

COWON S9 Flash UCI(User Creative Interface) Instruction

S9

©COWON

1. S9 Flash UCI (User Creative Interface) Overview

The S9 was developed for the purpose of supporting a full UCI. Thus, this device does not only allow change in the background or icons, but it also allows an advanced Flash developer to freely create a UI and control the device to the fullest. Although every single function of the device cannot be controlled from the UCI due to the restrictions of the embedded system and the Flash format, the S9 has been designed to allow UCI developers for the maximum developmental flexibilities. Hence, each function of the S9 is in a module to facilitate development and maintenance, with FS Commands required in controlling each function simple and concise.

Therefore, developers of the conventional Flash ActionScript can develop UCI's with little trouble simply by understanding the S9's 'input device' and its 'FS Command' process based on the conventional Flash ActionScript.

In order to provide a smooth UCI development environment, different modes of the S9 have been categorized by each function, and each mode is put into a module. Namely, S9's functions such as music, radio, videos, documents, and pictures are categorized into a single mode and established as a single Flash content. Because of this the GUI of each mode can be created independently. It is also possible, however, to accommodate processing a variety of modes within a single Flash content.

The S9 uses a large number of 'FS Commands' for the purpose of controlling a variety of functions. In a bid to organize the 'FS Commands' efficiently, the 'FS Commands' were grouped into by a similar functionality, as a tree structure.

With the grouping of modes and 'FS Commands', it is possible for anyone to join the pleasure of UCI development, even if you are not familiar with the FS Commands completely. In other words, since the music-related FS Commands are grouped together, it is possible to create a music UCI only if you are able to understand the music-related FS Commands,

2. S9 Flash Engine Specification

- 1) Flash Version : Flash Player 7
- 2) ActionScript Version : ActionScript 2.0
- 3) Display Resolution : 272 * 480
- 4) Frame Rate : 24 fps, changeable with launcher.swf
- 5) Sound : Not supported in the UI
- 6) Font : Vector Font supported

Due to the nature of the embedded system, the output is only in 'system fonts' or 'user fonts' even if fonts are assigned in the Action Script.

- 7) Rotation : Unrestricted with the exception of the Text Field. For Text Fields, only 'Single lines' support Rotation.
- 8) Memory : 15MB

3. S9 Software Structure

- 1) Description of Structure

The Flash of S9 consists of four classes, as seen in 'Figure 3-1', to accommodate a full UCI.

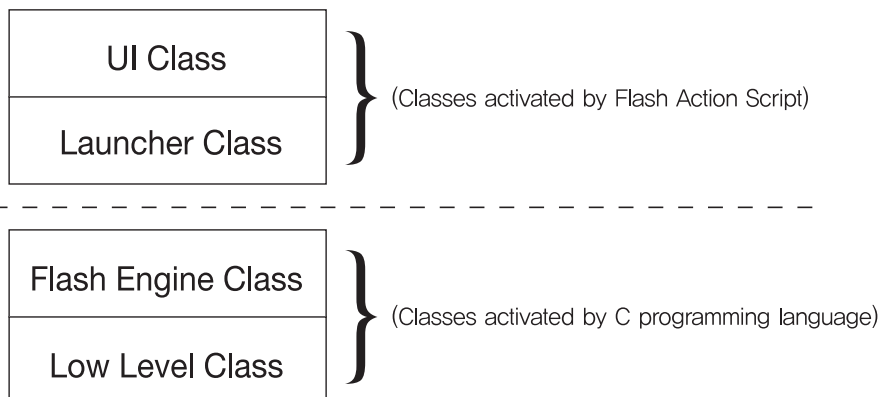


Figure 3-1

- Low Level Class : Controls Low Level devices (CPU, AMOLED, Codec etc.)
- Flash Engine Class : Activates Flash contents and relays Low Level class events to the Flash contents
- Launcher Class : Loads and unloads Flash contents, located on 'Level0' in the Flash, that applies to the UI
- UI Class : Located on Flash 'Level1' and applies to general UI

This structure applies only to the general actions. 'USB' / 'Charging Screen' / 'Firmware Upgrade', 'Start-Up Screen', 'Shutdown Screen', etc. are directly controlled by the Low Level class and the UCI is not applied. Also, when AMOLED is set to Off, only the 'Low Level class' is activated for structural reasons.

Flash contents files in charge of the Launcher and UI class included in the firmware are executed virtually. However, if they are in a device-specific folder ('System/Flash UI'), those files are executed first. To play music, for example, 'loadMovieNum("System//Flash UI//music.swf",1)' is used to load 'music.swf'. If there is a file in the applicable folder, that file is executed first and, if there is no file, the virtual 'music.swf' file within the firmware is executed.

2) Low Level Class

This class is the lowest class which controls Low Level devices such as the CPU, AMOLED, Codec, etc. Low Level devices are only controlled by the commands that are pre-defined through the FS Commands in the Flash Script. FS Commands are essentially Synchronous Commands and only a few commands are activated asynchronously. Asynchronous FS Commands are separately mentioned in the FS Command descriptions. The FS Commands that are not mentioned can be assumed to be synchronous.

3) Flash Engine Class

This class interprets the Flash contents and activates them according to the Action Script within the contents. Furthermore, it calls FS Commands according to the Action Script to control Low Level devices, or relays to the Action Script events of the Key, Touch, G-Sensor, etc. which may occur in the Low Level device.

To activate Flash contents, the Flash Engine allocates 15MB (which may differ depending on the firmware version). If the contents memory usage exceeds 15MB, external Dynamic Memory is automatically allocated and used. However, since Dynamic Memory is also used in playing music and videos, it is recommended to keep the Flash contents under 15MB if possible.

The Flash Player (in the case that the user plays back the Flash contents on the browser) is activated on a separate memory which is 5MB.

4) Launcher Class

The Launcher class is established by 'launcher.swf' and is responsible for loading and unloading Flash contents files applied to UI. This file uses Load_SWF() as the Global function, which developers use to load desired Flash contents to 'Level1' with this function.

When the Load_SWF() function is executed, 'unloadMovieNum(1)' function is called first to run UnloadMovie on the Flash contents which had been in the LoadMovie status previously. Afterwards, depending on the ItemNum value entered, 'loadMovieNum("System//Flash UI//music.swf",1)' is called and the Flash contents are run.

- Function Pattern

```
global.Load_SWF=function(ItemNum:Number,FileName:String):Void
```

- Input Value

ItemNum: The index for the Flash contents to run, and the value is as follows. The following Global variables are defined in 'launcher.swf'. However, value '5' and '9' does not activate anything, and values 'global.MODE_MAIN', 'global.MODE_MAIN2', and 'global.MODE_MAIN3' all load the same MainMenu. To run a different MainMenu, 'EtcUsrSetMainmenu' FS Command needs to be used. For example, while calling by different values 'global.Load_SWF(global.MODE_MAIN)', 'global.Load_SWF(global.MODE_MAIN2)', and 'global.Load_SWF(global.MODE_MAIN3)' still activates the same MainMenu, calling 'global.Load_SWF(global.MODE_MAIN)' after using 'EtcUsrSetMainmenu' and setting the values from 0 to 2 loads MainMenu according to the set value.

Now that Text and Flash modes are only able to execute a single file, the browser needs to run first, then, the contents are to be selected from the browser, in order to operate the contents without a problem. Therefore, loading 'text.swf' with 'global.MODE_TEXT' requires having selected 'Text' from the browser first. Flash contents do not require a separate UI and are played directly from the firmware. The purpose of 'global.MODE_FLASHBROWSER' and 'global.MODE_TEXTBROWSER' lies in designating the most recently opened document or Flash file at the time of browser initialization.

global.MODE_MUSIC : A number type variable with the value '0' and executes 'music.swf'
 global.MODE_VIDEO : A number type variable with the value '1' and executes 'movie.swf'
 global.MODE_RADIO : A number type variable with the value '2' and executes 'radio.swf'
 global.MODE_RECORD : A number type variable with the value '3' and executes 'record.swf'
 global.MODE_DMB : A number type variable with the value '4' and executes 'dmb.swf'
 global.MODE_TEXT : A number type variable with the value '6' and executes 'text.swf'
 global.MODE_PICTURE : A number type variable with the value '7' and executes 'picture.swf'
 global.MODE_DICTIONARY : A number type variable with the value '8' and executes 'powerdicrun.swf'
 global.MODE_ETC : A number type variable with the value '10' and executes files according to the input 'FileName'

 global.MODE_MAIN : A number type variable with the value '11'
 global.MODE_MAIN2 : A number type variable with the value '12'
 global.MODE_MAIN3 : A number type variable with the value '13'
 global.MODE_SETTING : A number type variable with the value '14' and executes 'setting.swf'
 global.MODE_BROWSER : A number type variable with the value '15' and executes 'browser_total.swf'
 global.MODE_FLASHBROWSER : A number type variable with the value '16' and executes 'browser_total.swf'
 global.MODE_TEXTBROWSER : A number type variable with the value '17' and executes 'browser_total.swf'

FileName : A value which is only processed when 'MODE_ETC' is input for ItemNum, it can execute Flash contents that has FileName. For example, if the input value is 'calculator.swf', 'System/Flash UI/calculator.swf' file is executed.

- Return Value
 Void

5) UI (User Interface) Class

This class contains the UI which an actual user uses to control the device. UCI is created in this class.

4. Flash ActionScript Structure

1) Description of Flash ActionScript Structure

As explained in '2-1', Flash ActionScript is designed to make the UI class activated by running the user-selected Flash UI file in the Launcher class. However, there are functions need to be under the control of "Low Level Class" such as Music UI, Video UI, GUI process, File DB, etc. If each element were used by defining the FS Command, the complexity would increase and the stability of the device would be likely to decrease. Therefore, the S9 has been designed to define each mode and use a single FS Command to switch between different modes, as in from music to video or from video to radio. With it, the 'Low Level class' control at the mode initialization process is processed with a single FS Command.

2) Description of Mode FS Command

Mode-related FS Command is 'EtcModChangeMode'. Because the 'Low Level class' which corresponds to each mode is controlled by this FS Command, the FS Command must be correctly called according to each mode at the beginning of the Flash Action Script that deals with the GUI such as 'music.swf', or 'radio.swf'. But, since the 'Flash' and 'Text' modes do not have playlists but play individual files, 'EtcModChangeMode' FS Command must be called at the beginning of the browser Action Script.

Currently, Flash Script allows the processing of only one single mode in one Flash UI file, however, multiple modes can be controlled in one Flash UI file. That is, so long as the 'EtcModChangeMode' FS Command is properly called, music or radio can be controlled while the Text Viewer is in operation. In addition, you can switch to radio while listening to Music on one screen.

Operation by each mode is as follow:

- 'Music' : Stops the mode previously in operation, configures the DB and updates the playlist with the most recently played music file.
- 'Video' : Stops the mode previously in operation, configures the DB and updates the playlist with the most recently played video file.
- 'Radio' : Stops the mode previously in operation, and configures radio streaming with the most recently configured radio frequency.
- 'Record' : Stops the mode previously in operation, and configures the DB with the most recently recorded file.
- 'Mobile TV' : Only supported in a device with a DMB support, it stops the mode previously in operation and streams the most recently viewed DMB channel.
- 'Flash' : Stops the mode previously in operation, and configures the DB with the most recently played Flash file.
- 'Text' : Operates in multi-modes. Configures the DB with the most recently viewed text file while running the previous mode as the background.
- 'Picture' : Operates in multi-modes. Configures the DB with the most recently viewed picture file while running the previous mode as the background.
- 'Dictionary' : Operates in multi-modes. Configures the electronic dictionary with the most recently viewed word while running the previous mode as the background.

3) FS Command for Each Mode

FS Commands are structured as seen in 'Table 4-3'.

Key	Common Mode, Music Mode, Video Mode, FM Radio Mode, Mobile TV Mode, Record Mode
Get Parameter	JetEffect 2.0, Display, Timer, System, Music Mode, Video Mode, Record Mode, FM Radio Mode, Mobile TV Mode, Bluetooth, Etc
Set Parameter	JetEffect 2.0, Display, Timer, System, Music Mode, Video Mode, Record Mode, FM Radio Mode, Mobile TV Mode, Bluetooth, Etc
Etc	Browser, Mode, Text, User Data

'Figure 4-1'

FS Commands related to keys are structured by each mode. To ensure the stability of the device, however, inputting the FS Command for a different mode such as 'Video' or 'Radio' while in the 'Music' mode is processed as an error. Namely, if the Key FS Command for the other modes is entered in current mode, the 'Low Level class' does not process the Key and returns an error in the ActionScript. For more detailed information on FS Commands, please refer to the separate FS Command documentation.

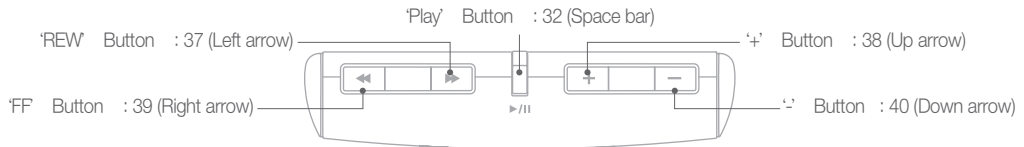
4) In creating a UI, there are parameters which need to be saved in the 'Low Level class'. The 'Flip' function in music is one such case. In such cases, the required parameters can be saved in the ActionScript in the following manner. Only the overview will be dealt with here. For more information, please refer to the FS Command documentation.

- Using the storage space allocated per mode : 'GetEtcUIConfig' and 'SetEtcUIConfig' FS Commands are available.
- Using the storage space open to ActionScript developers : 'EtcUsrGetNumber', 'EtcUsrSetNumber', 'EtcUsrGetString', and 'EtcUsrSetString' FS Commands are available. However, since the user can change the UCI frequently, simple header file is required as a unique identifier in order to use the FS Commands. Additionally, these FS Command values cannot be guaranteed if the user uses many different versions of UCI's rather than a UCI created by a single developer. For this reason, it is recommended to use storage spaces allocated for each mode.
- Using Storage Space with SharedObject

5. Input Mechanism

1) The buttons at the top.

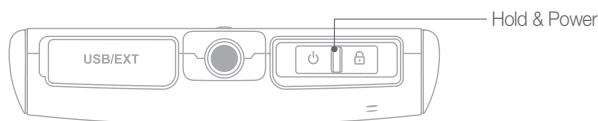
S9 has the '+', '-', 'Play', 'REW', and 'FF' buttons at the top. The buttons are designed to trigger events repeatedly – holding down the button repeats the Button Down Event every 35 ms. On the contrary, when you release the button, the Button Up Event is activated. The Button Codes related to the Action Script in the Flash engine is as follow:



2) The buttons at the bottom.

S9 has the 'Hold' and 'Power' buttons at the bottom. The following details the individual function for each button.

- 'Hold' Button : The button Code is 145(ScrLk). Because there is no need for a repeated operation, the Button Down and Button Up Events are occurred once in order upon activating Hold On and Off. If the 'Hold' Button Event is activated, the Flash developer should read the 'GetSysHoldKey' and 'GetSysCtrlHoldState' values and configure the GUI according to each status.
- 'Power' Button : No Button Event is activated since this button is only executed at the Low Level Class.



3) Touch

Since Flash does not have the function which processes Touch event, once S9 touch panel is pressed, Flash Engine recognize it as Mouse Event and activates the Event. The process for each touch action is as follow:

- Touch when AMOLED is Off: This is processed only in the Low Level class. The AMOLED status is changed to the On and no events are triggered until the touch is taken off.
- Touch when AMOLED is On: A Mouse Up Move Event is triggered first followed by a Mouse Down Event. Afterwards, a Mouse Down Move Event is triggered for each frame. The location of the touch can be seen in the Mouse X, Y value.
- Taking off the touch when AMOLED is On: A Mouse Down Move Event is triggered followed by a Mouse Up Event.

4) Gravity Sensor

The S9 contains a gravity sensor. When the unit rotates at a 0, 90, 180, or 270 degree angle, 122 Key Code (F11 Key) is triggered. Therefore, when a F11 Key Event (a Key Down Event followed by a Key Up Event) is triggered in Action Script, the status value of the gravity sensor is obtained through the 'GetEtcTASValue' FS Command and the GUI can be modified accordingly.

When Hold or AMOLED is Off, gravity sensor event (F11 key) is not triggered. Therefore, gravity sensor status needs to be processed like followings.

- When 'F11' Key is triggered: The value 'GetEtcTASValue' is read and the GUI is processed accordingly.
- When 'F12' Key is triggered: 'F12' Key is activated when AMOLED switches from Off to On, or when beginning playback. In this case, the Hold status should be read first (through the "GetSysHoldKey" FS Command) and, if the Hold is Off, please read the value of 'GetEtcTASValue' and configure the GUI accordingly. More about the 'F12' Key will be discussed in the 'Virtual Key' section.
- When 'Scroll Lock' Key is activated : While reading The 'GetSysHoldKey' FS Command, if you find the Hold having been switched from On to Off, please read the value of 'GetEtcTASValue' and configure the GUI accordingly.

5) Virtual Key

With S9, the Flash Engine does not operate when AMOLED is Off. Therefore, when AMOLED switches from Off to On, the last screen before switching to Off is displayed. And, there is a similar case when the device moves on to the next song on the music GUI, because moving to the next song is processed within the firmware and Flash does not know whether the unit has moved on to the next song. In order to resolve such phenomena, the S9 defines and uses a virtual key called the 'F12' Key for the purpose of updating the Flash GUI. For this reason, the GUI should be updated when the 'F12' Key is activated in the Action Script. The circumstances in which the 'F12' Key is activated are as follow:

- When AMOLED is switched from Off to On
- When the current music or video content finishes playing and the unit moves on to the next content
- When the current music or video content finishes playing and the unit switches to the 'Stop' status.
- When a Bluetooth Key is activated and the device operates 'Play' or 'Stop'
- When the Flash GUI needs to be updated in other firmware

6. Flash Player

For Flash playback on S9, the playback flag is set in the 'Low Level class' when selecting the Flash contents and then, as the Flash GUI Frame is finished, a separate Flash Player is generated in the Flash Engine to play the selected Flash contents. The difference between the Flash Player and Flash UI is as follows:

- 1) Display Resolution : 480 * 272
- 2) Sound : Mp3 and Stream Mp3 supported
- 3) Memory : 5MB
- 4) Play Long Key : Switch to Flash Engine after closing the Flash Player in the 'Low Level Class'
- 5) '+', '-', 'REW', 'FF' Long Key : A Key Down Event triggered in Flash GUI every 150ms

7. Loading Virtual Images (Album Artwork, JPEG Thumbnail)

On the S9 unit, virtual files can be loaded with the use of Movie Clip Load (loadMovie(), loadClip()) functions, etc.). The S9 provides a separate function not listed in the Flash Spec as it controls by using tags and folder DB, and there are sometimes JPEG files within a file.

1) Loading Album Artwork in Music Mode

- Loading with the '*.MUS' extension loads the album artwork of the file.
ex) loadClip ("0.MUS", "MC_AlbumArt")
- The number in front is the index of the file which loads the album artwork. If there are 10 files in the playlist, the index that can be loaded ranges from 0 to 9.
- The total number of files in the playlist can be viewed with the 'GetEtcTotalPLNum' FS Command.
- If a music file supports multiple album artworks, up to 6 album artworks can be loaded with 'GetAudAlbumArtTotalNum' and 'SetAudAlbumArtIndex'. If a fourth album artwork is loaded where there are only three, it is processed within the firmware to load the first album artwork.

2) Loading Preview in Video Mode

- Loading with the '*.VID' extension loads the preview of the file.
ex) loadClip ("0.VID", "MC_Preview")
- The number in front is the index of the file which loads the preview. If there are 10 files in the playlist, the index that can be loaded ranges from 0 to 9.
- The total number of files in the playlist can be viewed with the 'GetEtcTotalPLNum' FS Command.

3) Loading Story Board in Video Mode

- Loading with the '*.STR' extension loads the Story Board of the file.
ex) loadClip ("0_11.STR", "MC_Story")
- The first number in front is the index of the Story Board file, and the number behind "_" is the Story Board index of the file. For example, loading a '2_10.STR' file signifies loading the 10th Story Board in the 2nd file in the playlist.
- The total number of files in the playlist can be viewed with the 'GetEtcTotalPLNum' FS Command. There are 12 (0-11) Story Board images in a single file.

4) Loading Image Files in Picture Mode (Picture Viewer)

- Loading with the '*.PM0' and '*.PM1' extensions loads the file with the corresponding index. '*.PM0' loads regular images and '*.PM1' loads the JPEG thumbnail. JPEG thumbnail can be used to accelerate the speed of loading images.
ex) loadClip ("11.PM0", "MC_Viewer")
- The number in front is the index of the image file to be loaded. If there are 10 files in the playlist, the index that can be loaded ranges from 0 to 9.
- The total number of files in the image list can be viewed with the 'GetEtcTotalPLNum' FS Command.

5) Loading Image Files in Picture Mode (Picture Browser)

- Loading with the '*.PB0' and '*.PB1' extensions loads the file with the corresponding index. '*.PB0' loads regular images and '*.PB1' loads the JPEG thumbnail. Because the image browser loads many files, using the thumbnail is recommended.
ex) loadClip ("11.PB0", "MC_PBrowser")
- The number in front is the browser list index to be loaded. If there are 10 files in the list, the index that can be loaded ranges from 0 to 9.
- The total number of files in the browser list can be viewed with browser-related FS Commands such as 'EtcBrwSetInitialization', 'EtcBrwSetNextStage', and 'EtcBrwSetPrevStage'.