Clairitec

HMI Solution &
Graphic Products

IronGraph / StarGraph

Technical Support

Table of Contents

Chapter 1: Presentation of the training

First of all, thank you for your interest in our HMI solution!

It is the goal of this training that you master very quickly our HMI solution for all products of the “IronGraph” family.

Throughout the step-by-step creation of our HMI example, you will get to know about the technical characteristics of your HMI board, learn in an easy and efficient way the graphic commands and the way they function, and last but not least learn about the way of configuring, creating, and compiling your graphic project by making use of the software program GraphConverter.

From theory to practical terms: After each chapter, outlining in detail a precise point of the way your HMI board functions, you will have the possibility to practice the programming through numerous examples in C language. You can therefore directly put into practice your new knowledge and familiarize yourself with the programming of your HMI.

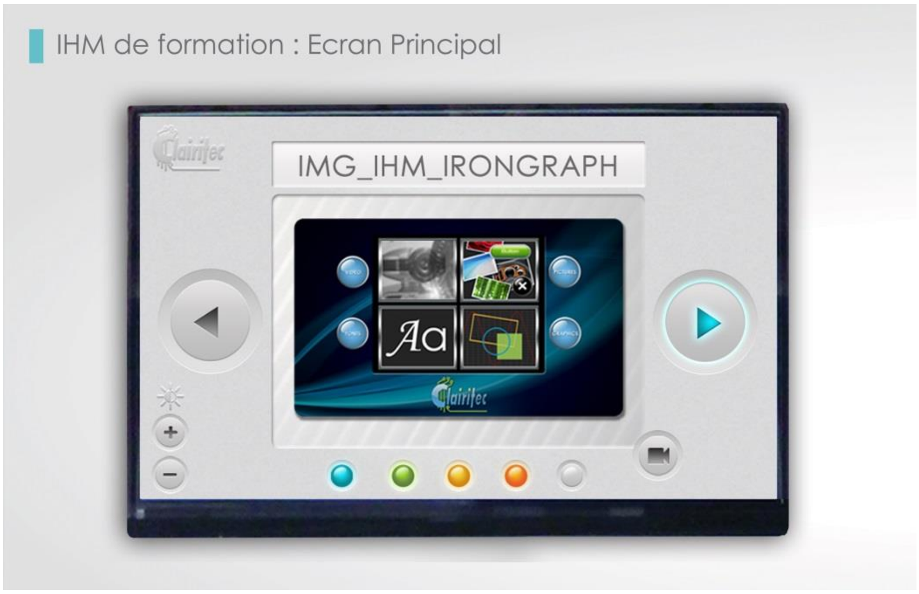
The training is based upon a StarterKit, identical to the one you use. All StarterKits of the “IronGraph” family contain an HMI board, connected to a LCD display with touch panel, as well as all necessary connecting cables in order to directly start the development of your HMI without needing to worry about the “hardware”.

A. The HMI example

The graphic project delivered with your StarterKit contains all graphic elements, in the resolution you have chosen, which make up the HMI example used for the training. Throughout the entire document we will work with the WVGA format (800 x 480 pixel, “landscape” format) for the HMI example.

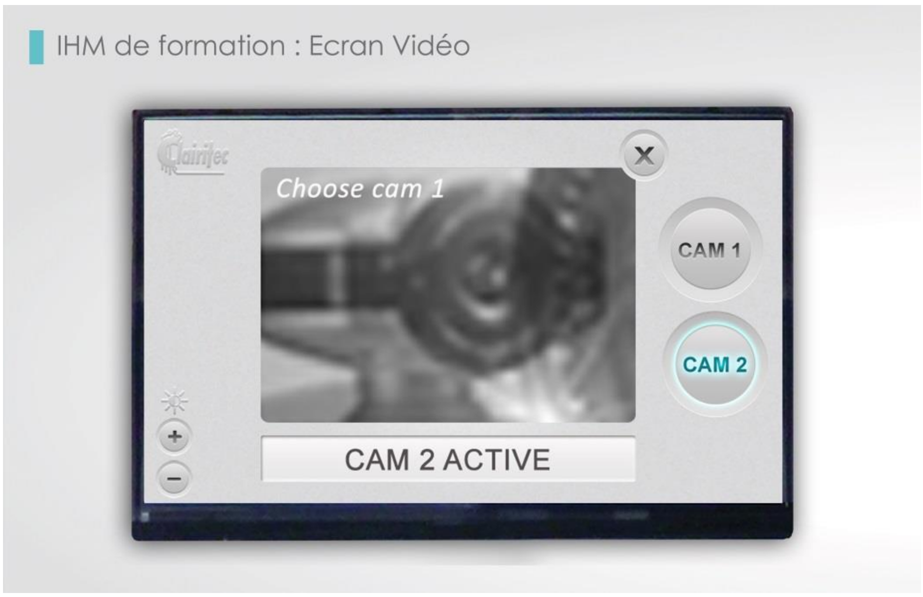
The main HMI screen of the training project displays a rectangle with 2 touch buttons “SKIP” and “SKIP_BACK” which allow to switch forward or backward the five HMI screens of the project. In the lower left corner 2 touch buttons “PLUS” and “MINUS” allow to adjust the backlight of the LCD. The HMI screen also contains a text zone to display the title of the shown interface and several colored “lights” which indicate the index of the shown interface.

(*The HMI screen example is shown on the following page)*



In the lower right corner, a “VIDEO” button is situated which allows to skip to a second HMI screen, showing the videos of the two connected cameras.

This second HMI screen contains two buttons to choose which camera should be shown in the video window, a “CLOSE” button which allows to go back to the main HMI screen, as well as the two buttons allowing to adjust the backlight.



Chapter 2: The hardware

The architecture of an HMI board of Clairitec allows for an "all-in-one" integration into your application since it contains all necessary hardware and software components to interact with your mainboard and a LCD display of your choice.

A. The IronGraph board is compatible with RS232 and CAN 2.0B protocols

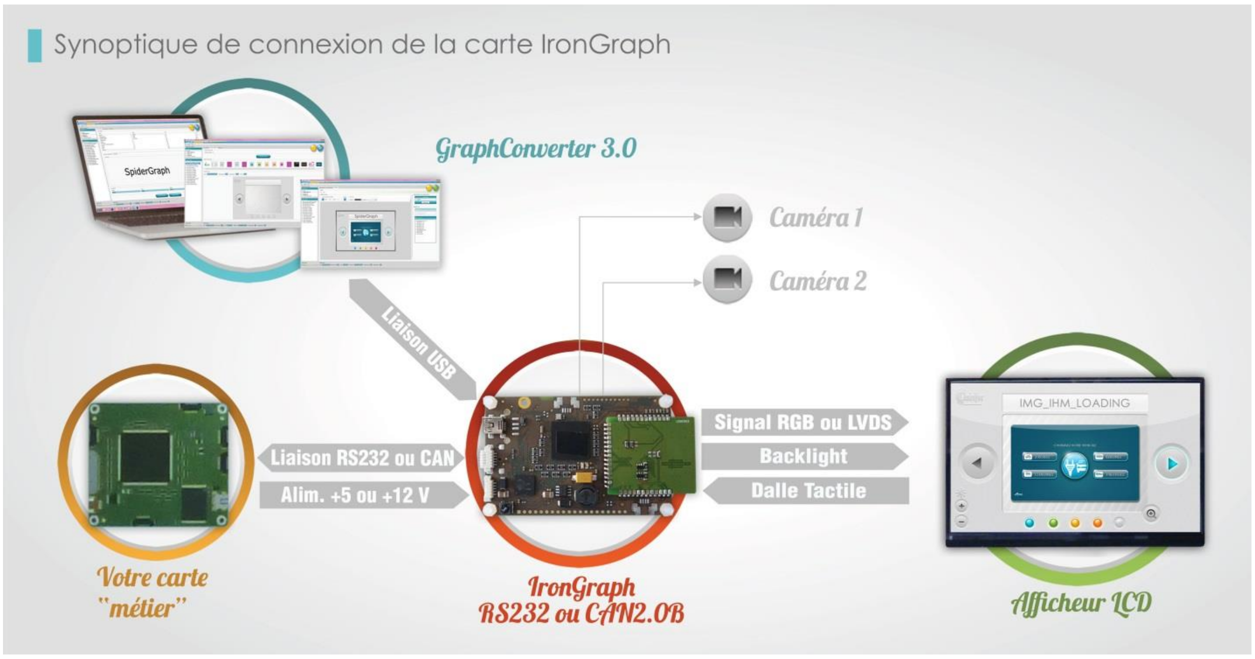
The RS232 IronGraph board contains two RS232 communication ports (1 PC and 1 TTL).
THE CAN IronGraph board contains one RS232 communication port (PC) and one CAN 2.0B port.

The two types of models are equipped with a USB2.0 port, dedicated for the upload of the graphic library and of the pre-defined HMI screens into the internal memory of the HMI board. Updates of the firmware are uploaded through the USB connection as well.

The IronGraph based hardware contains all necessary signals to steer TFT RGB displays with a maximal resolution of 800 x 480 pixel, their LED backlight and their resistive or capacitive touchscreen

No additional components need to be installed from your side, only the connectivity might need some adaption from your part.

The IronGraph board can be powered either with +5V or with +6 to +36V.



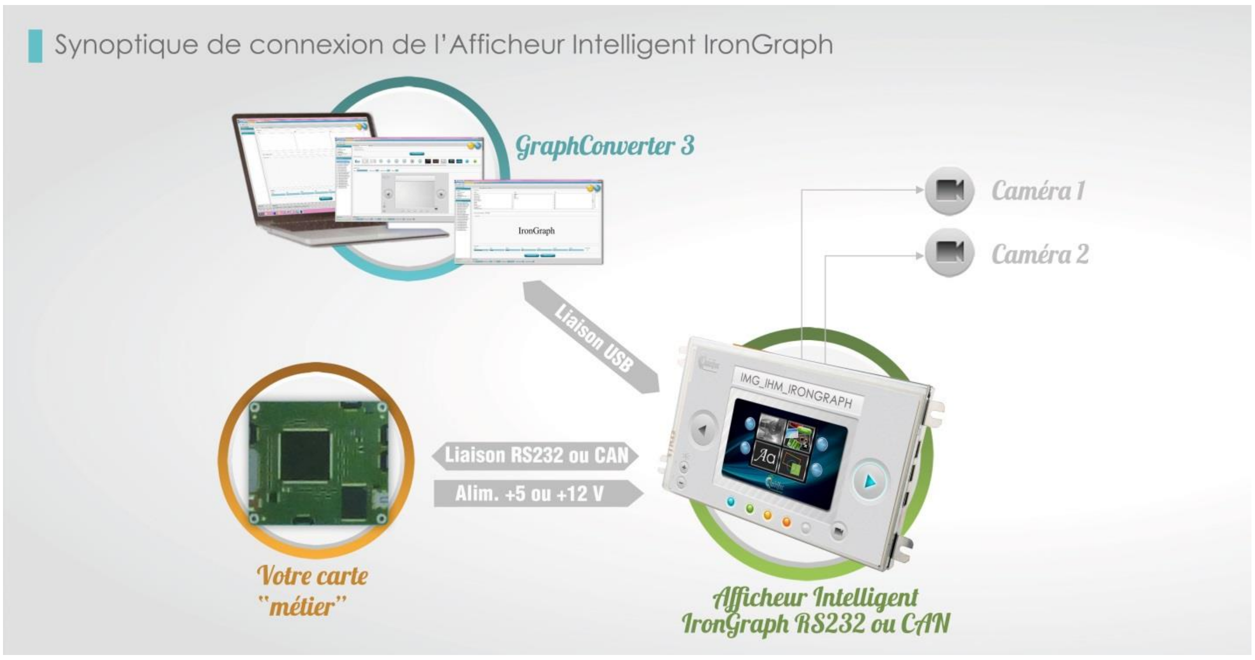
IronGraph and StarGraph support two graphical layers, one background page and one layer page with an Alpha layer that allows transparency.
These 2 graphic pages cover the entire resolution of the selected screen.

The color depth is 16 million colors formatted in RGB 8: 8: 8, the Alpha layer of the layer page is 8 bits, allowing 255 levels of transparency, (0 = 100% transparent, 255 = 0% transparent , (or completely opaque)).

IronGraph has a 3rd "graphic" layer that can display one of the 2 analog video inputs in PAL or NTSC formats.

B. The Intelligent Display IronGraph RS232 or CAN2.0B

The Intelligent Display IronGraph contains the Iron Graph HMI board (RS232 or CAN2.0B), a LCD display with touchscreen. All components are integrated into a casing with connectors to connect the Intelligent Display to your mainboard or to your computer, the latter for the communication with GraphConverter.

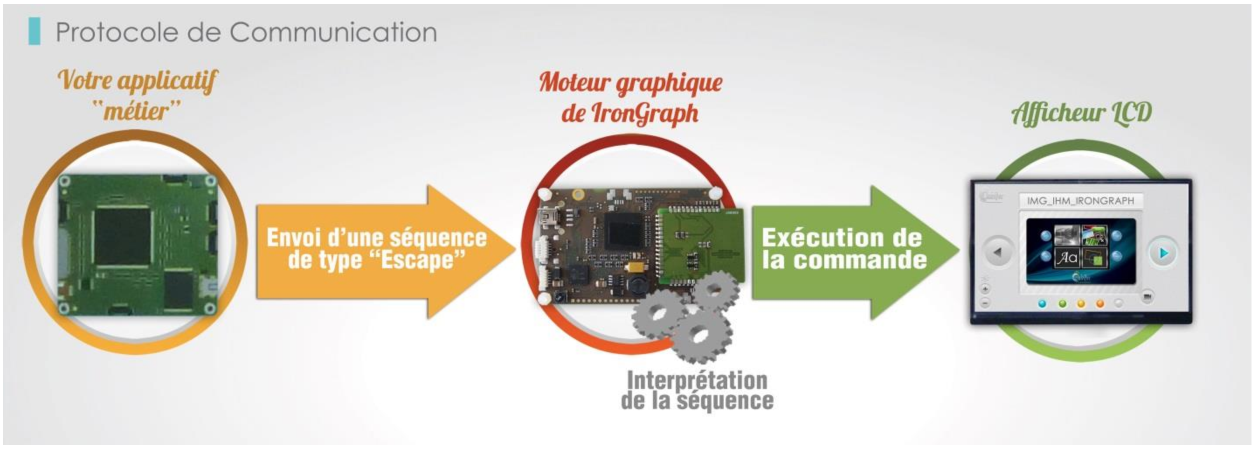

Synoptique de connexion de l'Afficheur Intelligent IronGraph
GraphConverter 3
Caméra 1
Caméra 2
Liaison USB
Liaison RS232 ou CAN
Alim. +5 ou +12 V
Votre carte "métier"
Afficheur Intelligent IronGraph RS232 ou CAN

Chapter 3: The firmware

The firmware is identical for all HMI board of the "IronGraph" family. This document takes the IronGraph board itself as an example. The specific characteristics of other boards could be outlined in a different, product-specific training at a later point in time.

A. Communication protocol for the HMI board

The communication with the HMI board takes place through "Escape" sequences, send from your mainboard to the HMI board via serial connection RS232 or CAN.


Protocole de Communication
Votre applicatif "métier"
Envoi d'une séquence de type "Escape"
Moteur graphique de IronGraph
Interprétation de la séquence
Exécution de la commande
Afficheur LCD
IMG_IHM_IRONGRAPH

An "Escape" sequence is a series of characters which is received and interpreted by IronGraph as a command. Every command is initialized by the code ASCII "ESC" (ESCape being 27 in decimal), followed by the code ASCII of the command.

The frame protocol is slightly different between a RS232 hub and a CAN 2.0B hub, but the principal function remains the same in both cases.

<u>General protocol of a RS232 command:</u>

A command is carried out through a sequence of bytes, the number being dependent on the type of command.

Byte 1	Byte 2	Byte 3	…	Byte N
Ascii Escape Code	Ascii Command Code	Code option1	,,,	Code option N

The board makes essentially use of the "ESC" character and of the automatic counting of the received characters to execute the corresponding command.

<u>General protocol of a CAN command:</u>

A command is carried out through a sequence of frames, the number being dependent on the type of command. A frame is no longer than 8 bytes.

Byte 1	Byte 2	Byte 3	…	Byte 8
Index, (of the base 0), of the CAN frame	Ascii code of the command/or 1st data of the frame index+1	Data 2	,,,	Data 7

Here, the board in CAN mode waits for the 1st byte which corresponds to the index, number of the frame. In the case of a "mono-frame" command, the index is always 0.

In the case of a command with multiple frames which are part of the same command:
At the reception of a new command, *index* = 0 => Byte 2 = number of the command.
For the following frames *index* = n+1 => byte 2 = 1st data of the following frame completing the command.
This means:
Index = 0 => Byte 1 = Code of the new command (ASCII character).
Index = n => Byte1 = Data1 of the frame number n

The number "n" allows IronGraph to verify the order of reception of the frames constituting a command.
Byte 2: ASCII character of the command if *index* = 0, 1st data of the frame "n" completing the command
Byte 3 → Byte 8: Data "arguments" of the command.

B. Management of the command cache memory

The cache memory of 8192 bytes allows to pile up the produced commands at the frequency of the host, while the graphic engine of IronGraph treats them based on the necessary time of execution.
After each treated command, the content of the respective command is automatically erased form the cache memory.

In the case of the cache memory being fully charged/overloaded, only the commands which have been completely compiled are executed while the others are left aside.
The HMI board automatically sends an alarm code when the cache memory reaches the level of 1000 bytes, as well as another alarm code when the cache is full.
In the case of a very important number of commands being produced, it is necessary to query the register of the cache memory status in order to reassure oneself that enough memory is available in the cache (command "ENQ" for IronGraph).

When the cache memory exceeds 8132-1000 bytes, the HMI board automatically sends the error codes "0x63" (critical limit reached), then "0x64" when the cache memory is full.

C. Safety of the protocol

If characters are lost (incomplete command), the HMI board does not treat this command. In any type of situation, the board will not block itself.

Attention: The protocol is only secured for favoring the speed of display. All electric precautions need to be taken into consideration for the transmission of information of the HMI boards of Clairitec. Great precaution need to be taken with regard to the supported transmission length: using reinforced cables and/or connectors or shortening the distance in case of rapid transmission speed (530000 Bauds for RS232, 500KBds for CAN). Consult the technical documentation of IronGraph for more details.

D. Error management

IronGraph receives and treats your commands, but the HMI board might also send you code to indicate an error.

RS232 protocol:

Code	**0x54:** Reset code Byte automatically send after the initialization phase to indicate to the Host that the IronGraph board is ready to receive commands.

This code also allows to warn the Host of an unexpected reset, requiring an update of the HMI (management of the HMI screens).

Format of the ERROR frame: This frame is automatically send once an error has been detected.

- Prototype:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
0xFF	Error type	ERROR CODE (MSBH)	ERROR CODE (MSBL)	ERROR CODE (LSBH)	ERROR CODE (LSBL)

- Description:

0xFF: Fixed code

Error type: Type of the error

In the current version, IronGraph only gives out RS232 errors or "Buffer Escape"

TYPEERROR_ESCAPEBUFFER = 5
TYPEERROR_rs232 = 6

Byte3 → Byte6 : Error CODE (32 bits)

- Error codes:

TYPEERROR_RS232 (in decimal)

```
RS232_OK                =0
RS232_TXE_TIMEOUT       =1 //1 character has not been send out in the appropriate time
ERROR_BREAK             =80 // Break or Overrun,
ERROR_FRAMING           =81 // Framing upon the arrival of the stop bit
ERROR_CMDE_1            =82 // ESC of the command expected but not detected
ERROR_CMDE_2            =83 // the command does not exist or two ESC one after the other
ERROR_TACTILE           =85 // Error touchscreen
BUFFER_200_BYTES        =99
BUFFER_FULL             =100
```

TYPEERROR_ESCAPEBUFFER (in decimal)

```
CMDEESC_WRONG          =1      // Faulty command (lack of the index=0?)
CMDEESC_INCOMPLETE   =2        // Index error or lack of the character ZERO for a PutString,
CMDEESC_THRESHOLD_ALAMR=3 // Only 1000 bytes of memory left in the cache memory
CMDEESC_BUFFER_FULL=4          // chache memory full
CMDEESC_BUFFERSEND_FULL=5,  // cache memory (for sending) full
CMDEESC_ERR=6                   // Escape command OK, but error in the execution of the command
                                   (error detailed in ENUM_CODEERR_GRAPHIC).
```

CAN Protocol:

Reset frame, automatically send after a reset of the HMI board.

- Prototype:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Speed	Option (*)	ID Command (MSB)	ID Command (LSB)	ID reading registers (MSB)	ID reading registers (LSB)	ID reading control (*) (MSB)	ID reading control (*) (LSB)

- Description:

Speed: Indicates the speed of communication.
Option: Indicates the programmed options (example control frame active or not).
Byte3 →Byte8 : ID of the 3 types of messages.

Error code frame automatically send after a reset of the HMI board.

- Prototype:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
ID Command (LSB)	ID Command (MSB)	0xFF	Error type	ERROR CODE (MSBH)	ERROR CODE (MSBL)	ERROR CODE (LSBH)	ERROR CODE (LSBL)

- Description:

ID Command: Allows to determine the faulty unit (in the case of several units in a network).
0xFF: Fixed code
Error type: Type of error

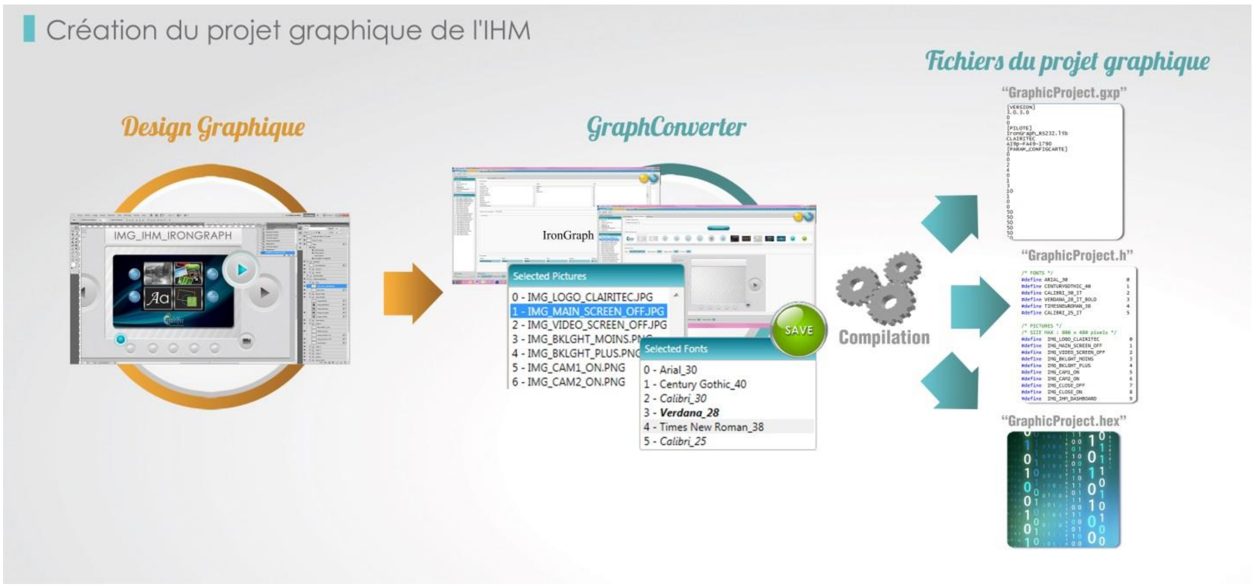
TYPEERROR_CAN = 4
TYPEERROR_ESCAPEBUFFER = 5

Byte5 →Byte8 : Error CODE (32 bits)

E. Main commands for the management of the IronGraph board and its display

ESC 'V'	Put the HMI board into standby mode
ESC 'W'	Software reset of the HMI board and the display
ESC 'e'	Activate or deactivate the management of the "status" registers
ENQ	Read the status of the cache memory for the commands
ESC 'K'	Adjust the brightness level of the display
ESC 'k'	Turn-on or turn-off the backlight

Chapter 4: The graphic project



Selected Pictures

0 - IMG_LOGO_CLAIRITEC.JPG
1 - IMG_MAIN_SCREEN_OFF.JPG
2 - IMG_VIDEO_SCREEN_OFF.JPG
3 - IMG_BKLGHT_MOINS.PNG
4 - IMG_BKLGHT_PLUS.PNG
5 - IMG_CAM1_ON.PNG
6 - IMG_CAM2_ON.PNG

Selected Fonts

0 - Arial_30
1 - Century Gothic_40
2 - *Calibri_30*
3 - ***Verdana_28***
4 - Times New Roman_38
5 - *Calibri_25*

A. The graphic design

Every HMI project begins with the specification of the navigation elements and the sequence of HMI screens as well as their organization.

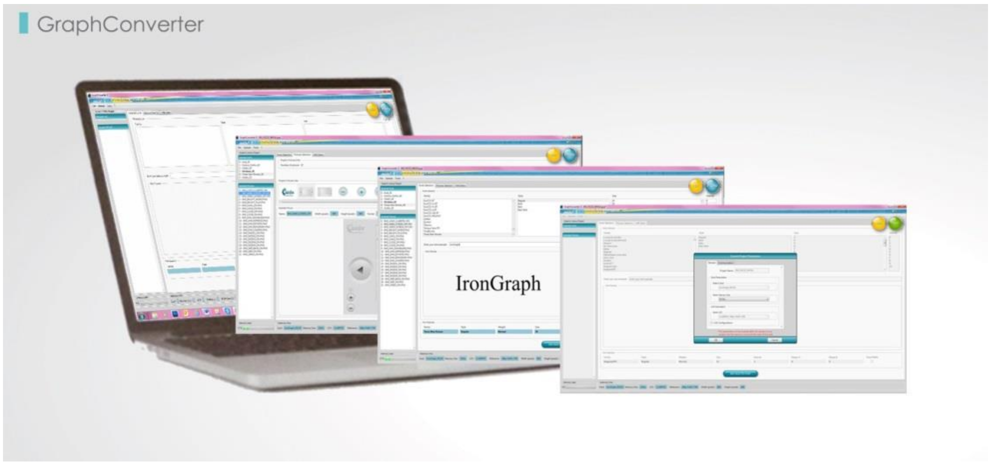
After the usage has been defined, the next step is to create the different graphic elements of your interface by using a software program for graphic design.

This way, all visual elements, from the background of the HMI screens to the navigation elements such as icons and buttons, also taking into account the choice of colors and text fonts, are created and reviewed by a graphic designer. Afterwards, the elements are selected, compiled, and uploaded into the internal memory of the Irongraph HMI board by using the PC program GraphConverter.

What we call a “graphic project” is the entity of necessary files for IronGraph to display the graphic elements (images and texts) on the display you have chosen. We will now see how to generate these files by creating a project and a graphic library with GraphConverter.

B. The software program GraphConverter

GraphConverter is the software program delivered with our HMI boards. It allows you to configure your project and to create your graphic library and set the menus of your HMI.

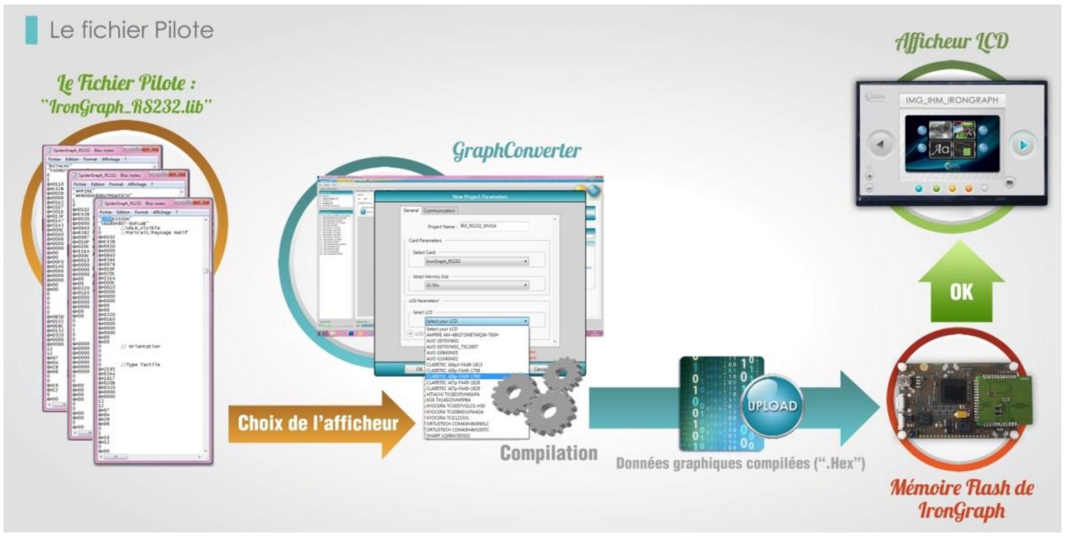


1. Project settings

The software program GraphConverter is compatible with all HMI boards of Clairitec. However, it needs to know the specific characteristics of the board used, such as the communication protocol, speed of transmission, size of the internal memory, color depth, compatible picture formats, type of display used (with or without touchscreen, QVGA, VGA, WVGA, etc…resolution), the standard settings for the calibration of the touchscreen, etc.

All this information is contained in the "driver" files. The most standard ones are already installed when installing GraphConverter. Some are modifiable in GraphConverter. There is a driver file for each type of HMI board (for example, the driver "IronGraph_RS232.lib"contains all transmission speeds possible with this HMI board).

This files is therefore very important because it allows you, once your project has been configured in GraphConverter, to program IronGraph by uploading the binary file of the graphic data and configuration data (HEX file), generated by GraphConverter.


Le fichier Pilote
Le Fichier Pilote : "IronGraph_RS232.lib"
GraphConverter
Afficheur LCD
OK
Choix de l'afficheur
Compilation
UPLOAD
Données graphiques compilées (".Hex")
Mémoire Flash de IronGraph

Advantages of the driver file:

- The client configures his project by making use of the proposed data (flash memory size, communication type with the host (RS 232 or CAN), transmission speed of the communication, type of display, etc.).

- The driver file adapts itself : its structure allows easy updates without a modification of the code of GraphConverter (for example adding a new display reference). Moreover, it always stays compatible will former, current, and future HMI boards of Clairitec.

2. Creation of the graphic library

GraphConverter is not only used to configure your project, but also to create your graphic library which is making up your HMI. This graphic library is then uploaded into the internal memory of the HMI board.

With only a few mouse clicks you create your graphic library by selecting on your computer the text fonts and images you would like to use for your HMI.

The text font are encoded by GraphConverter to be compatible with the HMI boards. You can work with any type of text font as long as it is in the TrueType format (.TTF).

For the IronGraph HMI board, pictures can be in .JPEG, .PNG, or .BMP format. The maximum resolution of the images depends on the used display.

Note: Consult the user manual of GraphConverter for more detailed information about the creation of your graphic library with GraphConverter.

3. The graphic project files

All necessary files to configure your HMI board and manage your graphic library are automatically generated by GraphConverter upon saving your graphic project.

Your graphic project is composed out of a data file (.gxp), a compiled data file (.hex) and a header file (.h) for your application in C language. A folder “Pictures”, in which all pictures making up your graphic library are stored, is also created upon saving your graphic project.

- The GXP file ("*project_name_GraphConverter.gxp*") :

The GXP file is the file which you open with GraphConverter. It is composed out of three groups, defining the entity of the graphic part of your HMI project.

The two main groups are the text fonts and images which you have chosen to use in your HMI. Each of these graphic objects is attributed a number (which is also its position number in the memory of IronGraph) and a name (a system name for the text fonts and the fie name for the images).

(*Note : Consult the documentation of your HMI board for more details about the organization of the text fonts and images in the allocated internal memory of the HMI board)*

The third group of data saved in the GXP file concerns the configuration of the hardware: display orientation, reading mode of the touchscreen, transmission speed with the host, etc.

- The binary HEX file ("*project_name_GraphConverter.hex*"):

The HEX file is the binary file which contains all the data from the GXP file in compiled form. It is this file which is uploaded into the internal memory of the HMI board.

- The H file ("*project_name_GraphConverter.h*"):

The H file (header file) is a file which is generated by GraphConverter and which you include into your application in C language[(1)]. It defines the names and numbers of the text fonts and images which make up your HMI. The numbers, attributed by GraphConverter, are used in the commands to display texts or images.

In order to explore GraphConverter, we invite you to open the GXP file :

- *From the provided USB stick, copy the folder " Exemples\IronGraph\RS232\WVGA\Projet GraphConverter\IRG_RS232_WVGA" to your PC[(1)],*
- *Open GraphConverter and click on "Open Project…"*
- *Select the file "IRG_RS232_WVGA.gxp[(1)]" in the folder " IRG_RS232_WVGA[(1)]" which you have just copied, and click on "Open".*
- *Once the project has been opened, click on the tabs "Fonts Selection" et "Pictures Selection" to open the interface for the text font and picture selection and to see the entity of graphic objects which make up your training project HMI.*
- *For a quick view of the configuration of the training select "Project Parameters" in the "Tools" menu of GraphConverter.*

NB : If you wish, you can also already open the H file[(2)] generated by GraphConverter in order to get to know its structure. In any case, we will get back to this file in the next chapter.

- *Open the folder "GraphConverter/IRG_RS232_WVGA[(1)]" on your PC.*
- *Open the file "IRG_RS232_WVGA.h[(1)]" with any type of text editor.*

Notes : (1) *For the CAN version, replace « RS232 » by « CAN » in the name of the folders, files and repertories.*

(2) *GraphConverter also contains an HMI Editor. You use in your application in C language the header file .h, generated by this Editor. Chapitre 12 : L'éditeur d'IHM de GraphConverter explains the HMI Editor.*

4. Upload of the graphic project into the internal memory of IronGraph

- Once you have created and saved your graphic library in GraphConverter, you have to upload the project configuration and data into the memory of IronGraph before starting to program.

Make sure that your StarterKit has been well connected to your PC by using an USB cable, and that you have well uploaded the graphic project associated to your HMI into the internal memory of IronGraph by the use of GraphConverter :

- Open the file "*IRG_RS232_WVGA.gxp[(1)]*" with GraphConverter,
- Click on the blue button "Upload" on the top right corner in the GraphConverter menu. A window opens in which you can select the binary file to be uploaded.
- Start the upload. The project configuration and graphic data will be uploaded into the internal memory of the HMI board.
- Once the upload has been finished, you can close GraphConverter.

Note : (1) *For the CAN version, replace « RS232 » by « CAN » in the name of the folders, files and repertories.*

Now that you have configured your HMI board and all the necessary data has been uploaded into it, we will see how to use this data and information to display and program your HMI.

Chapter 5: Programming

The programming indicates to IronGraph what to display depending upon user interfaction or information send from the main application.

The programming of the code to steer your HMI is done in your main application. There is thus no special competence to acquire to program your HMI and you can directly start to program in the environment to which you are used to.

For our training, we have chosen program our training project HMI in Microsoft Visual Studio Express 2010 (*which can be downloaded here*). This program is used to program your HMI and simulates your main application.
We will now open the training project:

- Copy the folder "*Exemples\IronGraph\RS232\WVGA\CodeSource\IRG_TrainingProject* " from the USB stick of Clairitec to your PC.
- Open Visual C++ and select "Open a project…"
- Select "*IRG_TrainingProject.sln*" in the folder which you have just copied and then click on "Open".

Upon execution, our program opens a window which allows you to chose the COM port upon which the USB/RS232 adapter cable[(2)] to your HMI board is connected. Click on the button "START IHM" and see how the main HMI screen of the training project HMI is displayed on the LCD display of your StarterKit.

We invite you to discover the training project HMI by testing its different buttons.

Notes : [(1)] *For the CAN version, replace « RS232 » by « CAN » in the name of the folders, files and repertories.*
[(2)]*For the CAN version, us the USB/CAN adapter.*

We will now explain you which are the necessary files for the programming of your HMI.

A. The API (Application Programming Interface)

As we have seen before, the communication with the HMI board is realized through the sending of “Escape” sequences from your main application through a serial RS232 connection.

1. The "ESCAPE" commands

We have developed a set of 28 “ESCAPE” commands which allow you to build and steer your future HMI from your main application, starting from the display of images and texts and going to the management of the touchscreen and the backlight of the display.

2. The functions in C language

Today, most of the main applications are programmed in C or C++. Therefore, we have developed a library of C functions based upon the “ESCAPE” commands in order to facilitate the programming of your HMI application. Each C function corresponds to an “ESCAPE” command.

In total, 28 functions are provided which allow you to communicate with the graphic engine of IronGraph and to steer your HMI from your application:

- The functions for the general management of the HMI board
- The functions for the management of the graphic layers
- The functions for the management of the touchscreen
- The functions for the color management
- The functions for the drawing of geometric forms

- The functions for the display of images
- The functions for the usage of text fonts

On the USB stick, delivered with the StarterKit, you will find the directory "Exemples\IronGraph\API_Libraries\IronGraph_StarGraph\IRG_Library_Files" in which you find all the commands in C language.

The files contain the declarations of constants which define the options and arguments of the commands:

- "IRG_GraphicEngine_Library.h ".
- "IRG_RS232_GraphicEngine_Library .c ", (for a usage in RS232 mode)
- "IRG_CAN_GraphicEngine_Library .c " , (for a usage in CAN mode)

Before starting to program your HMI, you will first have to integrate these files into your application.
You will also have to integrated the files listed below into your application, since they contain the declaration of RS232 or CAN functions, necessary for your hardware drivers (these files are to be adapted according to your mainboard).

« \IRG_Library_Files\HardWareLayer_Files\UART_RS232_Driver\"

- "UART_Driver_RS232.c"
- "UART_Driver_RS232.h"

"\IRG_Library_Files\HardWareLayer_Files\CAN_Driver\"

- "Driver_CAN.c"
- "Driver_CAN.h"

We now invite you to take a look at the content of these 4 files for the API (application programming interface) in your Visual C++. Project.

B. The main program

The main program are the functions which steer your HMI application. As we have seen above, this program contains all of the functions which allow you to indicate to IronGraph what to display in response to touchscreen events or in response to information input from your main application.

Most of the functions of the programming library, in particular the functions to display images and texts, use as parameters the reference data, compiled and uploaded into the internal memory of IronGraph by GraphConverter (HEX file).

In order to make the access to this reference data easier for you, GraphConverter creates automatically header files which contain the declaration of the number and name of the images and text fonts of the graphic project, as well as their coordinates (position) on the interface.

By integrating these files into your project, you automatically have all the references to the images and text fonts at your disposal.

The main program of the training HMI has as a double objective to simulate your main application and to give you a programming example of an HMI. You find the source code of this program in the folder:

- " CodeSource\IRG_TrainingProject / IRG_TrainingProject_Main.cpp".

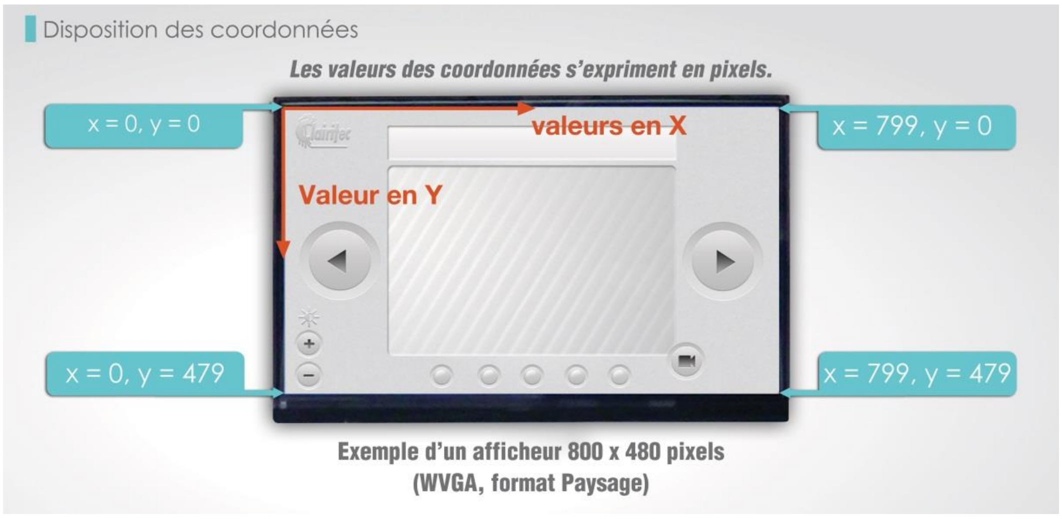
There you find the declaration of the necessary header files as well as the code of the different functions, steering the HMI and using the commands of the API.

Now that you have seen how to integrate the files, necessary for the programming of your HMI, into your application, we will now see each function of the API in more detail. You will learn how to work with the two graphic layers, how to define and activate the touchscreen zones, and how to dynamically display images and text.

Chapter 6: The coordinates of the display

A. The general principle

The coordinates on the display are given in pixel and are determined in relation to the left-upper coin of the display, whose coordinates are X=0 and Y=0.


Disposition des coordonnées
Les valeurs des coordonnées s'expriment en pixels.
x = 0, y = 0
valeurs en X
x = 799, y = 0
Valeur en Y
x = 0, y = 479
x = 799, y = 479
Exemple d'un afficheur 800 x 480 pixels
(WVGA, format Paysage)

The display of our training HMI has 800x600 pixel (WVGA), in landscape format. The value of the coordinates therefore varies from 0 to 799 (from left to right), and from 0 to 479 (from top to bottom).

B. "Landscape" or " Portrait" format

Each LCD display has a certain display format for which it has been designed, either landscape or portrait format. The visual characteristics of the display are optimized for this format by the manufacturer (vision angle, visible surface in relation to the mechanical surface,…). This format is called the “native format” of the display.

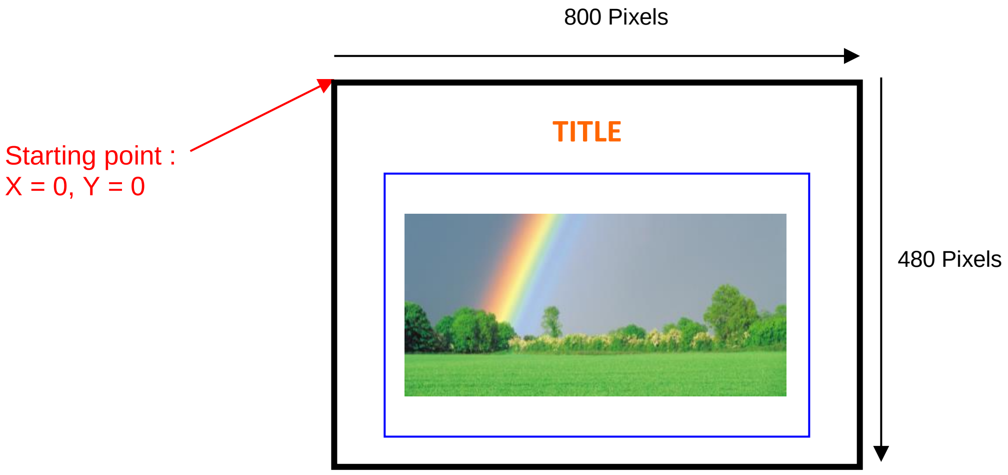
The HMI boards of Clairitec allow to modify this format: you can display in “portrait” format on a “landscape” display and the other way around.

When you modify the format, texts as well as all coordinates (of texts, images, primitive drawings, touch zones) are automatically adapted by GraphConverter.

When the display format is modified, the HMI board considers the display to be turned 90° clockwise (90° to the right).

Example of a WVGA display with native landscape format, modified to display in portrait format.

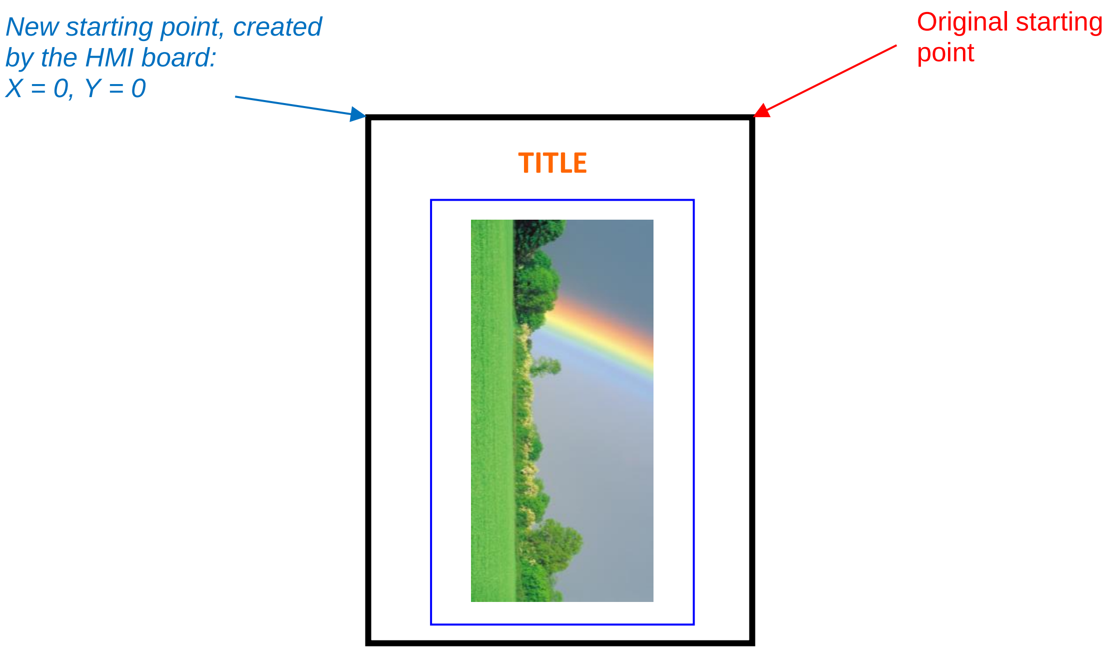
Native display format :



WVGA, native LANDSCAPE format

Format modified to display in portrait mode:

Rotation +90° "clockwise" through the HMI board:



WVGA, switched to PORTRAIT format

Chapter 7: Management of the 2 graphic layers

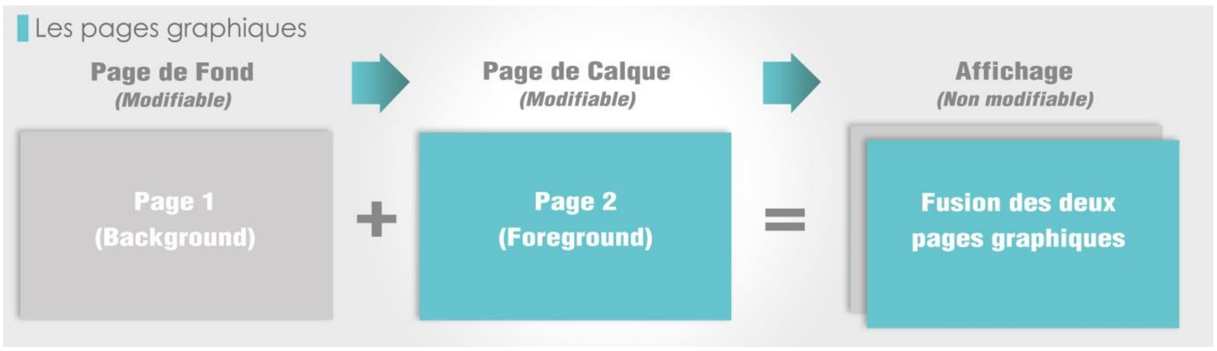
A. Definition of the graphic layers

A graphic layer is like a page on which you place the graphic elements composing your HMI.

For IronGraph, you have 2 graphic layers at disposition: a background layer and a foreground layer. On these two layers you can place images, display texts or draw geometric forms. The dimensions of these two graphic layers is depending on the resolution of the display. For example, when displaying on a WVGA screen, the dimension of the layers will be 800x600 pixel.

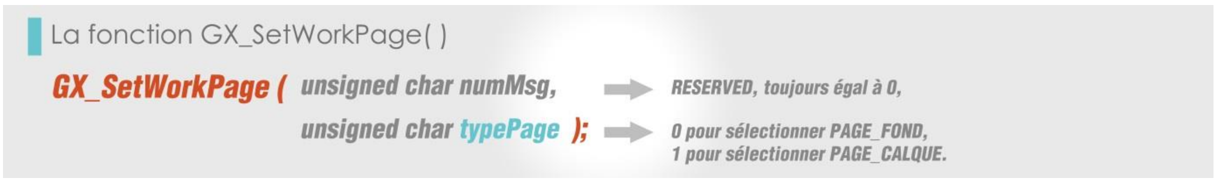
Each of these two layers is available to be "written" upon, but only the foreground layer manages transparence. You can display the two layers individually and chose which one to display over the other (you could display the background layer over the foreground layer). The background layer is called this way because it does not manage transparence, it is always opaque (non-transparent). If you display the background over the foreground layer (with the command GX_SetViewPage), the foreground layer will be completely hidden.
This feature could be useful for hiding the foreground layer, with a lot of graphic elements placed upon it, while it is loading.

When the foreground layer is displayed over the background layer, the background layer can be seen through all transparent surfaces. In this case, the graphic engine superposes and unites the two graphic layers, as shown in the schema below:



B. Selection of a graphic layer

In order to use a graphic layer and place graphic elements upon it, you will first have to select it by using the command GX_SetWorkPage ( ).



La fonction GX_SetWorkPage( )

```
GX_SetWorkPage ( unsigned char numMsg,        →  RESERVED, toujours égal à 0,
                 unsigned char typePage );    →  0 pour sélectionner PAGE_FOND,
                                                 1 pour sélectionner PAGE_CALQUE.
```

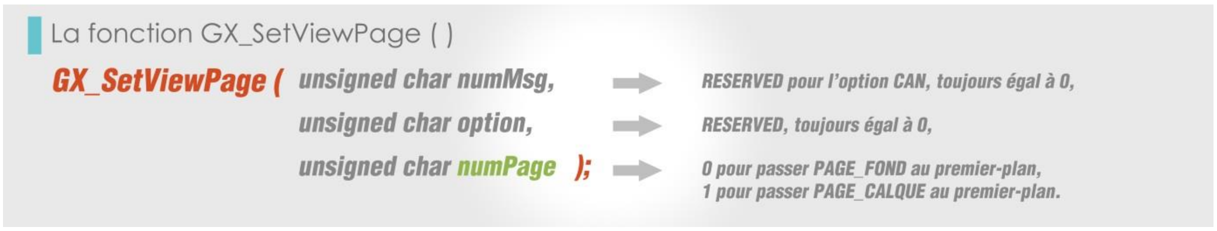
The page

The graphic layer selected with this command becomes the one used for writing. All the display commands (display images, texts, etc.) which you send are applied to this layer. The selected layer is therefore the active one, the one used for "writing".

Note : GX_SetWorkPage () selects the active layer on which you would like to display elements. This command does not modify the order of the graphic layers (which layer is displayed over the other), defined with the command GX_SetViewPage ( ). No matter which graphic layer is displayed over the other, the one selected for "writing" is the one which stays activated.

C. Displaying a graphic layer

The function "GX_SetViewPage" allows to display either the background layer or the foreground layer, merged with the background one:

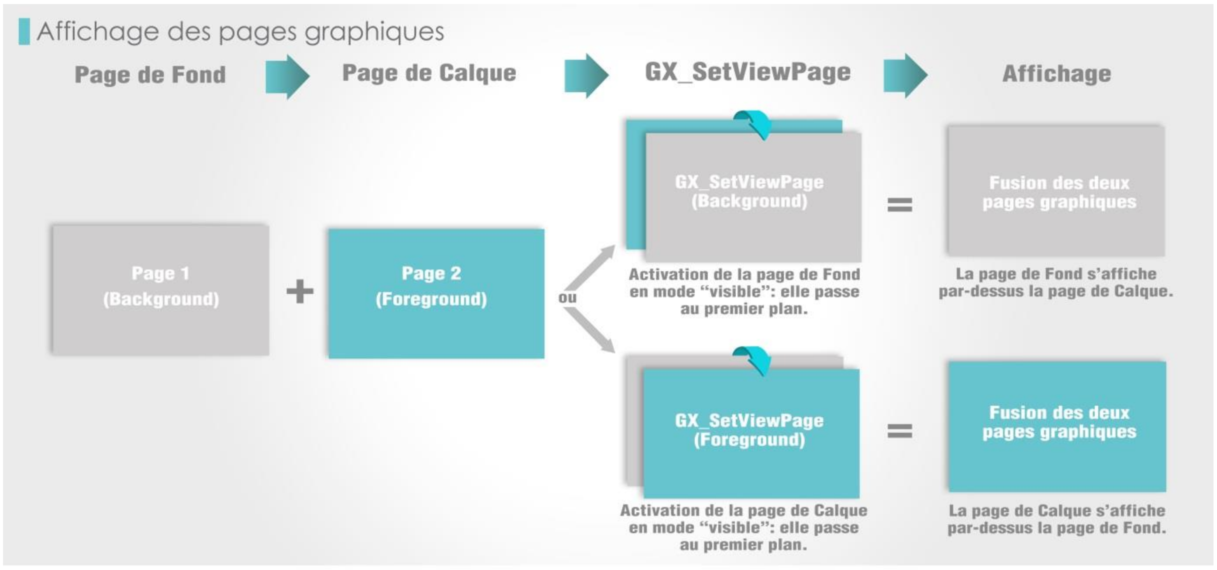


La fonction GX_SetViewPage ( )

```
GX_SetViewPage ( unsigned char numMsg,    →  RESERVED pour l'option CAN, toujours égal à 0,
                 unsigned char option,    →  RESERVED, toujours égal à 0,
                 unsigned char numPage ); →  0 pour passer PAGE_FOND au premier-plan,
                                             1 pour passer PAGE_CALQUE au premier-plan.
```

This command visually displays the selected layer: it gets displayed over the other.

During the merging of the two layers to create the final "look", the graphic elements placed upon them are placed one over another.



Affichage des pages graphiques
Page de Fond
Page de Calque
GX_SetViewPage
Affichage
Page 1 (Background)
+
Page 2 (Foreground)
ou
GX_SetViewPage (Background)
=
Fusion des deux pages graphiques
Activation de la page de Fond en mode "visible": elle passe au premier plan.
La page de Fond s'affiche par-dessus la page de Calque.
GX_SetViewPage (Foreground)
=
Fusion des deux pages graphiques
Activation de la page de Calque en mode "visible": elle passe au premier plan.
La page de Calque s'affiche par-dessus la page de Fond.

D. Main commands for the management of the graphic layers

GX_SetWorkPage **ESC 'p'**	Select one of the graphic layers as active one ('writing mode') (invisible on the screen)
GX_SetViewPage **ESC 'c'**	Display one of the graphic layers on the screen

Chapter 8: The active colors and drawing tools

A. Active colors

The format of the active colors consist of α, R, G, B :

- α or Alpha channel (0 to 255 or 0x00 to 0xFF),
- Red (0 to 255 or 0x00 to 0xFF),
- Green (0 to 255 or 0x00 to 0xFF),
- Blue (0 to 255 or 0x00 to 0xFF).

You can select the color of the foreground for the drawing objects (text, line, rectangle, pixel) or of the background for texts.

Les couleurs actives
Forecolor
Couleur d'avant-plan, appliquée aux outils de dessin et aux textes.
Backcolor
Couleur d'arrière-plan, appliquée uniquement pour le fond des textes.

IronGraph has a capacity of 16 million colors on the foreground layer and of 65.535 colors on the background layer. The foreground layer allows to create 256 levels of transparence (0 : transparent, 255 : opaque). Its format is (α, R, G, B) : 8 :8 :8:8.

The format of the background image, which does not manage transparence, is RGB 16 bits 5:6:5. This means that the value for red and blue is 5 bits, and 6 bits for green. When you program a color in 24 bits (RGB 8 :8 :8), i twill automatically be transformed to 16 bits 5 :6 :5.

A. The alpha channel

In order to work with different parts of an image separately and to put them one over another in order to create animated HMIs ("On/Off" buttons, sliders, gauges, counters, etc.), we use the alpha channel in order to turn pixels on the foreground layer transparent. This means that a transparent pixel (value 0 on the alpha channel) on the foreground layer will let you see the pixel of the background layer instead.

As we have done for the API (application programming interface) files, we now invite you to look in more detail at the definition of the basic colors of the training HMI, found in the following file:

- "IRG_GraphicEngine_Library.h".

For example:

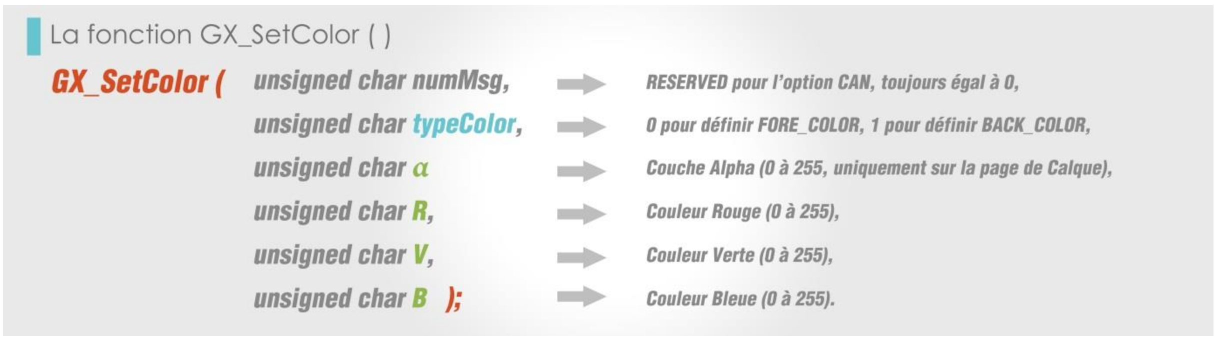
```
#define COLOR_BLACK                 0xFF, 0x00, 0x00, 0x00 (or: 255, 0, 0, 0)
#define COLOR_WHITE                 0xFF, 0xFF, 0xFF, 0xFF
#define TRANSPARENCE                0x00, 0x00, 0x00, 0x00
```

In addition to the basic colors of your project, found in the file "IRG_GraphicEngine_Library.h", found in the you can choose your own colors in your graphic software and add their definition to your application.

Note: In all diagram, which follow, the zones on the foreground layer where the pixel are transparent are indicated by the color magenta. We use the terminology "transparent color" for the pixels whose α layer is turned to 0.

1. Choosing a color

This function determines the active foreground color (forecolor) or background color (backcolor) for all commands which follow and which make use of this function (drawing functions and text functions):



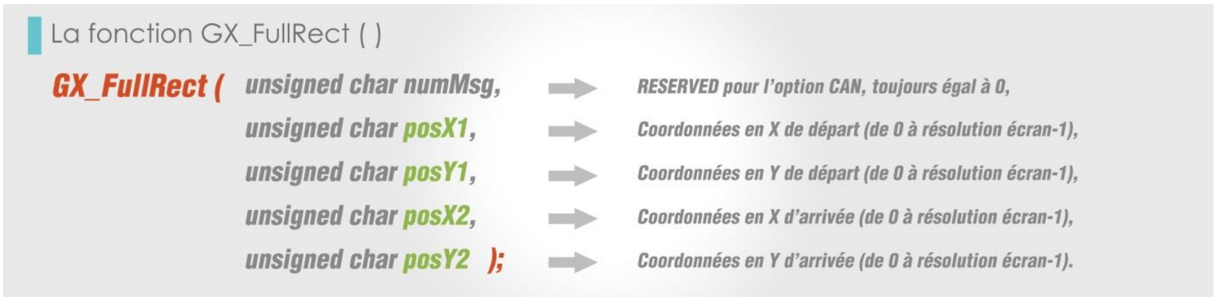
La fonction GX_SetColor ( )

GX_SetColor (	unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
	unsigned char typeColor,	0 pour définir FORE_COLOR, 1 pour définir BACK_COLOR,
	unsigned char α	Couche Alpha (0 à 255, uniquement sur la page de Calque),
	unsigned char R,	Couleur Rouge (0 à 255),
	unsigned char V,	Couleur Verte (0 à 255),
	unsigned char B);	Couleur Bleue (0 à 255).

B. Drawing tools

In the API (Application Programming Interface) you will find all commands associated with the drawing tools. Thanks to this interface, you can draw filled or empty rectangles, circles, lines and single pixels in the forecolor which you have previously selected by using the command GX_SetColor.

Since all drawing tools work on the same principle, we will only present you the function GX_FullRect ( ), which you will probably using the most:



La fonction GX_FullRect ( )

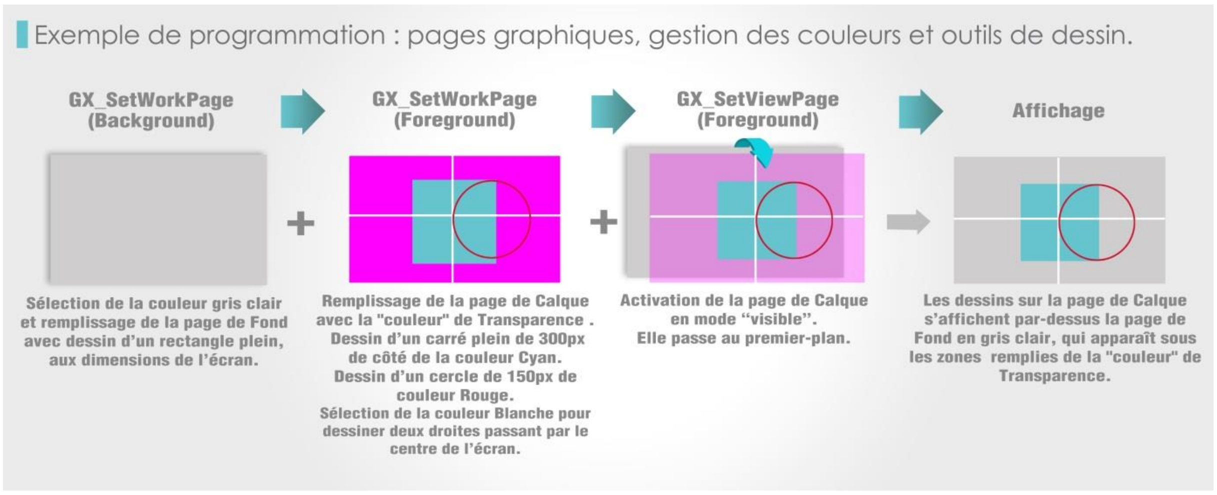
GX_FullRect (	unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
	unsigned char posX1,	Coordonnées en X de départ (de 0 à résolution écran-1),
	unsigned char posY1,	Coordonnées en Y de départ (de 0 à résolution écran-1),
	unsigned char posX2,	Coordonnées en X d'arrivée (de 0 à résolution écran-1),
	unsigned char posY2);	Coordonnées en Y d'arrivée (de 0 à résolution écran-1).

C. Main drawing commands

GX_SetColor **ESC ‘C’**	Select the active color of the foreground and background.
GX_Cls **ESC ‘E’**	Fill the entire screen with the active foreground color.
GX_SetPixel ( ) **ESC ‘P’**	Draw a pixel at X,Y in the active foreground color.
GX_Line ( ) **ESC ‘D’**	Draw a straight line (y = ax + b) in the active foreground color.
GX_Circle ( ) **ESC ‘C’**	Draw a circle in the active foreground color.
GX_Rect ( ) **ESC ‘R’**	Draw a rectangle in the active foreground color.
GX_FullRect ( ) **ESC ‘r’**	Draw a filled rectangle in the active foreground color.
GX_OpacityFullRect ( ) **ESC ‘B’**	Draw a filled rectangle in the active foreground color and adjustable opacity.

D. Examples

We will now take the knowledge about the graphic layers, the active colors, the alpha channel, and the drawing tools, which you have just acquired, and apply it to the example below:


Exemple de programmation : pages graphiques, gestion des couleurs et outils de dessin.
GX_SetWorkPage (Background)
GX_SetWorkPage (Foreground)
GX_SetViewPage (Foreground)
Affichage
Sélection de la couleur gris clair et remplissage de la page de Fond avec dessin d'un rectangle plein, aux dimensions de l'écran.
Remplissage de la page de Calque avec la "couleur" de Transparence . Dessin d'un carré plein de 300px de côté de la couleur Cyan. Dessin d'un cercle de 150px de couleur Rouge. Sélection de la couleur Blanche pour dessiner deux droites passant par le centre de l'écran.
Activation de la page de Calque en mode "visible". Elle passe au premier-plan.
Les dessins sur la page de Calque s'affichent par-dessus la page de Fond en gris clair, qui apparaît sous les zones remplies de la "couleur" de Transparence.

Code C++:

```
GX_SetWorkPage (PAGE_FOND);
GX_SetColor (RESERVED, FORE_COLOR, COLOR_GREY);
GX_FullRect (RESERVED, 0, 0, 799,479);

GX_SetWorkPage (PAGE_CALQUE);
GX_SetColor (RESERVED, FORE_COLOR, TRANSPARENCE);
GX_FullRect (RESERVED, 0, 0, 799,479);

GX_SetColor (RESERVED, FORE_COLOR, COLOR_CYAN);
GX_FullRect (RESERVED, 250, 90, 550,390);

GX_SetColor (RESERVED, FORE_COLOR, COLOR_RED);
GX_Circle (RESERVED, 400, 240, 150, 0, 360);

GX_SetColor (RESERVED, FORE_COLOR, COLOR_WHITE);
GX_Line (RESERVED, 0, 240, 799, 240);
GX_Line (RESERVED, 400, 0, 400, 479);
GX_SetViewPage (PAGE_CALQUE);
```

Chapter 9: Dynamically displaying text and images

We will now see how to use the two graphic layers and the alpha channel for dynamically displaying text and images of your HMI.

In most cases, the background layer is used to display the static parts of your HMI, this means the background image of your interface with all graphic elements in their static version “OFF”.

The foreground layer allows you to place and remove, over the static layer, texts and graphic elements in the version “ON” in a dynamic way.

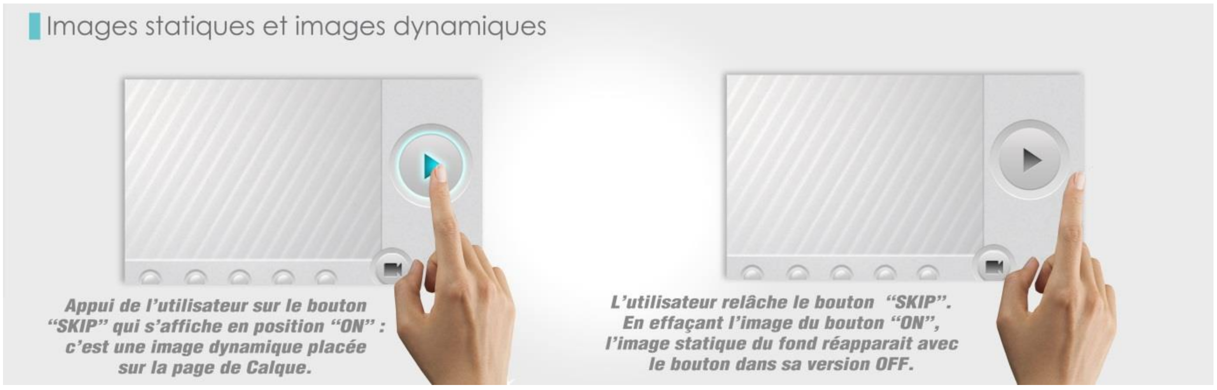
By displaying the foreground layer over the background layer, the dynamic elements get displayed and removed in front of the background layer, therefore creating a dynamic HMI.

C. Management of images

As we have seen, the background layer is used to display the interface in its static position “OFF”. Most of the time we will be displaying the background image with all graphic elements in “OFF” position.

The foreground layer is used to display the dynamic elements, this means the graphic elements which should visually change their appearance (into “ON” position”) when a certain situation is produced, for example the touching of touch zone.

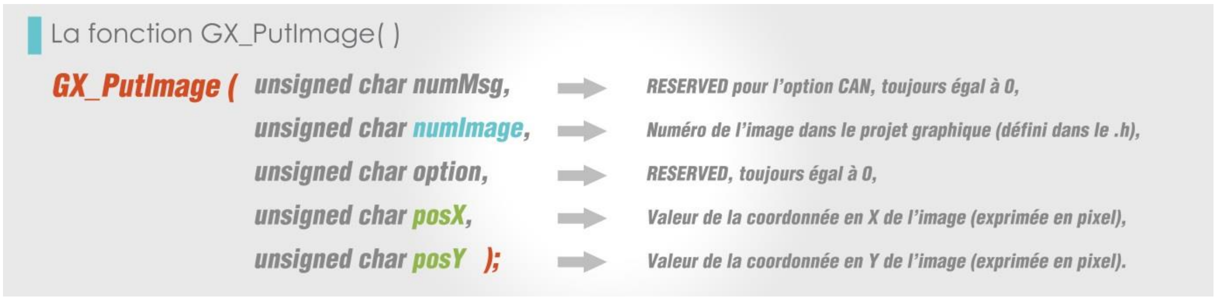
For example, when a user is touching the button “SKIP” of our traning HMI, the button should get displayed in “ON” position in order to indicate that it has been touched. When the user releases the button, the button should visually go back into “OFF” position.


Images statiques et images dynamiques
Appui de l’utilisateur sur le bouton “SKIP” qui s’affiche en position “ON” : c’est une image dynamique placée sur la page de Calque.
L’utilisateur relâche le bouton “SKIP”. En effaçant l’image du bouton “ON”, l’image statique du fond réapparait avec le bouton dans sa version OFF.

In order to realize this effect, we will explain you below how to display and remove images in a dynamic way by making use of the two graphic layers.

1. Displaying images

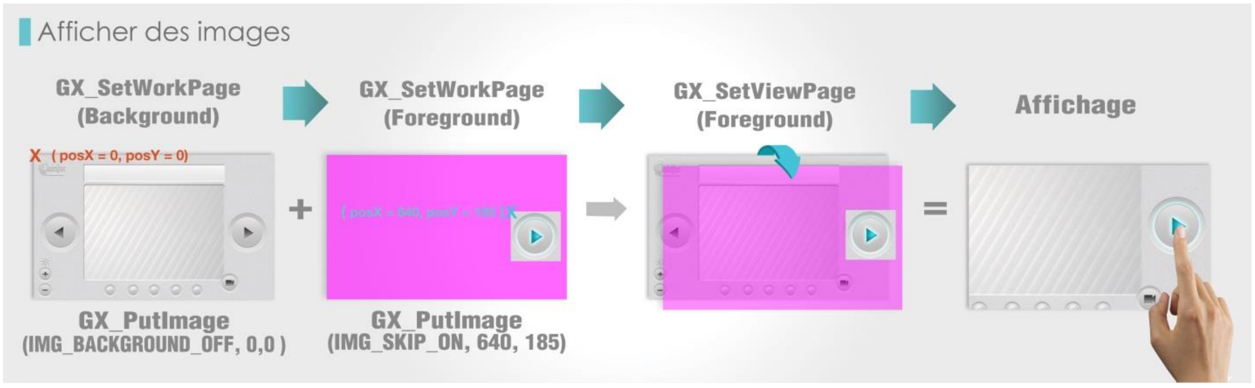
To display an image, you first need to select the graphic layer on which you would like to work (GX_SetWorkPage). Afterwards, you use the function GX_PutImage ( ) which takes as parameters the number of the image as well as its coordinates (position) on the display.



La fonction GX_PutImage( )

GX_PutImage (	unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
	unsigned char numImage,	Numéro de l'image dans le projet graphique (défini dans le .h),
	unsigned char option,	RESERVED, toujours égal à 0,
	unsigned char posX,	Valeur de la coordonnée en X de l'image (exprimée en pixel),
	unsigned char posY);	Valeur de la coordonnée en Y de l'image (exprimée en pixel).

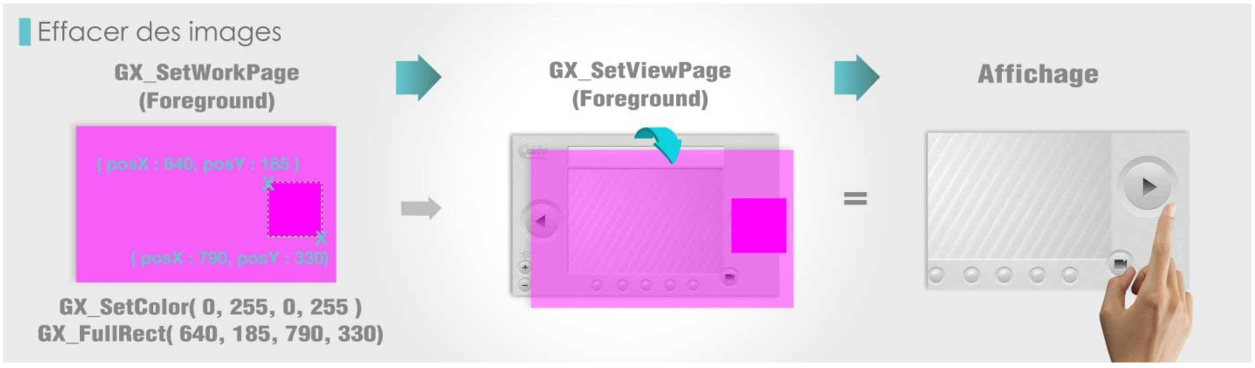
When we take again the example of the button "IMG_SKIP", the following steps are taken in order to display the button by making use of the two graphic layers:


Afficher des images
GX_SetWorkPage (Background)
GX_SetWorkPage (Foreground)
GX_SetViewPage (Foreground)
Affichage
X (posX = 0, posY = 0)
(posX = 640, posY = 185) X
GX_PutImage (IMG_BACKGROUND_OFF, 0,0)
GX_PutImage (IMG_SKIP_ON, 640, 185)

2. Removing images on the foreground layer

The dynamic elements, being placed on the foreground layer, can be "removed" by making use of the transparent channel. This "color", when paying applied to the pixels of the foregorund layer, turns the pixels transparent.

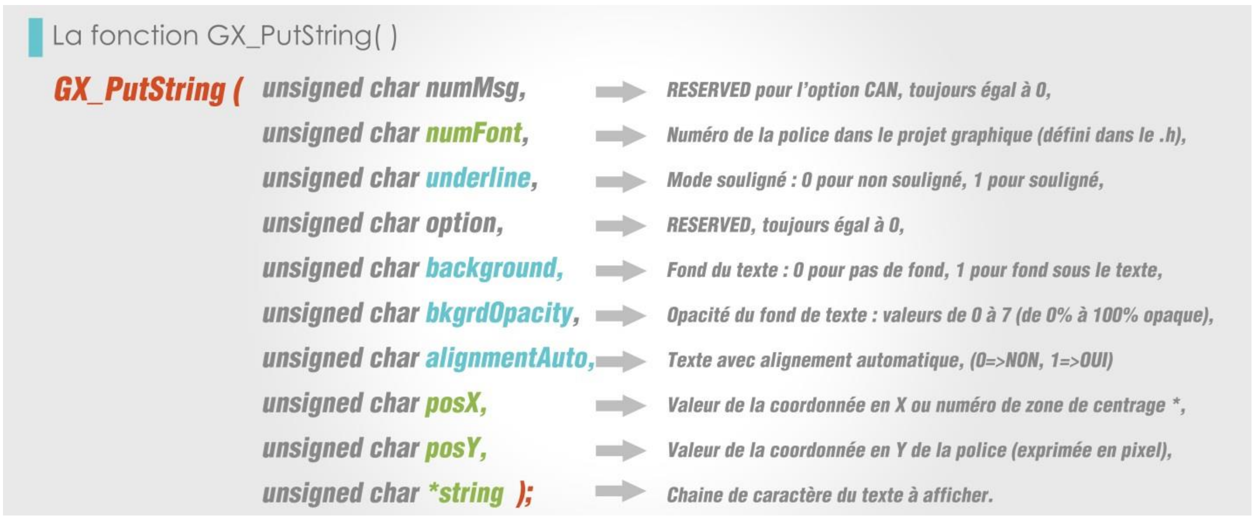
We will now use this characteristic in order to erase the pixels of an image which has been displayed. We will replace them by transparent pixels by drawing a filled rectangle with transparent "color" on top of the image on the foreground layer:


Effacer des images
GX_SetWorkPage (Foreground)
GX_SetViewPage (Foreground)
Affichage
(posX : 640, posY : 185)
(posX : 790, posY : 330)
GX_SetColor(0, 255, 0, 255)
GX_FullRect(640, 185, 790, 330)

D. Management of text

1. Displaying text

To display text, you first need to select the graphic layer on which you would like to work. Afterwards, you will use the function GX_PutString ( ). This function takes as parameters the number of the text font you would like to use, its coordinates (position) on the display, the actual text to display, as well as several parameters regarding the text options.

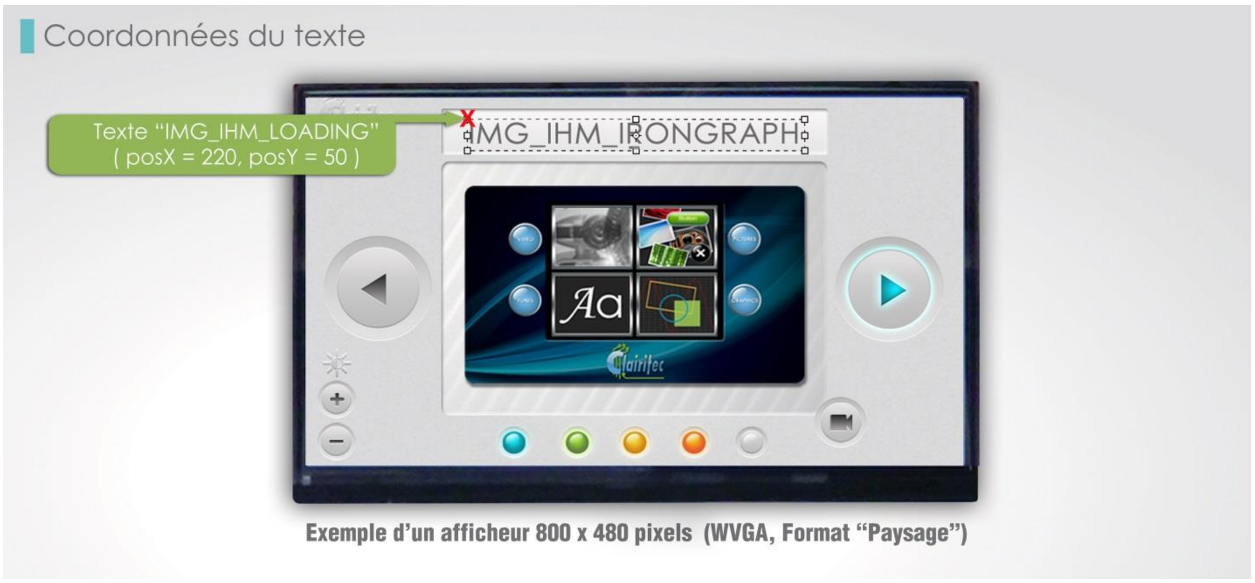


La fonction GX_PutString( )

GX_PutString (unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
unsigned char numFont,	Numéro de la police dans le projet graphique (défini dans le .h),
unsigned char underline,	Mode souligné : 0 pour non souligné, 1 pour souligné,
unsigned char option,	RESERVED, toujours égal à 0,
unsigned char background,	Fond du texte : 0 pour pas de fond, 1 pour fond sous le texte,
unsigned char bkgrdOpacity,	Opacité du fond de texte : valeurs de 0 à 7 (de 0% à 100% opaque),
unsigned char alignmentAuto,	Texte avec alignement automatique, (0=>NON, 1=>OUI)
unsigned char posX,	Valeur de la coordonnée en X ou numéro de zone de centrage *,
unsigned char posY,	Valeur de la coordonnée en Y de la police (exprimée en pixel),
unsigned char *string);	Chaine de caractère du texte à afficher.

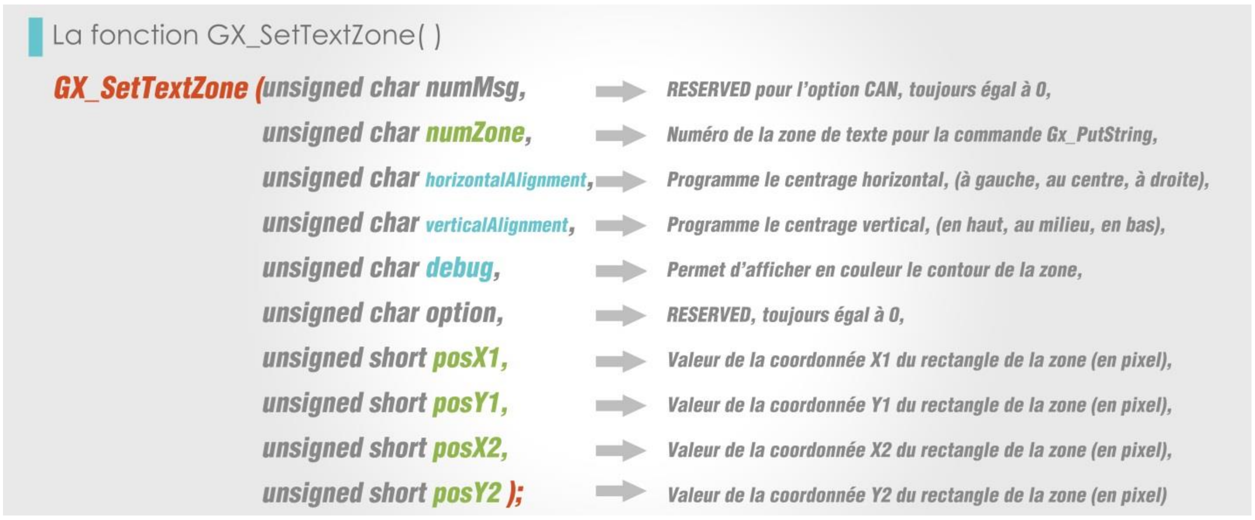
*****_See the command GX_SetTextZone below for more details about this option._

The value of the coordinates of a text, (option alignementAuto=NON), is indicated in pixel and takes as a reference the upper-left corner of the display:


Coordonnées du texte
Texte “IMG_IHM_LOADING” (posX = 220, posY = 50)
IMG_IHM_IRONGRAPH
Aa
Exemple d'un afficheur 800 x 480 pixels (WVGA, Format “Paysage”)

*However, an additional command called GX_SetTextZone () allows to choose the alignment of the text. Each text zone is defined with a number, as well as the alignment options.

To display a text with a specific alignment, you first need to create a text zone. Afterwards, you use the function GX_PutString with the argument alignmentAuto=OUI. Instead of entering the coordinates posX and posY, you need to put the number of the text zone, in which you would like to display the text, at the place of posX.



La fonction GX_SetTextZone( )

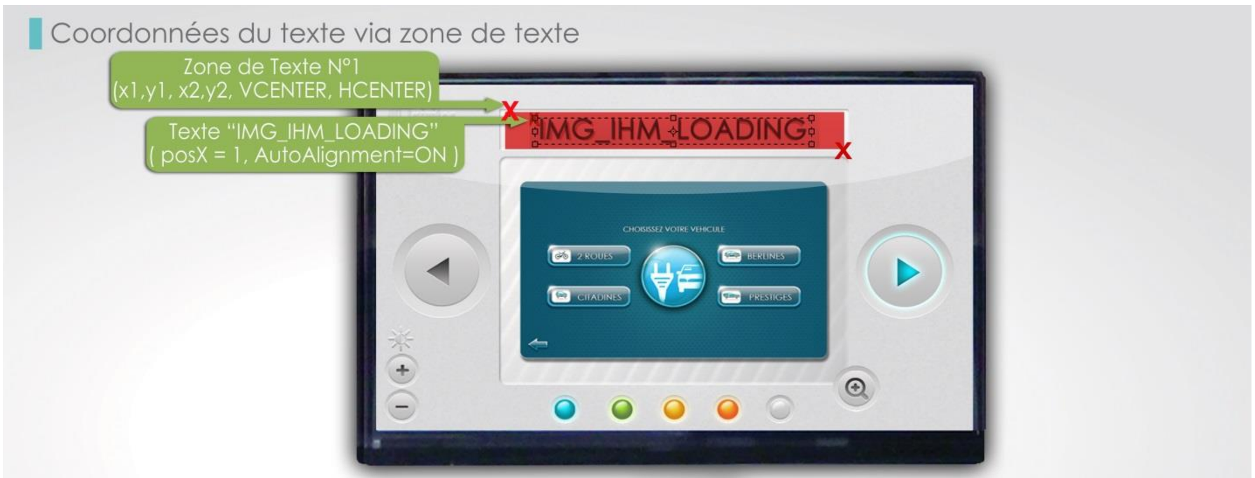
GX_SetTextZone (unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
unsigned char numZone,	Numéro de la zone de texte pour la commande Gx_PutString,
unsigned char horizontalAlignment,	Programme le centrage horizontal, (à gauche, au centre, à droite),
unsigned char verticalAlignment,	Programme le centrage vertical, (en haut, au milieu, en bas),
unsigned char debug,	Permet d'afficher en couleur le contour de la zone,
unsigned char option,	RESERVED, toujours égal à 0,
unsigned short posX1,	Valeur de la coordonnée X1 du rectangle de la zone (en pixel),
unsigned short posY1,	Valeur de la coordonnée Y1 du rectangle de la zone (en pixel),
unsigned short posX2,	Valeur de la coordonnée X2 du rectangle de la zone (en pixel),
unsigned short posY2);	Valeur de la coordonnée Y2 du rectangle de la zone (en pixel)

Once the characteristics of the text zone have been defined (alignment of the text and dimensions), you only need to send the command GX_PutString with the argument posX = numZone. The text will be aligned in the zone.

Note: If the text is longer or bigger that the dimensions/surface of the text zone, the text will be displayed at posX1 (text longer than the zone), and/or posY1 (text font is bigger in height than the text zone).

When you use the text in "Background ON" mode, (with a background color or transparence), the backcolor will fill out the entire surface of the text zone.

Example of text being displayed in a text zone with horizontal and vertical alignment:


Coordonnées du texte via zone de texte
Zone de Texte N°1
(x1,y1, x2,y2, VCENTER, HCENTER)
Texte "IMG_IHM_LOADING"
(posX = 1, AutoAlignment=ON)
X
IMG_IHM_LOADING
X
CHOISISSEZ VOTRE VEHICULE
2 ROUES
BERLINES
CITADINES
PRESTIGES

Exemple d'un afficheur 800 x 480 pixels (WVGA, Format "Paysage")

Once a text zone has been defined, it stays like this unless you define another one using the same number numZone with the command GX_SetTextZone.

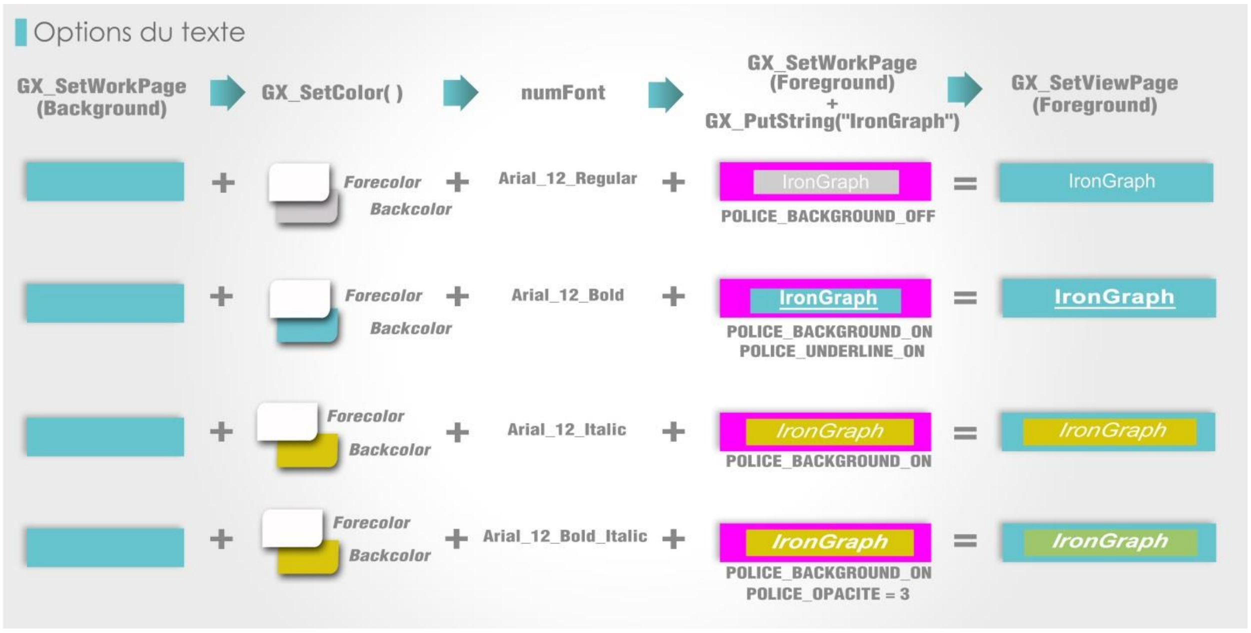
It is necessary to send as much commands to define text zones as the number of different text alignments wanted on the display. You can define up to 100 text zones for each graphic layer.

2. Text options

In GraphConverter, you select the text font to be used, and you can also save this text font in different styles (size, bold or in italic).

Afterwards, thanks to various options in the parameters of the function GX_PutString ( ), you can add effects to your text:

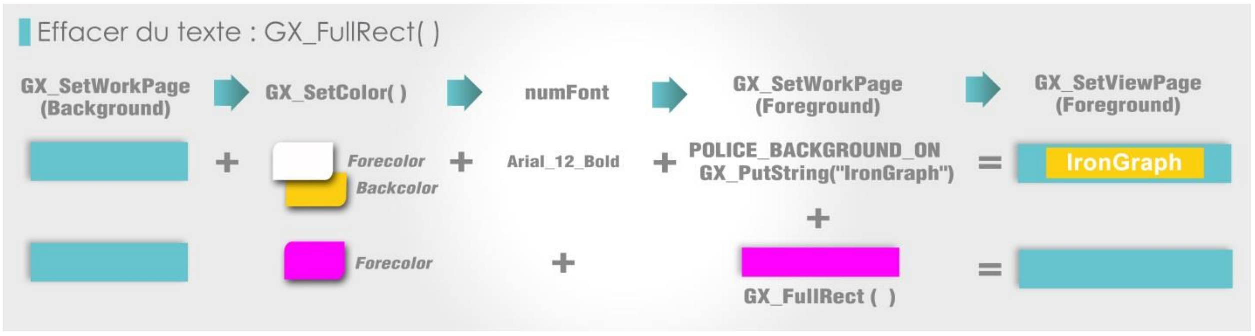
- With or without background (option "FONT_BACKGROUND_ON" or " FONT _BACKGROUND_OFF"),
- With or without alignment (option « ALIGNMENT_ON » or « ALIGNMENT_OFF »)
- Underlined or not (option " FONT _UNDERLINE_ON" or " FONT _UNDERLINE_OFF"),
- Opacity of the background (option " FONT _OPACITY", adjustable 1 à 7).



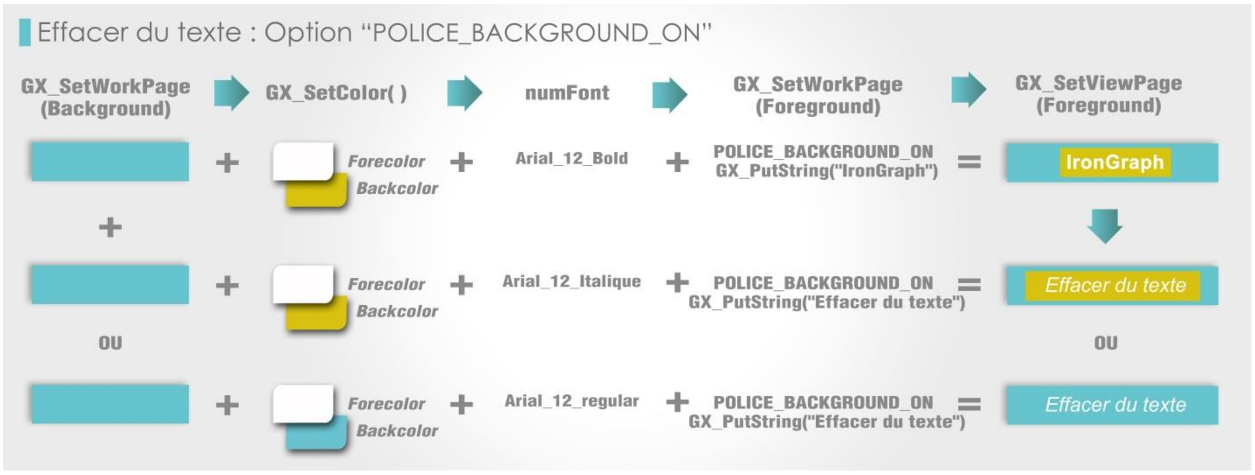
3. Erasing text on the foreground layer

There are two ways of erasing text:

➔ You can use the function GX_FullRect () to draw a filled rectangle with transparence on top of the text you would like to erase :

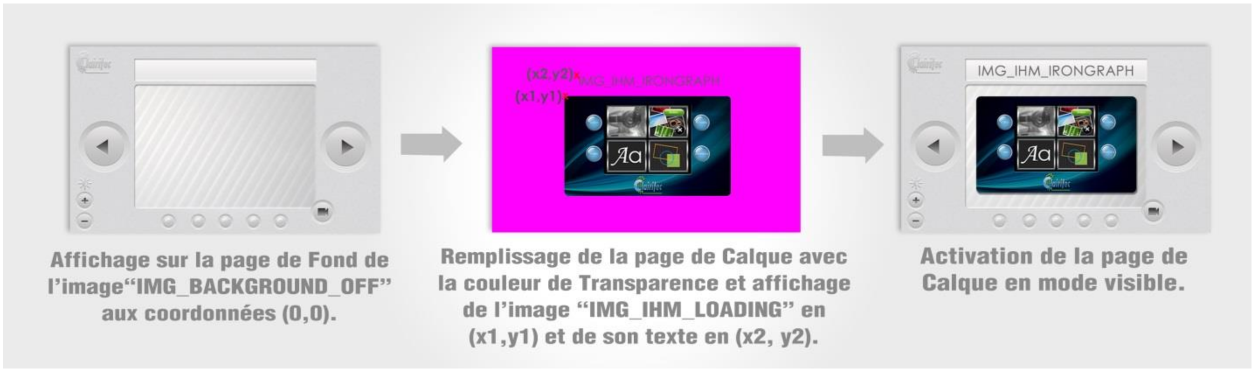

Effacer du texte : GX_FullRect( )
GX_SetWorkPage (Background)
GX_SetColor( )
numFont
GX_SetWorkPage (Foreground)
GX_SetViewPage (Foreground)
Forecolor
Backcolor
Arial_12_Bold
POLICE_BACKGROUND_ON
GX_PutString("IronGraph")
IronGraph
GX_FullRect ( )

➔ If you want to erase text in order to write another one at the same position (for example the title of the new picture which is displayed), you can use the function GX_PutString () with the option "FONT _BACKGROUND_ON". As we have already seen, this command allows to display a text with a certain background color. This background color of the text will be displayed on top of the old text, therefore "erasing" the previous text.


Effacer du texte : Option "POLICE_BACKGROUND_ON"
GX_SetWorkPage (Background)
GX_SetColor( )
numFont
GX_SetWorkPage (Foreground)
GX_SetViewPage (Foreground)
Forecolor
Backcolor
Arial_12_Bold
POLICE_BACKGROUND_ON GX_PutString("IronGraph")
IronGraph
Arial_12_Italique
POLICE_BACKGROUND_ON GX_PutString("Effacer du texte")
Effacer du texte
OU
Arial_12_regular
POLICE_BACKGROUND_ON GX_PutString("Effacer du texte")
Effacer du texte

E. Examples

Example of the display of a text and a picture on the two graphic layers


(x2,y2)
(x1,y1)
IMG_IHM_IRONGRAPH
Affichage sur la page de Fond de l'image"IMG_BACKGROUND_OFF" aux coordonnées (0,0).
Remplissage de la page de Calque avec la couleur de Transparence et affichage de l'image "IMG_IHM_LOADING" en (x1,y1) et de son texte en (x2, y2).
Activation de la page de Calque en mode visible.

Code C:

```
GX_SetWorkPage(PAGE_FOND);                   // Selecting of the background layer
GX_PutImage (IMG_BACKGROUND_OFF, 0, 0);      // Displaying the static background image of the HMI at
                                                position (0,0)
GX_SetWorkPage(PAGE_CALQUE);                 // Selection of the foreground layer
GX_SetColor (COLOR_TRANSPARENT);             // Selection of the transparent "color"
GX_FullRect (0, 0, 799,479) ou GX_Cls ()     // Filling the entire foreground layer with this "color"
                                                (transparence)
GX_PutImage (IMG_IHM_LOADING, x1, y1);       // Displaying the image at position (x1, y1)

GX_SetTextZone(1, x2, y2, x'2, y'2, VCENTER, HCENTER) ;//Defining the text zone n°1 with the surface
                                                        //x2, y2→x'2,y'2, //horizontal and vertical centering.

GX_PutString ("IHM-EXEMPLE", Alignmentauto_ON, 1, …); // Displaying the text "IHM EXEMPLE" in the text zone
                                                          n°1 with automatic alignment.

GX_SetViewPage(PAGE_CALQUE);                 // Turning the foreground layer visible.
```

F. Main commands to display text and images

GX_SetTextZone **ESC 'Z'**	Define a text zone with automatic alignment.
GX_PutString **ESC 'S'**	Display a chain of characters in a defined text font.
GX_PutImage **ESC 'I'**	Display an image, stored in the internal memory of the HMI board.

Chapter 10: The touchscreen

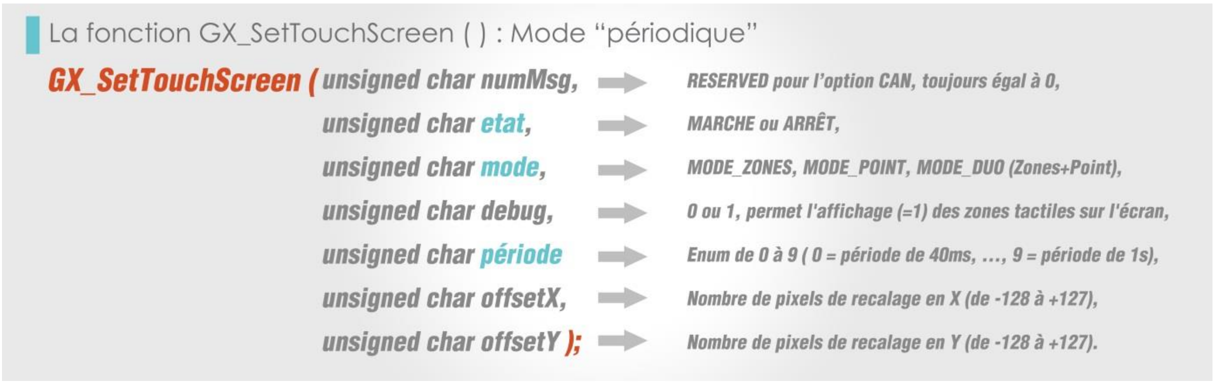
IronGraph manages the signals, emitted from the touchscreen, to detect when someone touches or (or releases) a specific point on the display.

A. Activating or deactivating the touchscreen

By default, the touchscreen is deactivated. The activation or deactivation of the touchscreen is done by using the command GX_SetTouchScreen ( ), where the respective parameter (ACTIVATE or DEACTIVATE) is entered.

- ➢ Activating the touchscreen means that the circuit of the touchscreen is activated, thus emitting automatically information about whether the touchscreen is touched.

- ➢ Deactivating the touchscreen "cuts" the interface with the touchscreen: IronGraph does not send any information about the touchscreen anymore. This function is useful for applications where a low energy consumption is very important.



La fonction GX_SetTouchScreen ( ) : Mode "périodique"

```
GX_SetTouchScreen ( unsigned char numMsg,   ➡  RESERVED pour l'option CAN, toujours égal à 0,
                    unsigned char etat,     ➡  MARCHE ou ARRÊT,
                    unsigned char mode,     ➡  MODE_ZONES, MODE_POINT, MODE_DUO (Zones+Point),
                    unsigned char debug,    ➡  0 ou 1, permet l'affichage (=1) des zones tactiles sur l'écran,
                    unsigned char période   ➡  Enum de 0 à 9 ( 0 = période de 40ms, ..., 9 = période de 1s),
                    unsigned char offsetX,  ➡  Nombre de pixels de recalage en X (de -128 à +127),
                    unsigned char offsetY ); ➡ Nombre de pixels de recalage en Y (de -128 à +127).
```

As you can see, this function also allows you to adjust the period of emission regarding the feedback of the touchscreen (touched or non-touched).

<u>For IronGraph in RS232 mode</u>, this command allows you to also to adjust the content of the frame emitted as feedback about the status of the touchscreen (parameter "mode").
IronGraph emits 1 byte, 4 bytes, or 5 bytes, depending on the mode which has been programmed:

- Mode « Zone », 1 byte, number of the touch zone which has been touched.
- Mode « Point », 4 bytes, coordinates of the point touched.
- Mode « Zone+Point », 5 bytes, containing the number of the zone touched + the coordinates of the point touched within the zone.

<u>For IronGraph in CAN mode</u>, the frame emitted as feedback always has the same format (zone + coordinates), regardless of the type of mode you program in the parameters of the function.

B. The different types of feedback of the touchscreen

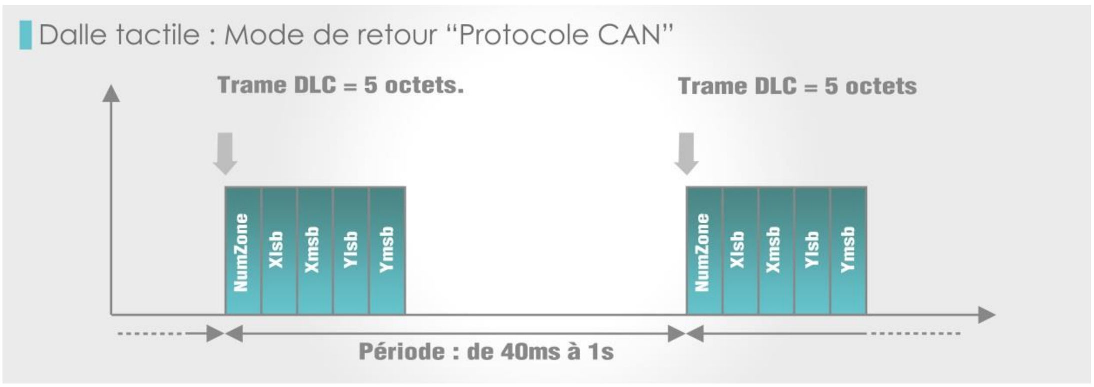
Once the touchscreen has been activated with the respective command, the IronGraph board scans the status of the touchscreen and gives a periodical feedback about it.

The format of the feedback given about the status of the touchscreen depends on the type of communication protocol used: RS232 or CAN.

1. CAN protocol

The frame is identical, regardless of the feedback mode, programmed in the command GX_SetTouchScreen.
Its DLC is fixed to 5 bytes.

Byte1	Byte2	Byte3	Byte4	Byte5
Number of zone touched or released	Coordinate X where the touchscreen has been touched (MSB)	Coordinate X where the touchscreen has been touched (LSB)	Coordinate Y where the touchscreen has been touched (MSB)	Coordinate Y where the touchscreen has been touched (LSB)

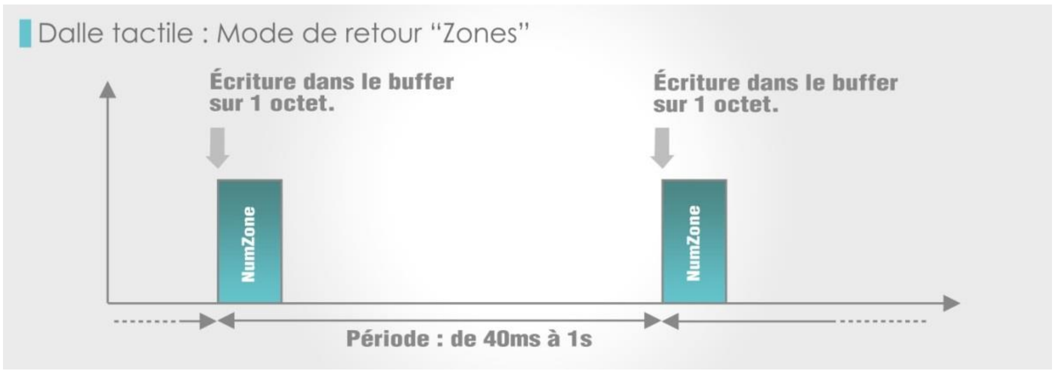


2. RS232 protocol - "zone" mode

When "zone" has been chosen as a feedback mode, IronGraph gives as feedback the number of the touch zone which has been activated (see the paragraph *Définition des zones tactiles d'un écran*).

This information is sent with 1 byte:

- Bit 7: 0 or 1 (zone touched/not touched)
- Bit 6 to Bit 0: number of the respective touch zone


Dalle tactile : Mode de retour "Zones"
Écriture dans le buffer sur 1 octet.
Écriture dans le buffer sur 1 octet.
NumZone
NumZone
Période : de 40ms à 1s

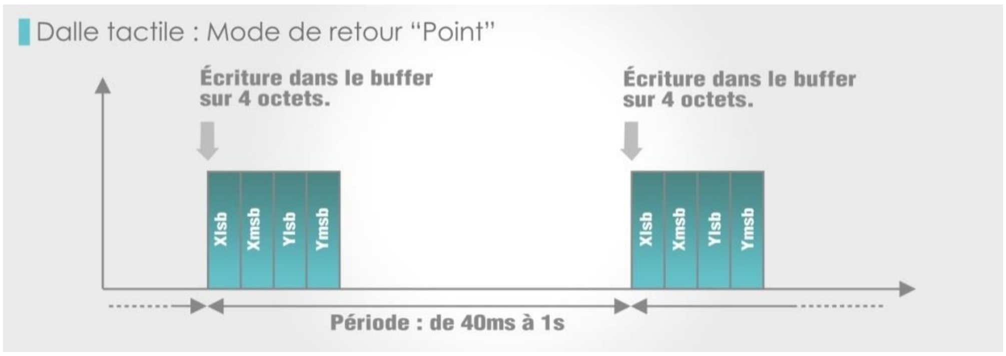
Note: The "zone" mode is the one used most by our clients. It allows you to define up to 50 touch zones for each HMI screen, each touch zone representing a touchable element of the user interface (for example a button).

3. RS232 protocol - "point" mode

When "point" has been chosen as a feedback mode, IronGraph gives as a feedback the coordinates of the touched point on the display.

This information is sent with 4 bytes:

- Byte 1 : Xlsb,
- Byte 2 : Xmsb,
- Byte 3 : Ylsb,
- Byte 4 : Ymsb.


Dalle tactile : Mode de retour "Point"
Écriture dans le buffer sur 4 octets.
Écriture dans le buffer sur 4 octets.
Xlsb
Xmsb
Ylsb
Ymsb
Xlsb
Xmsb
Ylsb
Ymsb
Période : de 40ms à 1s

Note: The "point" mode is not used very often for touchscreen user interfaces. It can be useful if you would like to draw single pixels where the touchscreen is touched (for example to simulate a graphic tablet with a pen). In this case, it is recommended to use a capacitive touchscreen.

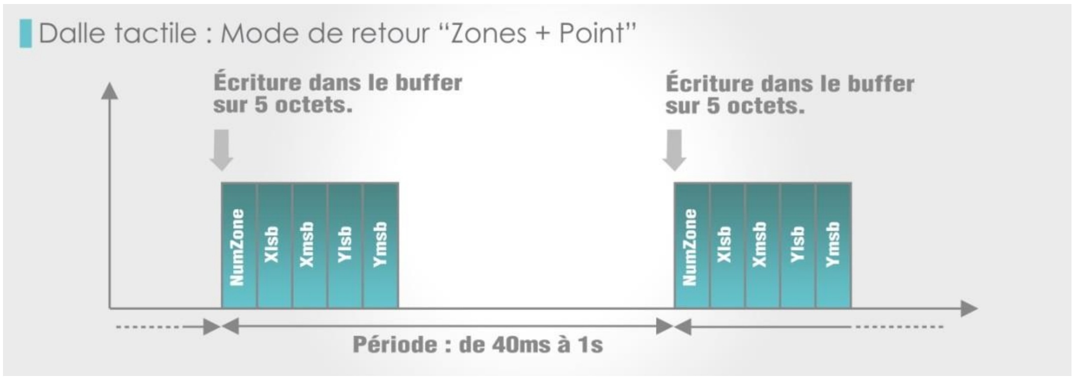
4. RS232 protocol - "zone + point" mode

When "point" has been chosen as a feedback mode, IronGraph gives as a feedback the number of the touch zone which has been activated, as well as the precise coordinates of the touched point on the display.

This information is sent with 5 bytes :

Byte 1: Zone
- Bit 7: 0 or 1 (zone touched/not touched)
- Bit 6 to Bit 0: number of the respective touch zone

Byte 2 to 5: Coordinates of the point
- Byte 1 : Xlsb,
- Byte 2 : Xmsb,
- Byte 3 : Ylsb,
- Byte 4 : Ymsb.


Dalle tactile : Mode de retour "Zones + Point"
Écriture dans le buffer sur 5 octets.
NumZone
Xlsb
Xmsb
Ylsb
Ymsb
Écriture dans le buffer sur 5 octets.
NumZone
Xlsb
Xmsb
Ylsb
Ymsb
Période : de 40ms à 1s

Note: The "zone + point" mode is used when you would like to display a dynamic element which follows your finger, for example a cursor to regulate a setting. You first define the zone of your cursor with the "zone" mode and then, once this zone is touched, you make use of the coordinates of the precise point (x, y) in order to display the image of your cursor.

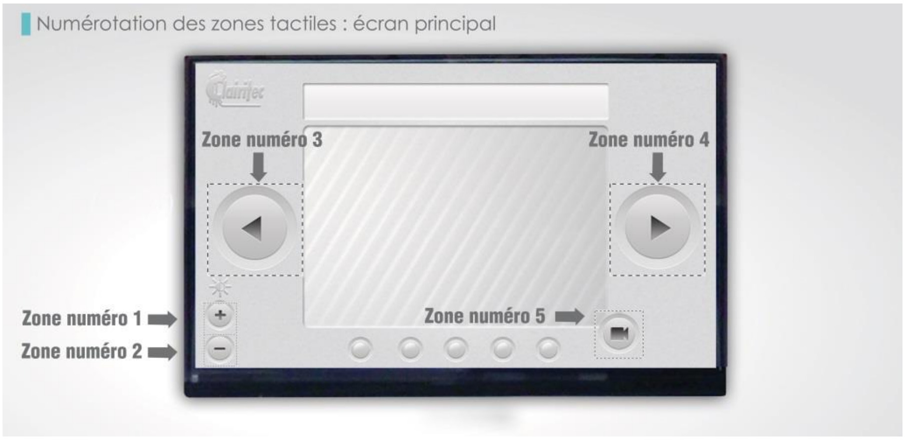
C. Definition of the touch zones of an HMI screen

1. Numbering of the touch zones

When using the "zone" or "zone + point" mode, the touch zones on the different HMI screens will be numbered and their coordinates will be saved.

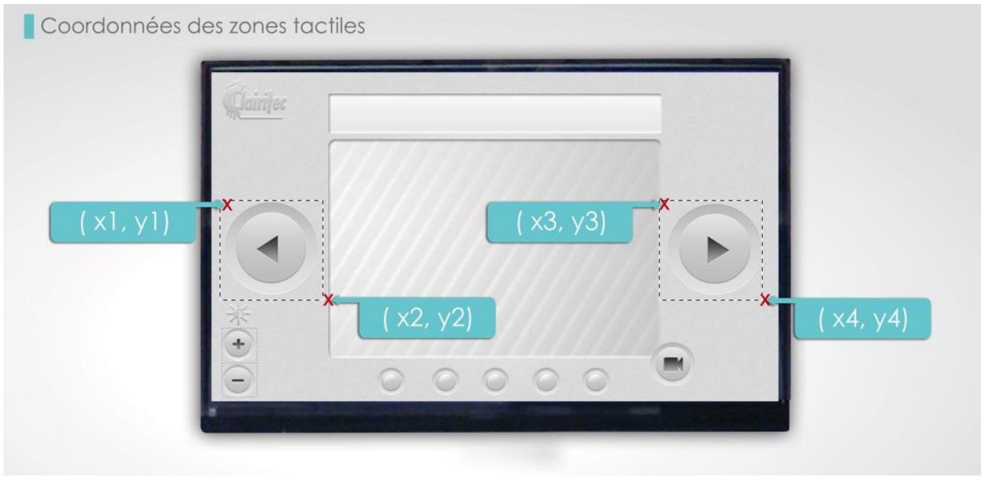
A touch zone is a rectangular zone, containing the touchable element. There are as many touch zones as touchable objects on an HMI screen. Each zone is defined by a number which is used in the feedback (this information taking 1 byte), emitted by IronGraph when a zone is touched or released.

Si vous optez pour le mode de retour "Zones" ou "Zones + Point", vous devez numérotez et définir les coordonnées des différentes zones tactiles pour chaque écran composant votre IHM.

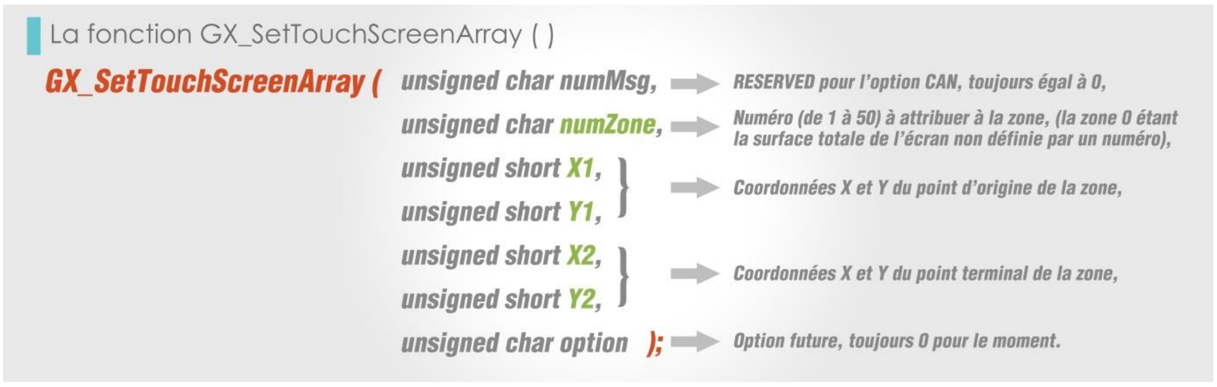


2. Coordinates of the touch zones

The coordinates of the touch zones are those of the rectangular zone, containing the touchable element.



The command to define a touch zone is the function GX_Set TouchScreenArray (). This function takes as parameters the number of the zone as well as its rectangular coordinates, defining the touchable area on the display.



La fonction GX_SetTouchScreenArray ( )

```
GX_SetTouchScreenArray ( unsigned char numMsg,   → RESERVED pour l'option CAN, toujours égal à 0,
                         unsigned char numZone,  → Numéro (de 1 à 50) à attribuer à la zone, (la zone 0 étant
                                                   la surface totale de l'écran non définie par un numéro),
                         unsigned short X1,  }
                         unsigned short Y1,  }   → Coordonnées X et Y du point d'origine de la zone,
                         unsigned short X2,  }
                         unsigned short Y2,  }   → Coordonnées X et Y du point terminal de la zone,
                         unsigned char option  ); → Option future, toujours 0 pour le moment.
```

We invite you to look in more detail at the definitions (numbering and coordinates) of the touch zones of the training HMI in the following file:

- "IRG_RS232_WVGA_HMI.h"*

*This file is created by the HMI editor. More details can be found in the chapter Chapitre 12 : L'éditeur d'IHM de GraphConverter.

D. Managing the touch zones

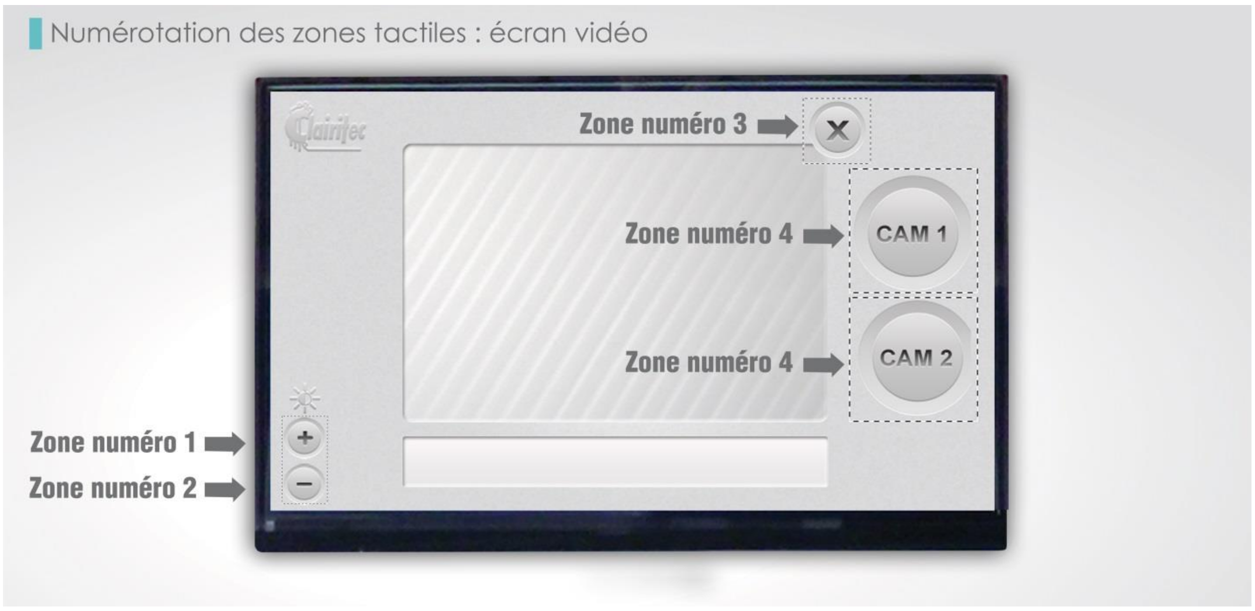
It is necessary to send as many commands to define touch zones as there are zones.

You can define up to 50 touch zones for each HMI screen. The zone number 0 is the 51st zone, covering all non-defined zones of an HMI screen.

A defined touch zone stays valid until the command GX_SetTouchScreenArray is sent again, using the same number of touch zone.

If your HMI has several HMI screens, the numbering (from 1 to 50) starts over again for each HMI screen. For each HMI screen, the touch zones need to be defined separately by using the command GX_SetTouchScreenArray.

For example, in our training HMI, the second HMI screen "VIDEO", also contains 3 touch zones:



E. Main commands to manage the touchscreen

GX_SetTouchScreen **ESC 't'**	Activate or deactivate the touchscreen.
GX_SetTouchScreenArray **ESC 'z'**	Define touch zones.
GX_TestTouchScreen **ESC 'm'**	Start the internal touchscreen test.

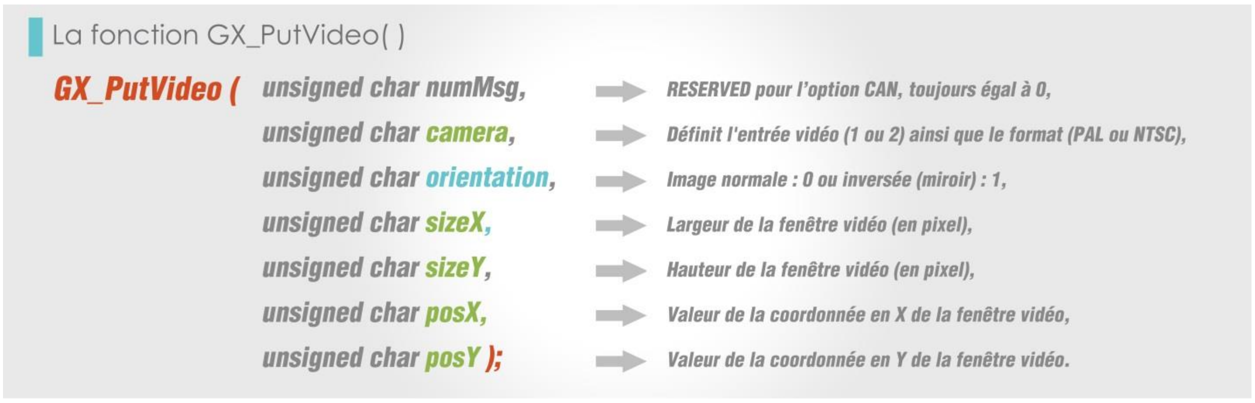
Chapter 11: Real time video – only for IronGraph

With the IronGraph board, you have the possibility to connect two cameras to the two video entries integrated on the board (however, only one real-time video can be shown at once).

The compatible video format is NTSC and PAL (analogical).

The size of the video window is customizable (length x height in pixel) and can be positioned at any point on the display.

To display the video flux from the camera, the function GX_PutVideo ( ) is used:



La fonction GX_PutVideo( )

GX_PutVideo (	unsigned char numMsg,	RESERVED pour l'option CAN, toujours égal à 0,
	unsigned char camera,	Définit l'entrée vidéo (1 ou 2) ainsi que le format (PAL ou NTSC),
	unsigned char orientation,	Image normale : 0 ou inversée (miroir) : 1,
	unsigned char sizeX,	Largeur de la fenêtre vidéo (en pixel),
	unsigned char sizeY,	Hauteur de la fenêtre vidéo (en pixel),
	unsigned char posX,	Valeur de la coordonnée en X de la fenêtre vidéo,
	unsigned char posY);	Valeur de la coordonnée en Y de la fenêtre vidéo.

Concerning the selection of the video entry and video format to be used, the definition of the 4 possible combinations can be found in the file "IRG_RS232_GraphicEngine_Library .h ":

```
#define CAMERA_1_PAL                    0
#define CAMERA_2_PAL                    1
#define CAMERA_1_NTSC                   2
#define CAMERA_2_NTSC                   3
```

Note: To avoid any distortion of the picture, it is advised to keep the video window in a 4:3 format.

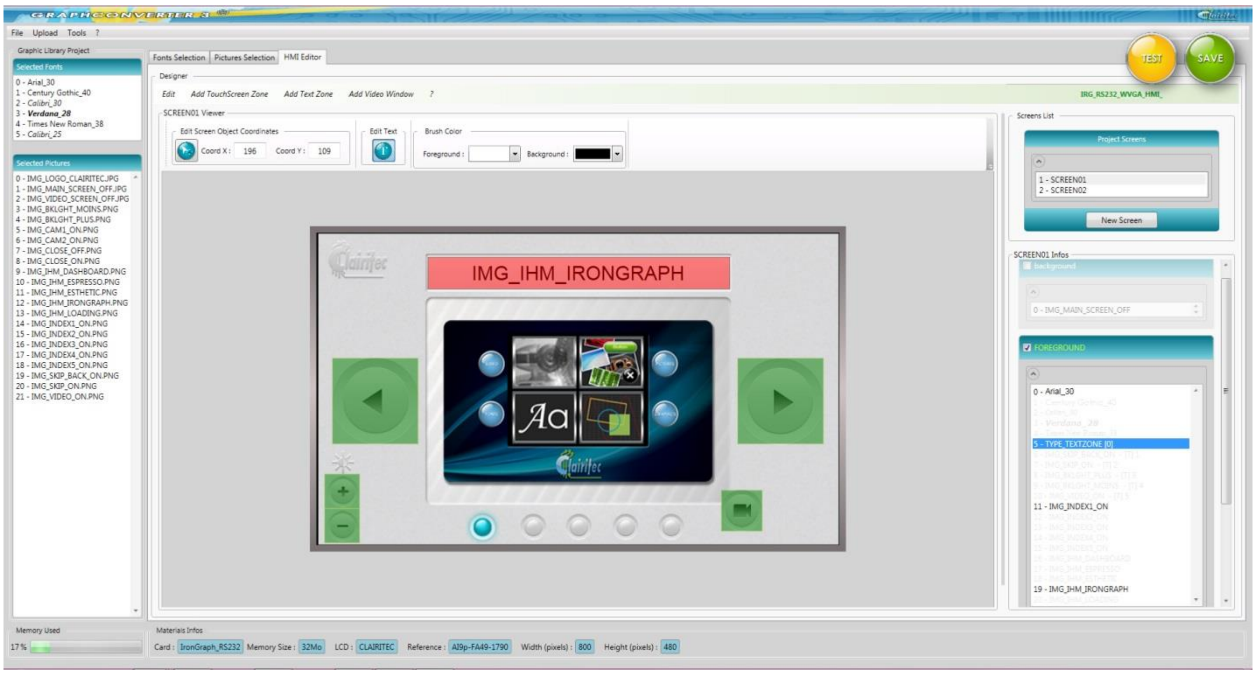
The moment IronGraph recevies the command GX_PutVideo , it creates the video window, in the size that you have chosen, on top of the 2 graphic layers. All commands to display graphic elements, which follow the activation of the video window, will display the respective elements, should these elements be positioned within the area of the video window, on top of the video window.

Note: In order to display images, primitive drawings, or text on top of the video, the respective elements have to be placed within the area of the video window.
To remove the video window, either a new HMI screen can be displayed, or its surface can be covered by one or several elements on one of the two graphic layers.

Chapter 12: The HMI Editor of GraphConverter

1. Presentation

The HMI editor allows you not only to place the objects of your graphic library on the HMI screens, it also allows you to define the initial status of all HMI screens which you create, and to save their configuration in the internal memory of the HMI board.



Selected Fonts
0 - Arial_30
1 - Century Gothic_40
2 - Calibri_30
3 - Verdana_28
4 - Times New Roman_38
5 - Calibri_25

Selected Pictures
0 - IMG_LOGO_CLAIRITEC.JPG
1 - IMG_MAIN_SCREEN_OFF.JPG
2 - IMG_VIDEO_SCREEN_OFF.JPG
3 - IMG_BKLGHT_MOINS.PNG
4 - IMG_BKLGHT_PLUS.PNG
5 - IMG_CAM1_ON.PNG
6 - IMG_CAM2_ON.PNG
7 - IMG_CLOSE_OFF.PNG
8 - IMG_CLOSE_ON.PNG
9 - IMG_IHM_DASHBOARD.PNG
10 - IMG_IHM_ESPRESSO.PNG
11 - IMG_IHM_ESTHETIC.PNG
12 - IMG_IHM_IRONGRAPH.PNG
13 - IMG_IHM_LOADING.PNG
14 - IMG_INDEX1_ON.PNG
15 - IMG_INDEX2_ON.PNG
16 - IMG_INDEX3_ON.PNG
17 - IMG_INDEX4_ON.PNG
18 - IMG_INDEX5_ON.PNG
19 - IMG_SKIP_BACK_ON.PNG
20 - IMG_SKIP_ON.PNG
21 - IMG_VIDEO_ON.PNG

Coord X : 196 Coord Y : 109
Foreground : Background :

1 - SCREEN01
2 - SCREEN02

0 - IMG_MAIN_SCREEN_OFF

0 - Arial_30
5 - TYPE_TEXTZONE [0]
11 - IMG_INDEX1_ON
19 - IMG_IHM_IRONGRAPH

17 %
Card : IronGraph_RS232 Memory Size : 32Mo LCD : CLAIRITEC Reference : AI9p-FA49-1790 Width (pixels) : 800 Height (pixels) : 480

The HMI editor allows you to place and position all graphic elements on each of the HMI screens which make up your user interface.

- Images
- Texts or rather text zones
- Touch zones
- Primitive drawings (rectangles, lines, etc.)
- Video windows (for IronGraph)

For each of those elements, you chose the available options (color, user mode, dimensions, etc.).

Afterwards you “hide” the elements which you do not want to be visible on the HMI screen upon loading it. This way, the initial status of the HMI screen is defined and saved in the internal memory of the HMI board. The HMI editor will create a header file .h which contains all coordinates and selected options of each of the objects placed on the HMI screen (also the ones which are hidden). This file helps you to write the code of the commands in C language in order to dynamically display the hidden elements afterwards.

The initial status of the HMI screens is saved in the same way in the HMI boards internal memory as the images and text fonts of the graphic library. The initial status of each HMI screen is retrieved by the command GX_PutScreen () which displays the chosen HMI screen (each HMI screen is attributed a number which is used in this command). This function is explained in more detail below.

2. The files of the HMI editor project

In the same way as the 3 files to manage all the graphic objects have been automatically created upon saving your graphic project, the HMI editor of GraphConverter will create three additional files when saving the HMI screens, created with the editor.

Your graphic project is composed out of a data file (.xml), a compiled data file (.hse), which will be uploaded into the HMI board, and a header file (.h) for your application in C language.

- The project file ("ProjectName_GraphConverter_HMI_xml"):

This file allows GraphConverter to maintain and re-open your HMI editor project. The name of the file is automatically attributed once you open the HMI editor and is based on the name which you have chosen for your graphic library. GraphConverter adds the code " _HMI_" to the name, and the extension is changed from .gxp to .xml.

Note: An HMI editor project is always linked to a corresponding graphic library.
If you want to create a new HMI editor project with the same graphic library as a basis, it is necessary to create a new project with a different name.

- The HMI editor header file ("ProjectName_GraphConverter_HMI_h"):

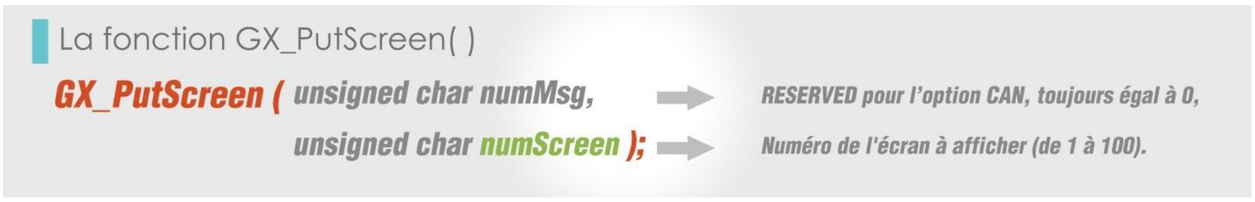
Once you have created your HMI screens with the HMI editor, GraphConverter will save the positions and characteristics of al graphic elements, touch zones, text zones, etc. By using the .h. file, you can fill in very easily the parameters of the corresponding API functions.

- The compiled data file ("ProjectName_GraphConverter_HMI_hse"):

This file contains the HMI screens which you have created in form of binary data. As it is the case with the HEX file, the HSE file is uploaded into the internal memory of the HMI board by GraphConverter.
This file describes all elements in their initial status (visible/invisible, active/inactive). Based on this file, the HMI board initializes, after the reception of the command GX_PutScreen (), an HMI screen with its two graphic layers and all elements placed upon them.

3. Displaying an HMI screen

The function GX_PutScreen initializes and displays all graphic objects (touch zones included) of a pre-defined HMI screen, saved in the internal memory of the HMI board.
The same way as GraphConverter attributes specific numbers to the images and text fonts of the graphic library, it also numerates the pre-defined HMI screens.
It is this number which is sent as a parameter with the command GX_PutScreen () in order to display an entire HMI screen.


La fonction GX_PutScreen( )

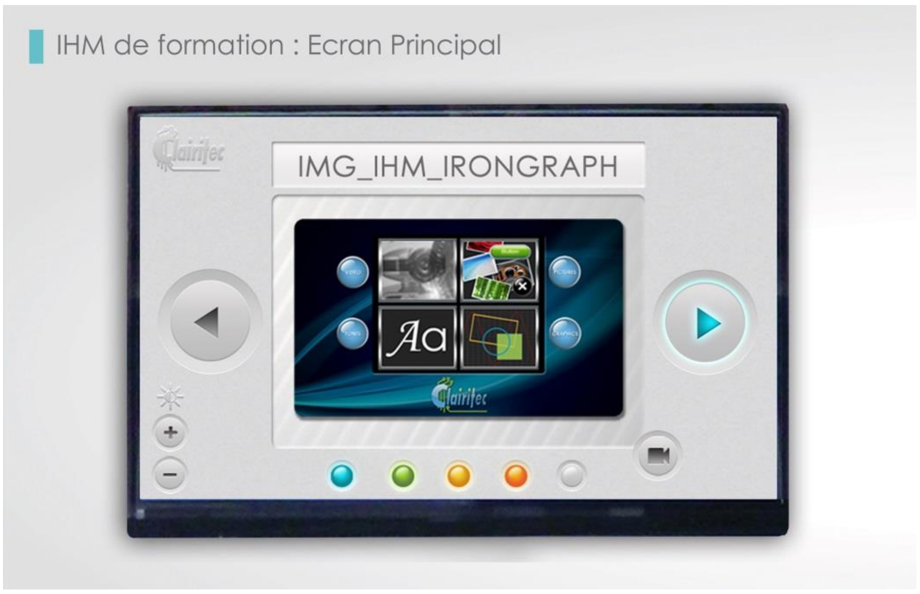
```
GX_PutScreen ( unsigned char numMsg,          → RESERVED pour l'option CAN, toujours égal à 0,
               unsigned char numScreen );     → Numéro de l'écran à afficher (de 1 à 100).
```

If the number of the HMI screen, sent with the command, is not referenced in the internal memory of the HMI board or if the number lies not within the standards, the HMI borne sends back an error message.

4. Example of an HMI editor project

As a reminder: Our project example is constituted out of two HMI screens. We have used the HMI editor to place all the elements of our HMI: images, touch zones, and text zones (with text alignment).

The main screen of the training HMI shows a rectangle with 2 touchable buttons “FORWARD” and “BACKWARD” which allow to change the background image shown in the middle of the screen. In the lower-left corner, 2 touchable buttons “+” and “-“ allow to adjust the backlight of the LCD. On the upper edge of the HMI screen, a text zone displays the title of the current interface, while on the lower edge, 5 colored lamps indicate which HMI screen, out of the 5 in total, is displayed.

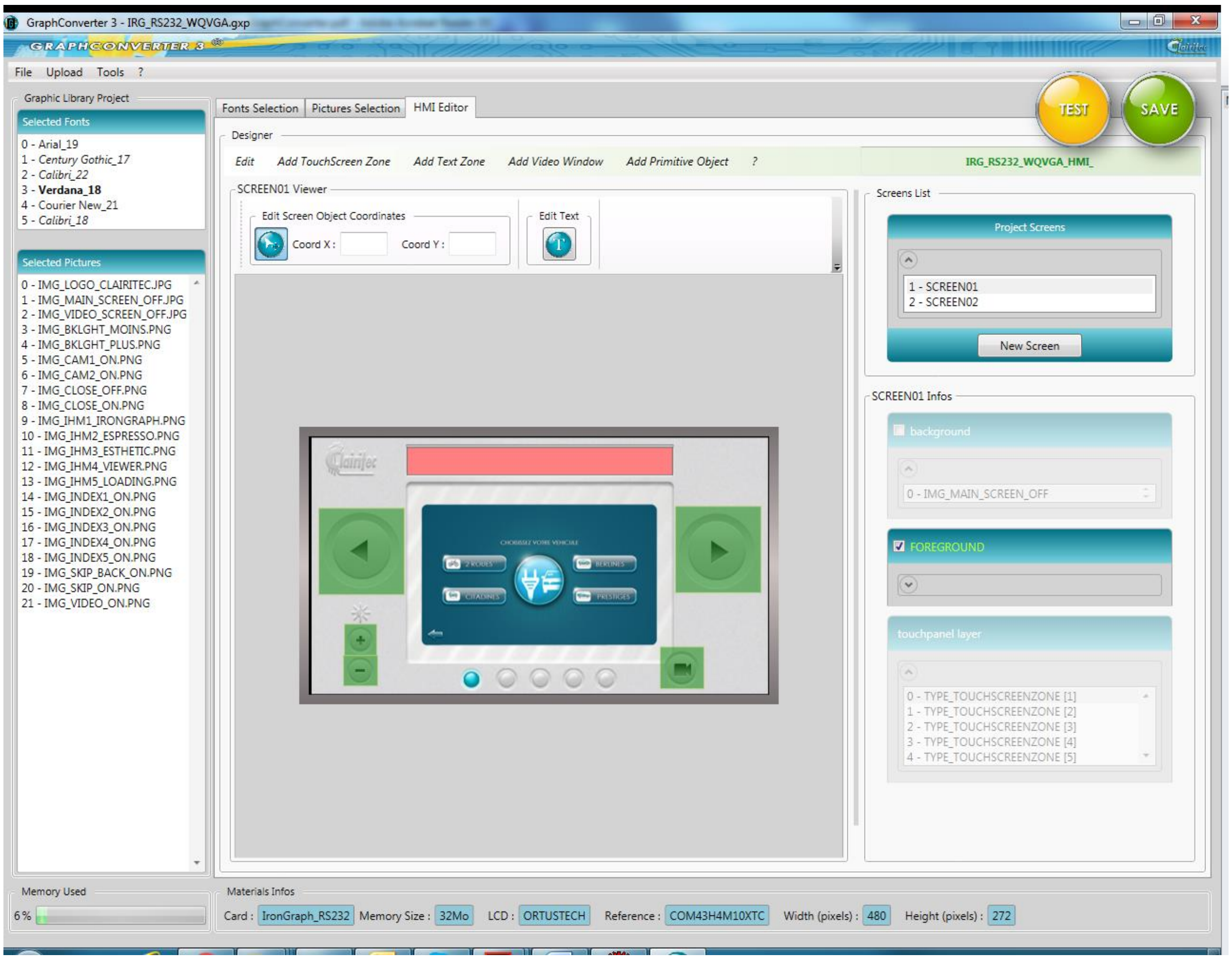


On the image above, the moment when a user touches the “FORWARD” button is shown. The “FORWARD” button (image °4 of our project) is therefore lit-up. The text of the HMI screen “IMG_IHM_IRONGRAPH” is displayed in the color and with the alignment previously chosen. The 4th indicator light (orange) is lit-up since the image shown in the middle of the screen constitutes the 4th one out of the 5 images in total.

The HMI editor has allowed us to place all these dynamic elements, to activate the touch zones which correspond to the “button” images, to activate the text zone with its automatic alignment option, and to define the text font to be used.

Once all of these elements had been placed, those ones which should not be displayed by default when loading the HMI screen with the command GX_PutScreen (), needed to be “hidden”.

Following a screenshot of the HMI editor with the HMI screen programmed to its initial status, ready to be uploaded to the HMI board:



Selected Fonts:
0 - Arial_19
1 - *Century Gothic_17*
2 - *Calibri_22*
3 - **Verdana_18**
4 - Courier New_21
5 - *Calibri_18*

Selected Pictures:
0 - IMG_LOGO_CLAIRITEC.JPG
1 - IMG_MAIN_SCREEN_OFF.JPG
2 - IMG_VIDEO_SCREEN_OFF.JPG
3 - IMG_BKLGHT_MOINS.PNG
4 - IMG_BKLGHT_PLUS.PNG
5 - IMG_CAM1_ON.PNG
6 - IMG_CAM2_ON.PNG
7 - IMG_CLOSE_OFF.PNG
8 - IMG_CLOSE_ON.PNG
9 - IMG_IHM1_IRONGRAPH.PNG
10 - IMG_IHM2_ESPRESSO.PNG
11 - IMG_IHM3_ESTHETIC.PNG
12 - IMG_IHM4_VIEWER.PNG
13 - IMG_IHM5_LOADING.PNG
14 - IMG_INDEX1_ON.PNG
15 - IMG_INDEX2_ON.PNG
16 - IMG_INDEX3_ON.PNG
17 - IMG_INDEX4_ON.PNG
18 - IMG_INDEX5_ON.PNG
19 - IMG_SKIP_BACK_ON.PNG
20 - IMG_SKIP_ON.PNG
21 - IMG_VIDEO_ON.PNG

Project Screens:
1 - SCREEN01
2 - SCREEN02

background: 0 - IMG_MAIN_SCREEN_OFF

touchpanel layer:
0 - TYPE_TOUCHSCREENZONE [1]
1 - TYPE_TOUCHSCREENZONE [2]
2 - TYPE_TOUCHSCREENZONE [3]
3 - TYPE_TOUCHSCREENZONE [4]
4 - TYPE_TOUCHSCREENZONE [5]

Memory Used: 6 %

Card : IronGraph_RS232 Memory Size : 32Mo LCD : ORTUSTECH Reference : COM43H4M10XTC Width (pixels) : 480 Height (pixels) : 272

The text zone on the foreground layer (red rectangle), the touch zones (green rectangles), placed over the touchable buttons, as well as the first image in the center are clearly visible.

All other elements should be displayed in a dynamic way (this means getting displayed only upon their activation through a specific command). For this reason, they have been turned “invisible”/“inactive”.

However, all elements (active or inactive) are saved in the HMI board with their respective characteristics and coordinates. This allows the retrieval of all necessary parameters for the dynamic way of displaying the hidden elements by using the respective commands.

Extract of the file IRG_RS232_WVGA.h for this HMI screen:

(Next page …)

```
/*-----------------------------------*/
/*      -  HMI EDITOR HEADER -       */
/*-----------------------------------*/

/*****  SCREEN01  ****/
#define SCREEN01 1

/*-------- BACKGROUND LAYER --------*/

/* FONTS */


/* ZONES OF TEXT MODE */


/* PICTURES */

/* --- PICTURE IMG_MAIN_SCREEN_OFF N° 0 --- */
#define SCREEN01_BACKGROUND_IMG_0    1,0
#define XY_SCREEN01_BACKGROUND_IMG_0     0,0
#define X_SCREEN01_BACKGROUND_IMG_0  0
#define Y_SCREEN01_BACKGROUND_IMG_0  0
#define W_SCREEN01_BACKGROUND_IMG_0  800
#define H_SCREEN01_BACKGROUND_IMG_0  480

/* DRAWING PRIMITIVES */


/*-------- FOREGROUND LAYER --------*/


/* FONTS */

/* --- FONT ARIAL_30 N° 0 --- */
/*Colors of text*/
#define BACKCOLOR_SCREEN01_FOREGROUND_TXT_0  0,255, 255, 255
#define FORECOLOR_SCREEN01_FOREGROUND_TXT_0  255,0, 0, 0
/*Style*/
#define STYLE_SCREEN01_FOREGROUND_TXT_0  0,0,0,False,0
/*String of text*/
#define SCREEN01_FOREGROUND_TXT_0    "IMG_IHM_IRONGRAPH"
/*Coordinates*/
#define XY_SCREEN01_FOREGROUND_TXT_0     231,39
```

Etc…

```
/* ZONES OF TEXT MODE */

/* --- ZONE OF TEXT N° 0 --- */
#define SCREEN01_FOREGROUND_ZOT_0    0, HCENTER, VCENTER, False, 0
#define XY_SCREEN01_FOREGROUND_ZOT_0     168, 36, 633, 86

/* PICTURES */

/* --- PICTURE IMG_SKIP_BACK_ON  - [T] 1 N° 6 --- */
#define SCREEN01_FOREGROUND_IMG_6    19,0
#define XY_SCREEN01_FOREGROUND_IMG_6     21,191
#define X_SCREEN01_FOREGROUND_IMG_6  21
#define Y_SCREEN01_FOREGROUND_IMG_6  191
#define W_SCREEN01_FOREGROUND_IMG_6  133
#define H_SCREEN01_FOREGROUND_IMG_6  133
/* --- PICTURE IMG_SKIP_ON  - [T] 2 N° 7 --- */
#define SCREEN01_FOREGROUND_IMG_7    20,0
#define XY_SCREEN01_FOREGROUND_IMG_7     644,191
#define X_SCREEN01_FOREGROUND_IMG_7  644
#define Y_SCREEN01_FOREGROUND_IMG_7  191
#define W_SCREEN01_FOREGROUND_IMG_7  133
#define H_SCREEN01_FOREGROUND_IMG_7  133
/* --- PICTURE IMG_BKLGHT_PLUS  - [T] 3 N° 8 --- */
#define SCREEN01_FOREGROUND_IMG_8    4,0
#define XY_SCREEN01_FOREGROUND_IMG_8     9,369
#define X_SCREEN01_FOREGROUND_IMG_8  9
#define Y_SCREEN01_FOREGROUND_IMG_8  369
#define W_SCREEN01_FOREGROUND_IMG_8  55
#define H_SCREEN01_FOREGROUND_IMG_8  54
/* --- PICTURE IMG_BKLGHT_MOINS  - [T] 4 N° 9 --- */
#define SCREEN01_FOREGROUND_IMG_9    3,0
#define XY_SCREEN01_FOREGROUND_IMG_9     11,422
#define X_SCREEN01_FOREGROUND_IMG_9  11
#define Y_SCREEN01_FOREGROUND_IMG_9  422
#define W_SCREEN01_FOREGROUND_IMG_9  55
#define H_SCREEN01_FOREGROUND_IMG_9  54
/* --- PICTURE IMG_VIDEO_ON  - [T] 5 N° 10 --- */
#define SCREEN01_FOREGROUND_IMG_10   21,0
#define XY_SCREEN01_FOREGROUND_IMG_10    619,394
#define X_SCREEN01_FOREGROUND_IMG_10     619
#define Y_SCREEN01_FOREGROUND_IMG_10     394
#define W_SCREEN01_FOREGROUND_IMG_10     64
#define H_SCREEN01_FOREGROUND_IMG_10     64
```

Etc…

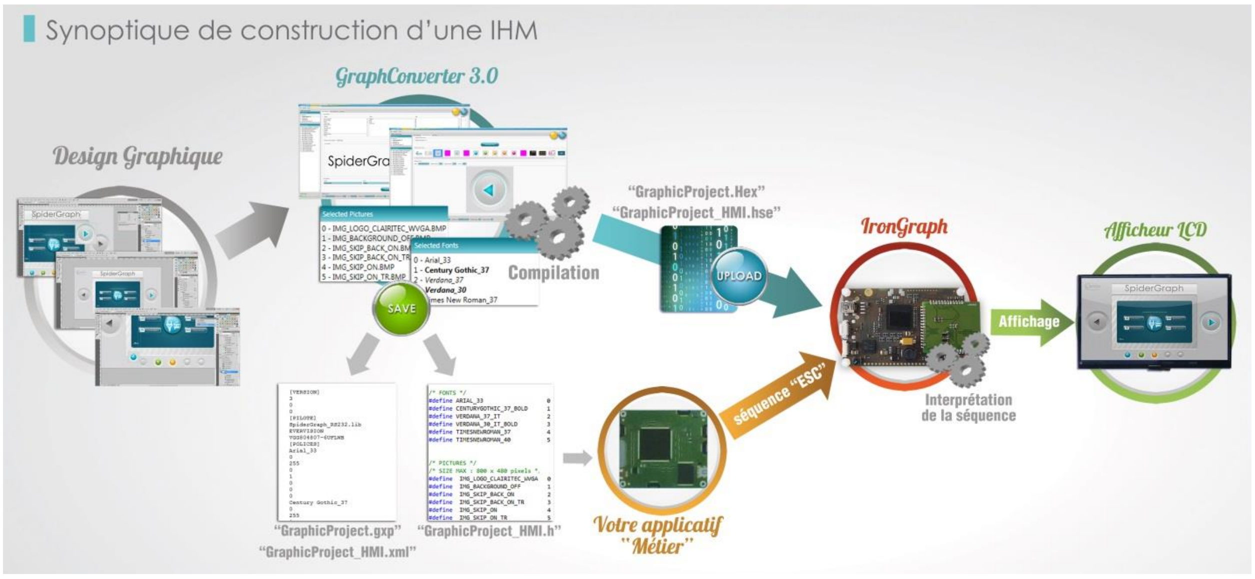
Extract of the code which displays the HMI screen SCREEN01 in its initial status and then displays the text in the pre-defined text zone.

(Next page …)

```
// Displaying the HMI screen in its initial status
GX_PutScreen(RESERVED, SCREEN01);
//Setting the background color of the text as defined in the HMI editor
GX_SetColor( RESERVED, BACK_COLOR, BACKCOLOR_SCREEN01_FOREGROUND_TXT_0);
//Setting the foreground color of the text
GX_SetColor( RESERVED, FORE_COLOR, FORECOLOR_SCREEN01_FOREGROUND_TXT_0);
//Displaying the text in the pre-defined text zone
GX_PutString (RESERVED,STYLE_SCREEN01_FOREGROUND_TXT_0,
ALIGNMENT_ON,NUM_SCREEN01_FOREGROUND_ZOT_0,0, (unsigned char*)SCREEN01_FOREGROUND_TXT_0);
```

The code in bold and blue indicates the #define which can be found in the .h file, created by the HMI editor.

Chapter 13: Conclusion



This schema resumes the mechanism of the CLAIRITEC concept.

For more details about the HMI boards, the Intelligent Displays, the command protocol, or the usage of GraphConverter and the HMI editor, consult the following documents:

- **SpecificationHardware.pdf** for the documentation about the hardware (dependent on the HMI board/Intelligent Display used).
- **FunctionsInC.pdf** for the API (Application Programming Interface) for the C functions.
- **RS232_Protocol.pdf** for the plain code of the RS232 Escape commands.
- **CAN_Protocol.pdf** for the plain code of the CAN Escape commands.
- **UserManual_GraphConverter.pdf** for the way of using GraphConverter and the HMI editor.

Thank you for your attention!

Chapter 14: Technical support

Clairitec

CLAIRITEC
11 avenue Henri Becquerel
33700 Mérignac
FRANCE

Web site: www.clairitec.com

Clairitec's services

Customer relation service: contact@clairitec.com

Technical support service: support@clairitec.com afaq ISO 9001 Qualité AFNOR CERTIFICATION

MADE IN FRANCE

CE

CERTIFIED

Clairitec

11, avenue Henri Bequerel - 33700 - MERIGNAC - FRANCE

Email: contact@clairitec.com
www.clairitec.com