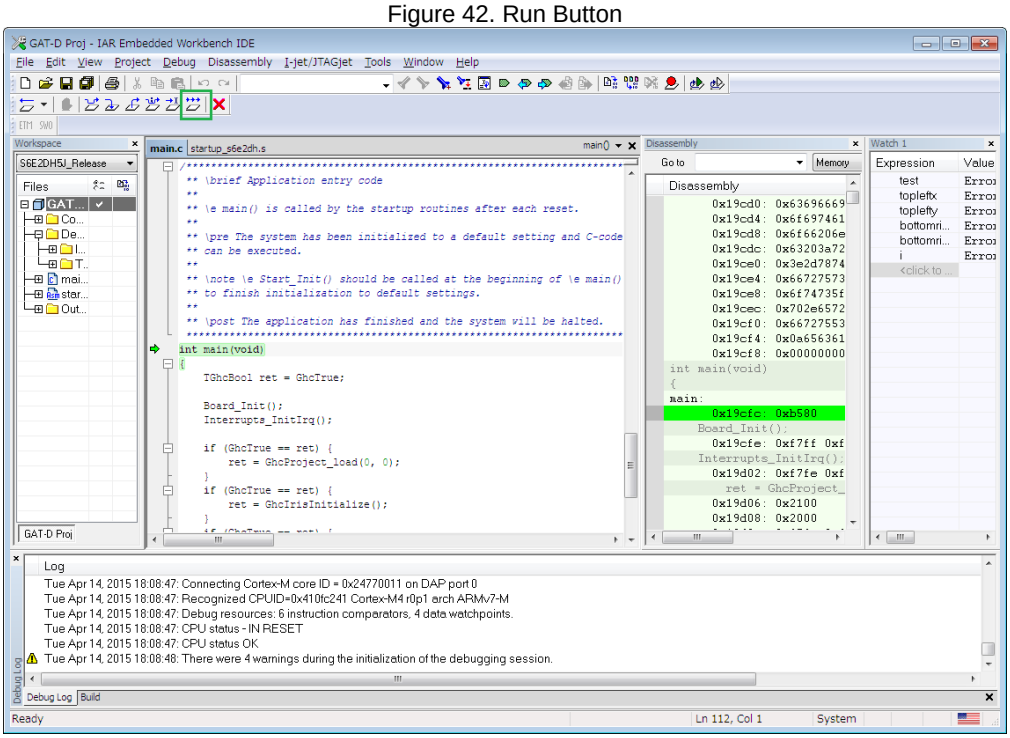
Step3: Click “Download and Debug” button on IAR EW IDE to down program

Figure 41. Debug Button



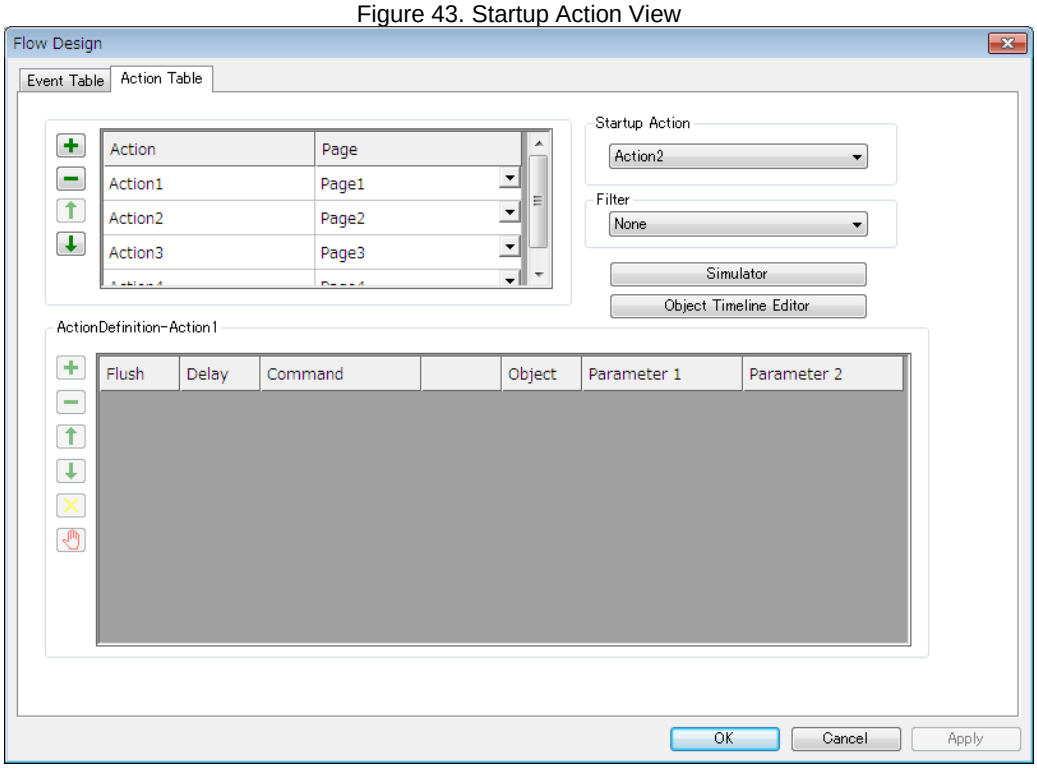
Step4: Click “Go” button to run.

Figure 42. Run Button



Step5: Confirm display on LCD is the same action selected on “Action Table”.

Figure 43. Startup Action View

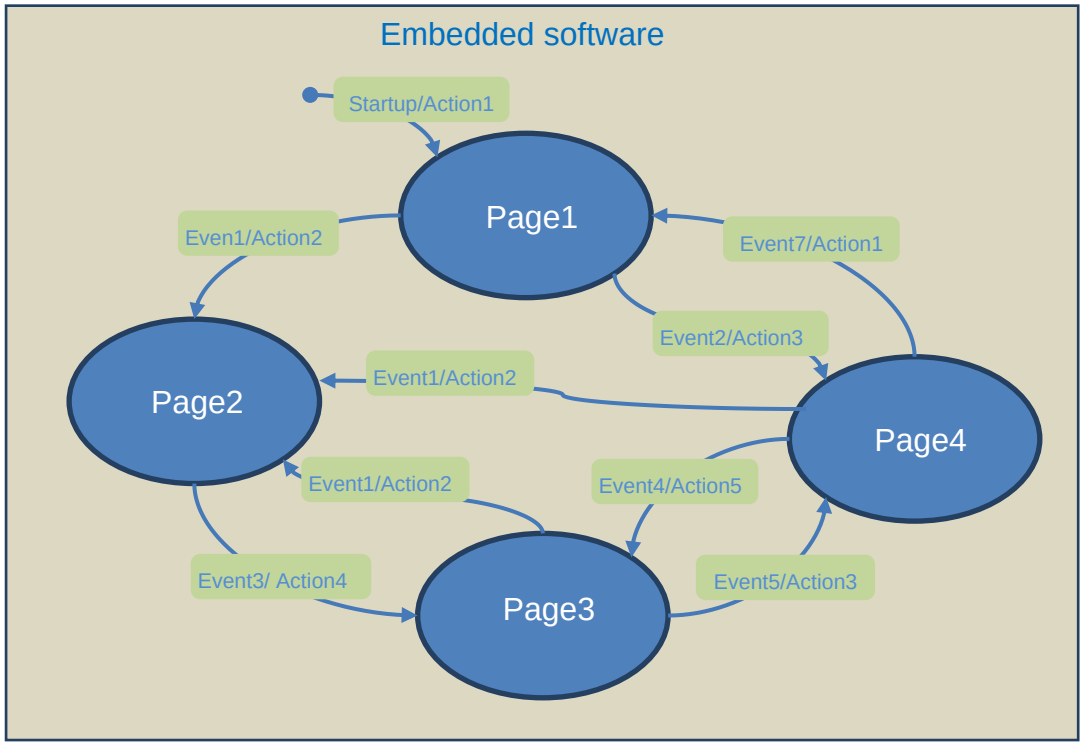


4 Application of GAT-D

4.1 Event Management

Follow figure described common model of a graphics based embedded software.

Figure 44. Embedded Software Model



Use GAT-D, above function can be described in PC side, and generate source automatically, instead of work in embedded side. With this, in embedded side only need to implement event confirmation, if one event is triggered, Call GAT-D API: GhcProject_sendEvent_Ex(eGEN_xxxx) the referenced action will be displayed.

xxxx: event name. See 5.1.6 also for details.

Follows are the sample.

Figure 45. Sample of Action Table

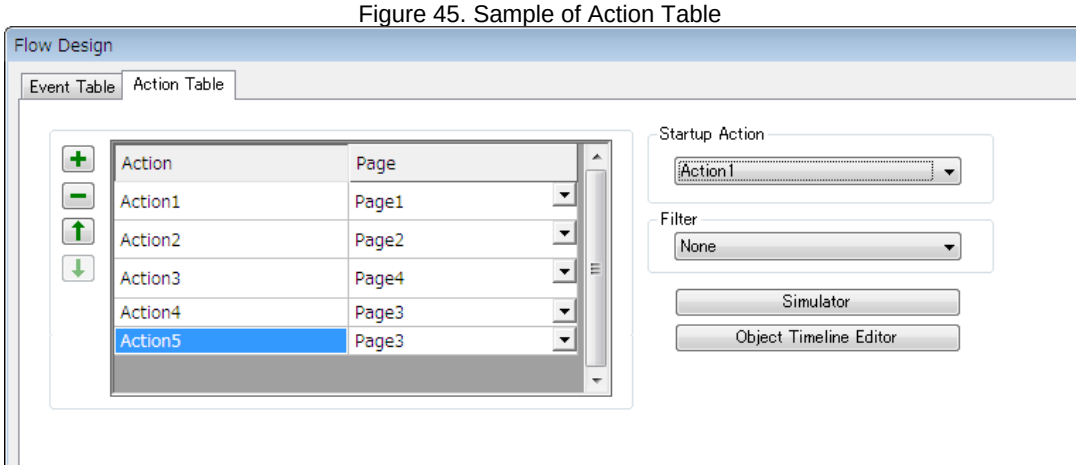
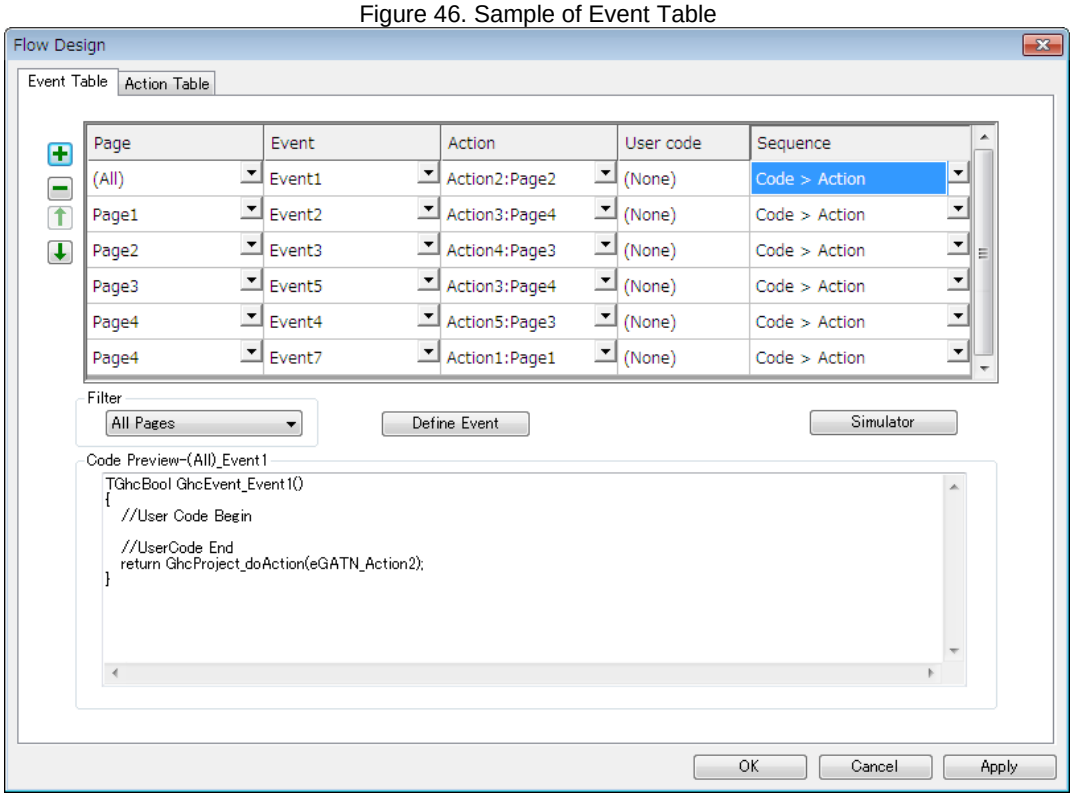


Figure 46. Sample of Event Table



Above illustration are from Flow Design of sample in S6E2DH-Graphics resolution-Ver1.0\03. Sample\01. GAT-D_Tool\01. Training\Training.zip.

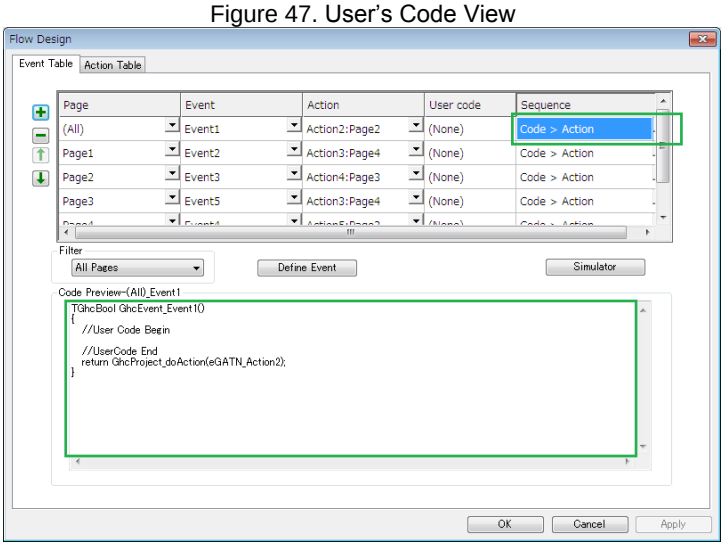
For event type, there are three kinds: System, Hardware and Object. In PC side there is no difference actually.

4.2 User's Source Code

Beside with event management, GAT-D provides a place for user to input standard C source code.

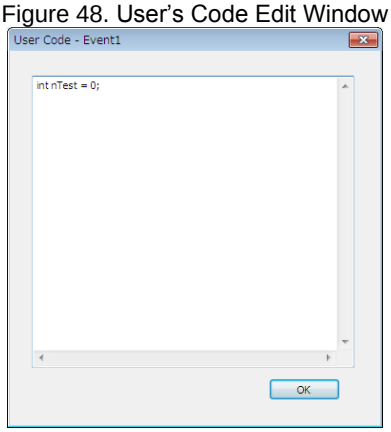
As follows:

Figure 47. User's Code View



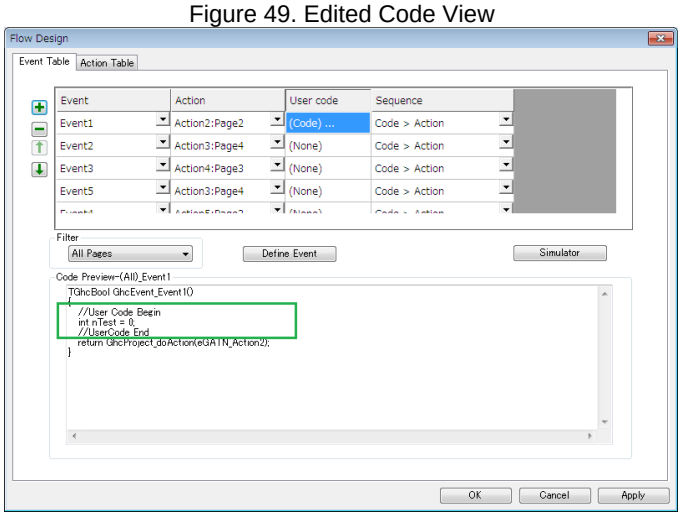
Double click “(None)”, an edit window will display. Input any standard C code as follow.

Figure 48. User’s Code Edit Window



Inputted code will be automatically added with:

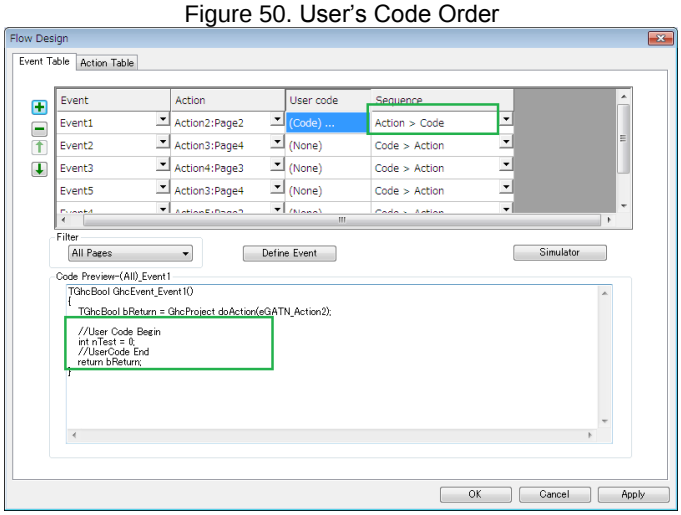
Figure 49. Edited Code View



With the “Sequence” selected “Code > Action”, the code will be implemented before action display.

If want to implement after action display, could select “Action > Code” like:

Figure 50. User's Code Order

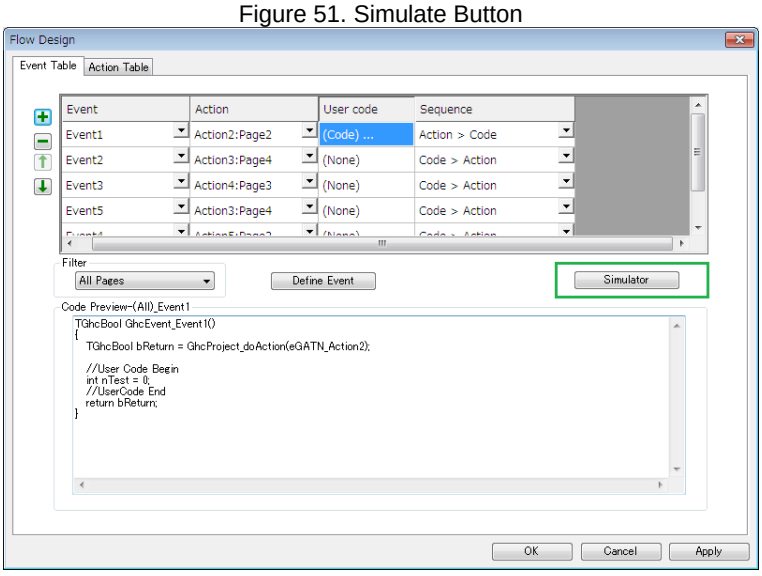


4.3 Display Simulation

After edited event table and action table, the behavior can be simulated in GAT-D.

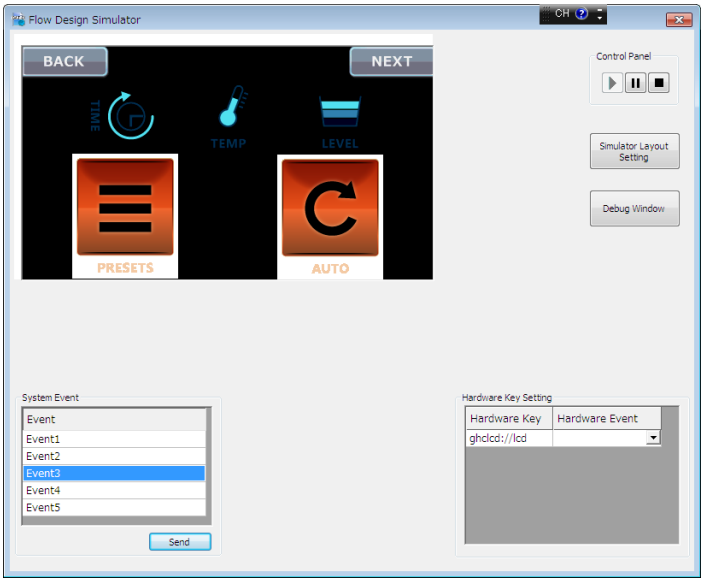
Click "Simulate" to display "Flow Design Simulator" window.

Figure 51. Simulate Button



Click "▸" will play "Startup Action", then select event click "Send", action will be played according to description of event table.

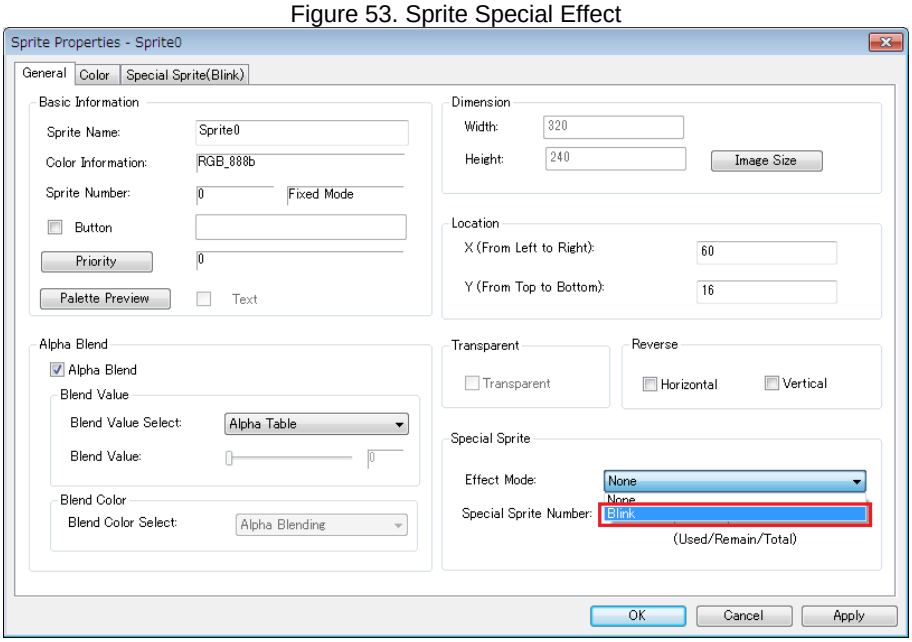
Figure 52. Simulation Window



4.4 Sprite Special Effect

GAT-D V01 support sprite special effect, follow figure show where to set it.

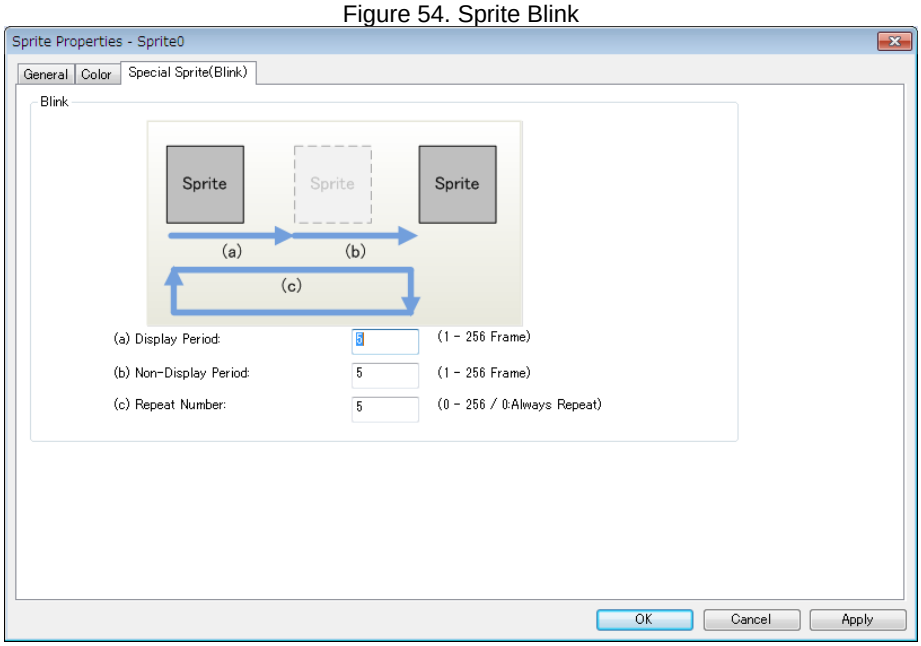
Figure 53. Sprite Special Effect



4.4.1 Blink

After set special sprite effect mode, the special effect could be set according follow figure.

Figure 54. Sprite Blink



For blink effect, it could be accomplished by edit object time line to display and none display repeatedly, but special effect set here could save memory of embedded side and save time to edit object time line, special sprite effect could do blink endlessly.

4.5 Touch Panel Resolution

S6E2DH device generally equipped LCD, also mainly equip touch panel. The most common touch panel operation is push button. To decide whether it is pushed on a button, need get the pushed coordinate and the button area. That's means manage all button area of each page. Usually button designed as a single object, and the object coordinate already managed by GAT-D. So GAT-D could give out the coordinates of a certain button.

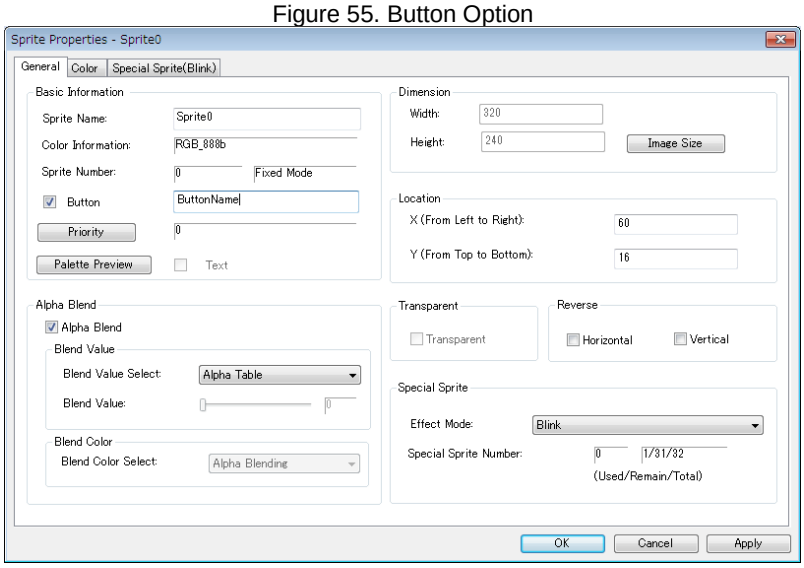
Besides, GAT-D also could give out the currently displayed page, so function software doesn't need manage a state transition of display.

If a push event happened, get current displayed page firstly, and get coordinate of button on the page then, with the pushed coordinate, whether it is a true button push could be cleared. If the push event is a display referenced, call GAT-D API: GhcProject_sendEvent_Ex() could display the actual action. See 4.1.

4.5.1 Button Name

To support LCD device which equipped touch panel, GAT-D API in embedded side supply an API GhcFunc_getButtonAreaByIndex() for user. Coordinate of button area can be got by calling the API. If an image or an object is a button and want to get its coordinate area by calling the API, in PC side need to check "Button" and input the "Button name". The "Button name" must apply to standard C expression which is used in calling API. See 5.1.7 also for details.

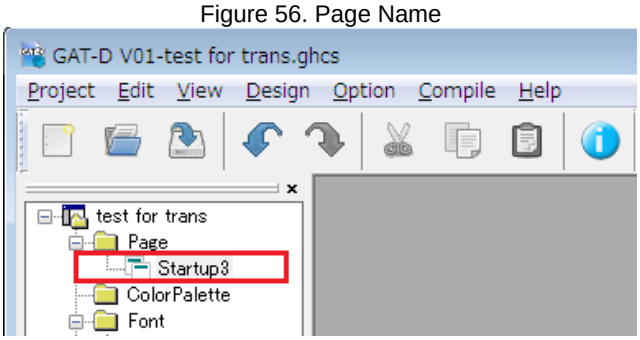
Figure 55. Button Option



4.5.2 Page Name

GAT-D API supply an API of GhcFunc_getCurrentPage () to user for getting current displayed page in embedded side. The page will return with define of page name, so name of page must apply to standard C expression. See 5.1.9 also for details.

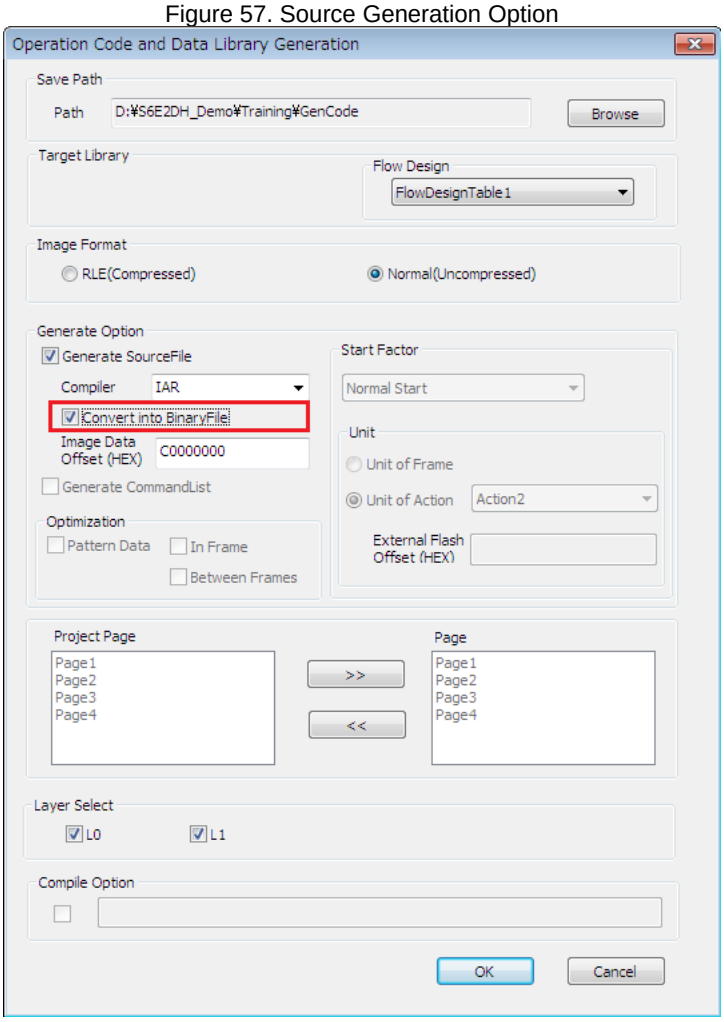
Figure 56. Page Name



4.6 Source Generate Option

S6E2DH series supply “USB Direct Programmer” to program flash image to internal flash and external flash, with this can program GAT-D’s image data to external flash directly. GAT-D can generate image data to flash image file which apply to “USB Direct Programmer”. Check “Convert into BinaryFile”, GAT-D will generate all image data to .mhx file.

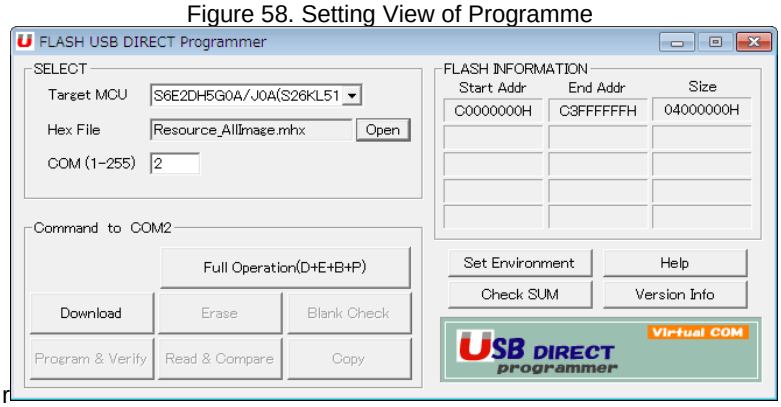
Figure 57. Source Generation Option



"Image Data Offset (Hex)" is the start address of external flash. Value is set to 0xC0000000 by default which is the start address of external flash used by Graphics Driver.

The start address must same with setting of "USB Direct Programmer" while download.

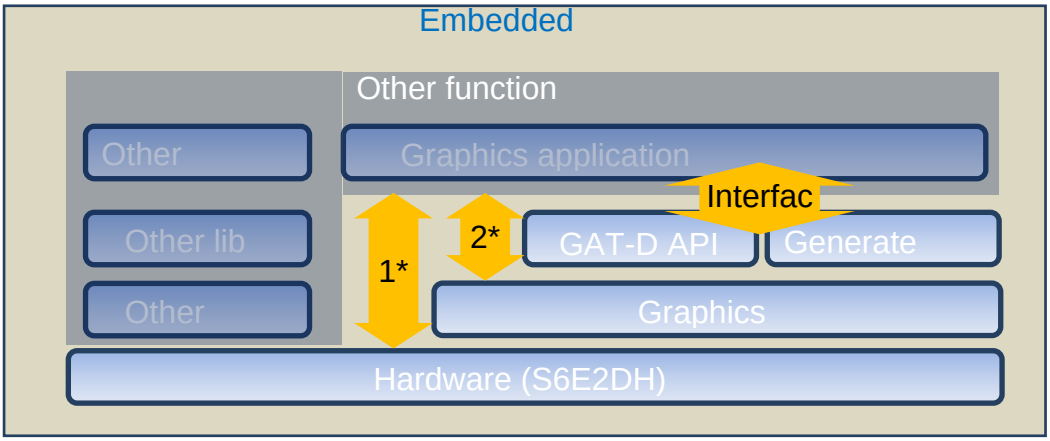
Figure 58. Setting View of Programme



5 GAT-D API

Follow figure explains where the interface is, and overview software.

Figure 59. Position of API



1* and 2*: User function source also could access Hardware or Graphics Driver, but not recommended

Follow table explains interface of GAT-D API Library and Generate source give out:

No.	Prototype	Note
1	TGhcBool GhcProject_load(TGhcU32 a_mcuMemorySize, TGhcU32 a_displayListSize)	Refer to 4.1.1
2	TGhcBool GhcIrisInitialize(void)	Refer to 4.1.2
3	TGhcBool GhcProject_setStartupAction()	Refer to 4.1.3
4	TGhcBool GhcProject_RunAction()	Refer to 4.1.4
5	TGhcBool GhcProject_StopRuningAction(void)	Refer to 4.1.5
6	void GhcProject_sendEvent_Ex(EGhcEventName eEventName)	Refer to 4.1.6
7	TGhcInteger GhcFunc_getButtonAreaByIndex(TGhcInteger a_Index, TGhcS32* a_pBtnTpLft_X, TGhcS32* a_pBtnTpLft_Y, TGhcS32* a_pBtnBtmRt_X, TGhcS32* a_pBtnBtmRt_Y)	Refer to 4.1.7
8	TGhcInteger GhcFunc_getPageOfButtonByIndex(TGhcInteger a_Index, TGhcU32* a_pPageId)	Refer to 4.1.8
9	TGhcInteger GhcFunc_getCurrentPage(TGhcU32* a_pCurrentPageId)	Refer to 4.1.9

5.1 Interface Description

5.1.1 GhcProject_load

Function	Load GAT-D project data in generated source		
Parameter	Type	Name	Description
	TGhcU32 1*	a_mcuMemorySize	Not used , set to 0
	TGhcU32	a_displayListSize	Not used , set to 0
Return	TGhcBool 2*		True: success; False: fail
Note	1*: TGhcU32 is typedef of unsigned int 2*: TGhcBool is typedef of bool		

5.1.2 GhcIrisInitialize

Function	Graphics driver initialization		
Parameter	Type	Name	Description
	Void	-	-
Return	TGhcBool		True: success; False: fail
Note			

5.1.3 GhcProject_setStartupAction

Function	Set action which will be played at startup		
Parameter	Type	Name	Description
	Void	-	-
Return	TGhcBool		True: success; False: fail
Note	The action to set is selected in GAT-D Tool. Refer to user's manual of GAT-D for details. Call GhcProject_RunAction () to display.		

5.1.4 GhcProject_RunAction

Function	Play the startup action		
Parameter	Type	Name	Description
	Void	-	-
Return	TGhcBool		True: success; False: fail
Note	Should be called after calling of GhcProject_setStartupAction ()		

5.1.5 GhcProject_StopRuningAction

Function	If there is an action playing, stop the playing.		
Parameter	Type	Name	Description
	Void	-	-
Return	TGhcBool		True: success; False: fail
Note	This calling should be arranged in another routine different with action is playing in.		

5.1.6 GhcProject_sendEvent_Ex

Function	Play the action which is referred to given event		
Parameter	Type	Name	Description
	EGhcEventName	eEventName	Given event name 1*
Return	Void		-
Note	The reference between action and event is set in Event Table of GAT-D Tool. See 4.1 Event also for details 1*: “eGEN_” should be added before event name.		

5.1.7 GhcFunc_getButtonAreaByIndex

Function	Get the area of button by given button name		
Parameter	Type	Name	Description
	TGhcInteger	a_Index	Given button name 2*
	TGhcS32* 1*	a_pBtnTpLft_X	Address to save x coordinate of top left corner
	TGhcS32*	a_pBtnTpLft_Y	Address to save y coordinate of top left corner
	TGhcS32*	a_pBtnBtmRt_X	Address to save x coordinate of bottom right corner
	TGhcS32*	a_pBtnBtmRt_Y	Address to save y coordinate of bottom right corner
Return	TGhcBool		True: success; False: fail
Note	1*: TGhcS32 is typedef of signed int. 2*: Same with inputted name while set object’s button option		

5.1.8 GhcFunc_ getPageOfButtonByIndex

Function	Get the page id which button on by given button name		
Parameter	Type	Name	Description
	TGhcInteger	a_Index	Given button name 1*
	TGhcU32	a_pPageId	Address to save id of page 2*
Return	TGhcBool		True: success; False: fail
Note	1*: Same with inputted name while set object’s button option 2*: The id is defined of page name in GAT-D Tool.		

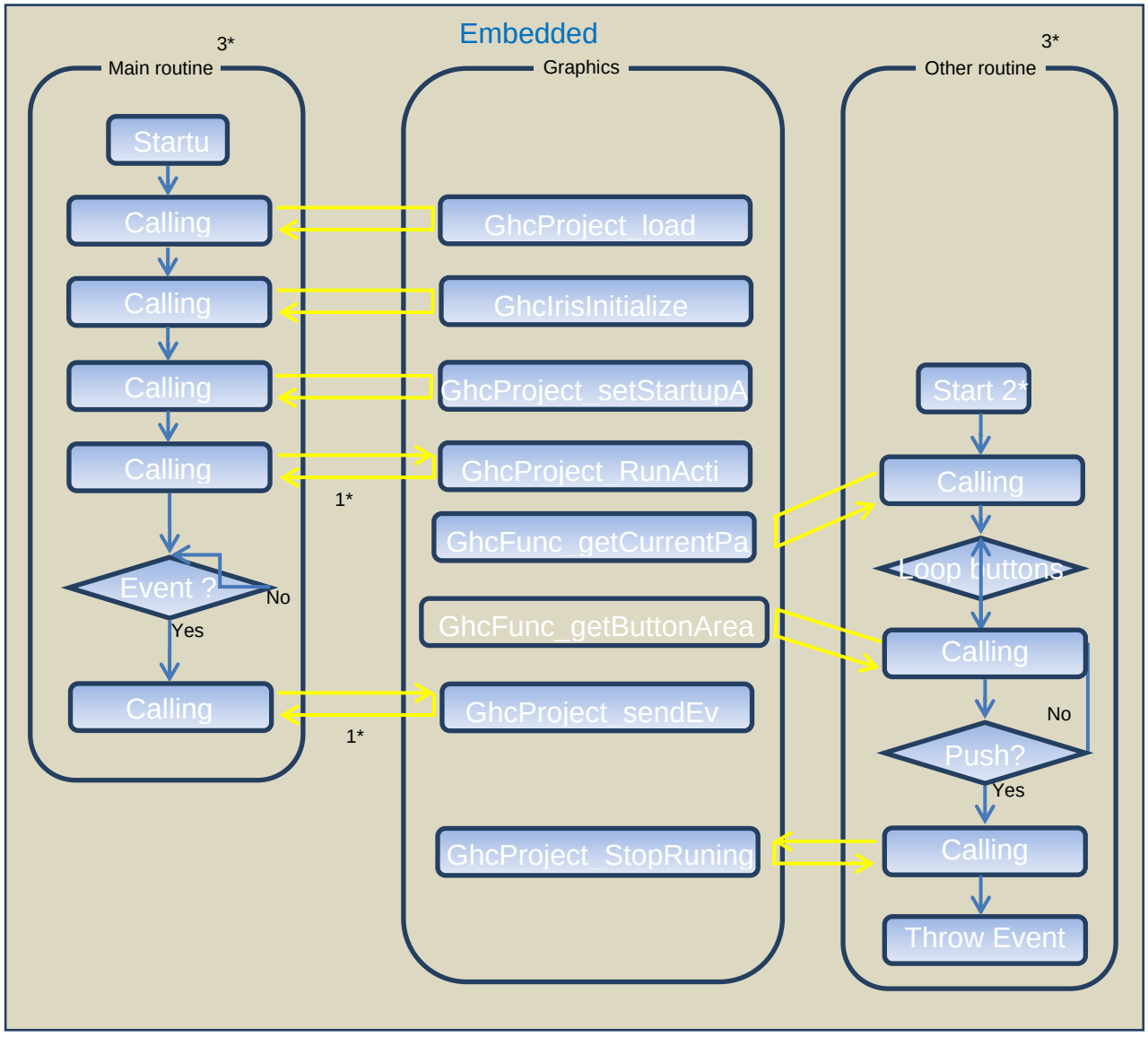
5.1.9 GhcFunc_getCurrentPage

Function	Get the current page id displayed on LCD		
Parameter	Type	Name	Description
	TGhcU32	a_pCurrentPageId	Address to save id of page 1*
Return	TGhcBool		True: success; False: fail
Note	1*: The id is defined of page name in GAT-D Tool.		

5.2 Join with Source

Follow figure explains how to use interface to join with user's source.

Figure 60. How to Join with User's Source



1*: The calling will not return until action playing complete or never return if its endless loop action, in this case, the only method is calling Ghcproject_StopRuningAction () in an interrupt routine.

2*: Assume it is operation in touch panel detection routine.

3*: These sources are implemented by GAT-D user.

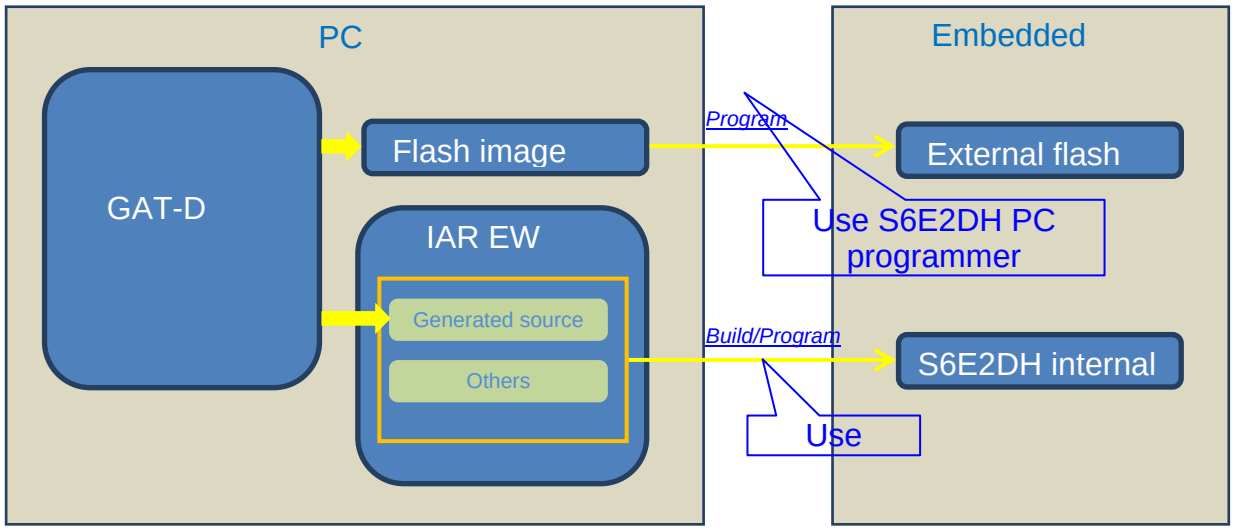
6 Program

6.1 Debugging Mode

While debugging, there are two methods to program internal and external flash if used Hyper Bus based external flash. If use HSSPI base external flash that would be Method A.

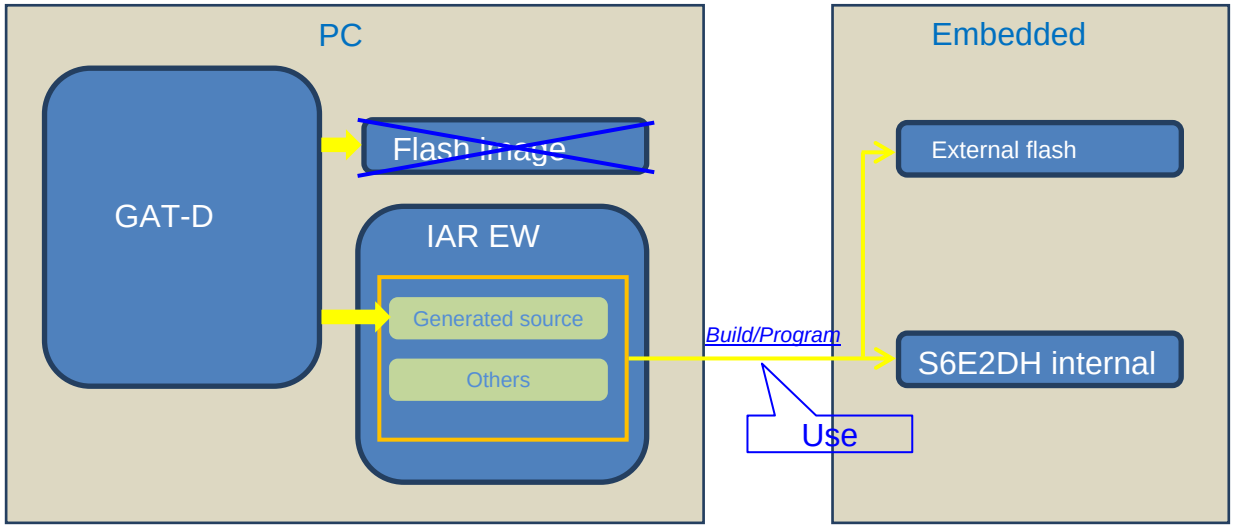
Method A:

Figure 6-1 Program Method A of Debug



Method B:

Figure 6-2 Program Method B of Debug



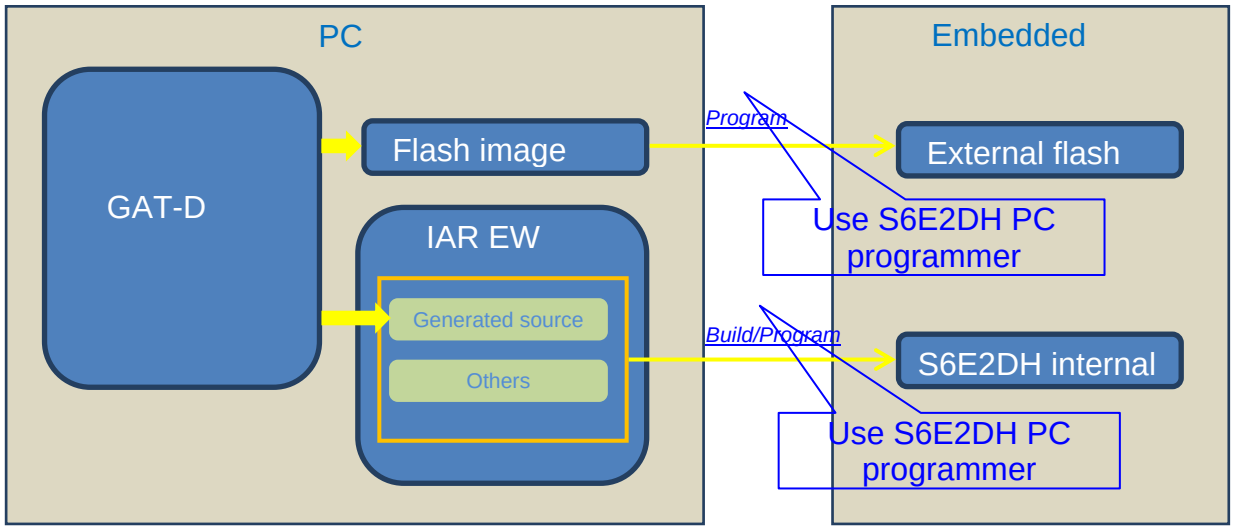
Note: Images generate to C source, also be programed to external flash.

GAT-D Tool to generate bin file or not could be selected in tool. Refer to user's manual of GAT-D for details.

6.2 Product Mode

Follow figure shows how to program product after development.

Figure 3. Program of Product



S6E2DH “USB Direct Programmer” support both internal and external (HSSPI and Hyper Bus) flash. Please refer to user’s manual of “USB Direct Programmer” for details.

Programmer of Hyper Bus based flash: USBDIRECT_V01L16c03_for_STK. Could find it in 00_s6e2dh_demeter_sw_framework\progtool of S6E2D-2DGraphicsDriver-v1.00.798-IAR.zip

7 Sample Project Instruction

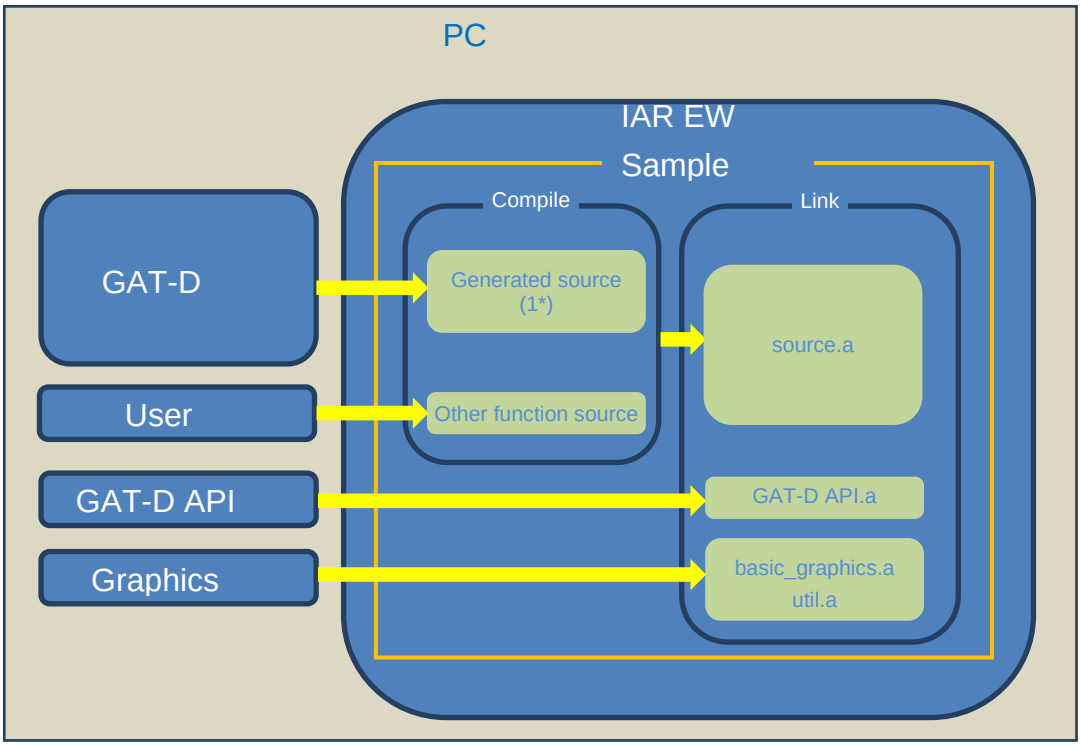
7.1 Construction

Sample project is at: 03. Sample\02. IAR\ 02. S6E2DH_Demo_Washmachine\S6E2DH_Demo_Washmachine Project.zip, but the compiling will not success unless put it to

04_sample of S6E2D-2DGraphicsDriver-v1.00.798-IAR.zip ‘s extract path.

Follow figure shows construction of sample project:

Figure 4. Sample Project Construction



1*: Include or not include image to external flash is OK

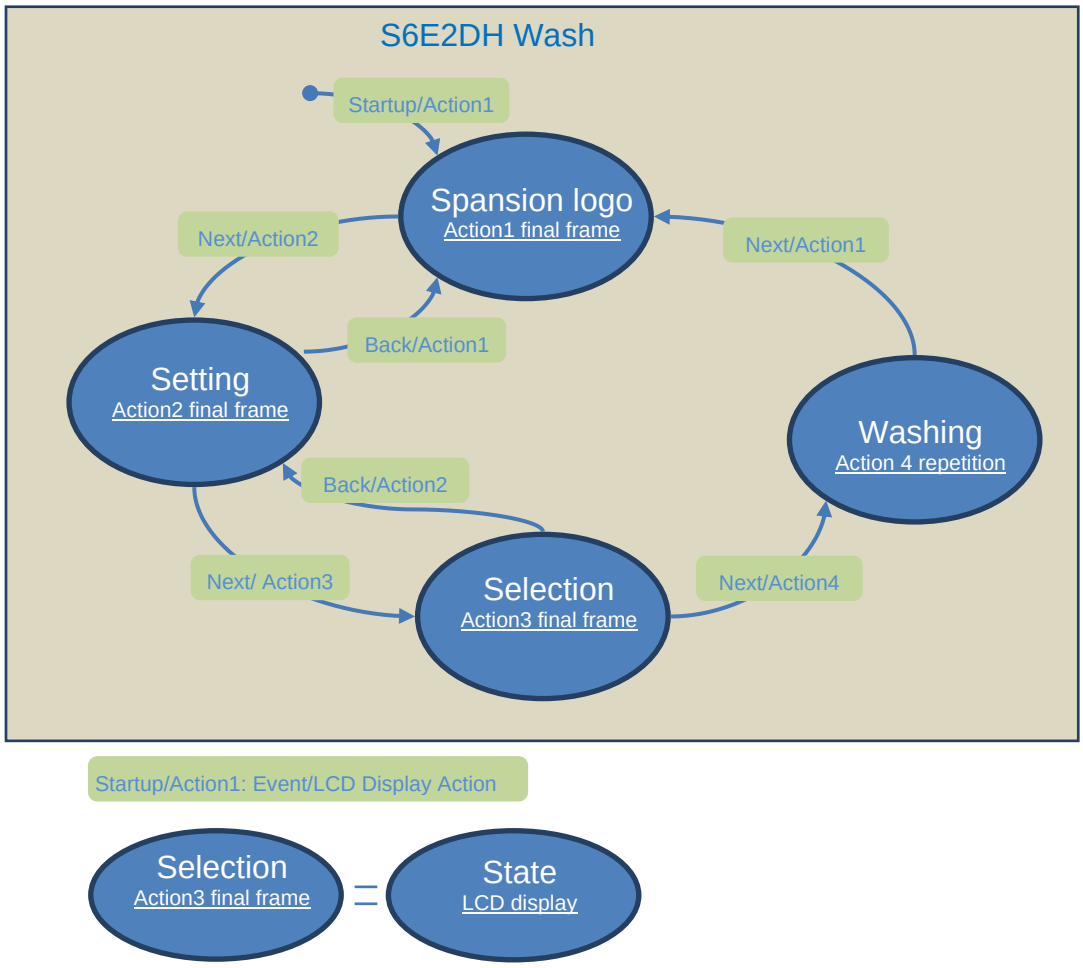
7.2 Function Overview

In path: 03. Sample\01. GAT-D Tool is a sample used GAT-D and S6E2DH_Demo_Washmachine Project.zip in path: 03. Sample\02. IAR\02. S6E2DH_Demo_Washmachine is an embedded software sample use generated source which generated by tool sample.

Note: The debugging method is using Method A. (Checked Convert into BinaryFile while source generation)

The summary function of these samples is as follows.

Figure 5. Sample Function



Spansion logo: Next event is Touch Spansion logo

Setting: Back event is press “Back” button on LCD

Next event is press “Next” button on LCD

Selection: Back event is press “Back” button on LCD

Next event is press “Next” button on LCD

Washing: Next event is press “Stop” button on LCD

Document History

Document Title: AN204388 - Getting Started for FM4 S6E2DH (GAT-D)

Document Number: 002-04388

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	_	TOAO	07/01/2015	Initial release
*A	5028487	TOAO	12/04/2015	Migrated Spansion Application Note “S6E2DH_AN709-00024” to Cypress format
*B	5873566	AESATP12	09/06/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.