T6963C Display Drive Version 1:41

Click Here to see a Video Demo

proposal

The T6963C chip is the old display driver already 19 years of existence used by microprocessors 8080.8088, z80,6502, etc., but that is still used and even a few implementations for microcontrollers from Atmel AVR indeed was the first thing I did when I met AVR assembler i did the drive to T6963C as learning exercise and it ok on the first run. Despite being a driver for monochrome displays the controller has some interesting features, such as simultaneous display of text and graphics may even make logical operation between the text and graphic such OR, AND, EXOR also allowing the use of attributes for text, usually This controller has enough memory to hold pages of text and video pages also has commands to set Pixel and writing and reading rapid with possibility of increment, decrement or no change in the memory position after reading or writing.

This Version 1:41 I'm releasing was done in C ++ using the Atmel Studio 7.0, in order to provide migration to Arduino boards with little change in the code. And assigned to at least 160x128 pixels Displays only for the execution of the Demo, but the drive is generic and can be used for any resolution. This drive during startup, is an interface check and checks as there is memory, thus allowing to drive implementation of two text pages and two graphic pages if available memory allows, and if there is not enough memory the drive uses minimal 1 text page and one graph page.

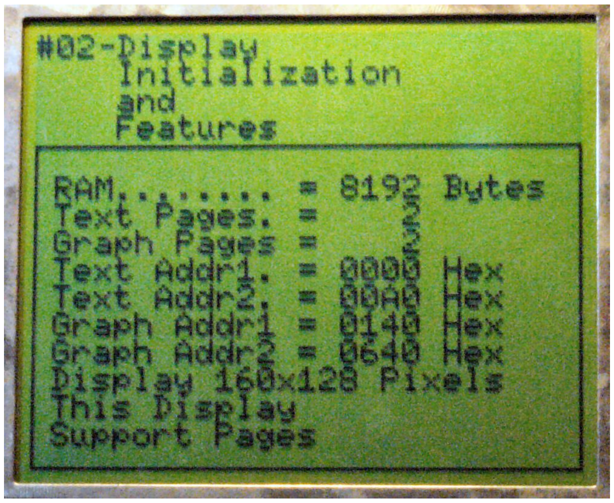
Implementation

This drive is designed to operate in microcontrollers from Atmel AVR ATmega and the current Demo operates in AVRs ATMega8,64,128,162,328P and is easily converted to other ATMegas with little change on the same code to Arduino.This limitation refers only to Demo having no limitation due to the drive for use on any AVR. The demo was done using an ATmega128 with 16Mhz crystal follows:.

This Version 1:41 has some interesting features, as

- **Check presence of the LDC -** Usually LCD drivers are initialized, assuming that it is present on the doors or the microcontroller bus in use or microprocessor and can thus cause problems, such as locking, incorrect printing characters, etc. With this check can start the display and the drive is responsible for returning the state of their presence, hence enabling an error processing with eg warning through a beep, an LED flash etc. This drive configuration and informed besides the display of pins and a pin and port to connect an LED so that it flashes if the interface of any problems during startup size.

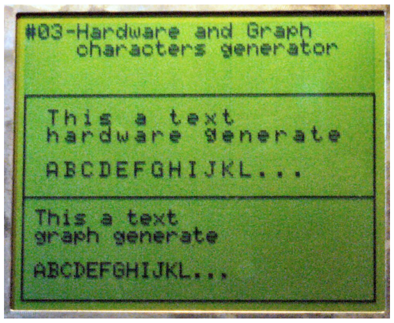
- **Start with checking the memory** - During startup the drive calculates the amount of memory and configures the set to contain one page of text and graphics or 2 pages of text or graphic, when the existence of two pages this facilitates in the case of ha graphics make animations with plenty reduction "flicking" image.



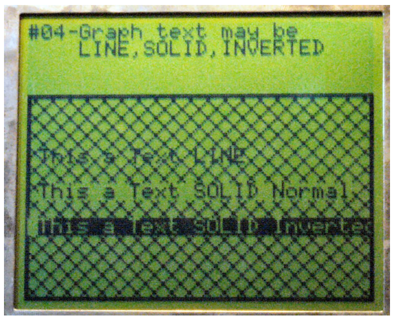
```
#02-Display
    Initialization
    and
    Features

RAM........ = 8192 Bytes
Text Pages. =    2
Graph Pages =    2
Text Addr1. = 0000 Hex
Text Addr2. = 00A0 Hex
Graph Addr1 = 0140 Hex
Graph Addr2 = 0640 Hex
Display 160x128 Pixels
This Display
Support Pages
```

Startup page on the Demo,-.

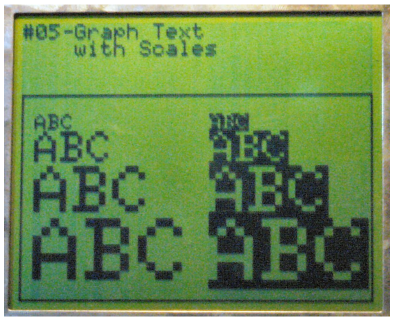
- **use texts in native or mode** graphics The T6963c has an internal character generator which is used in Text mode for text functions and also graphic text generated by software, when in graphical mode there is a possibility scale the text expanding its size by 2, 3, 4 etc. times



```
#03-Hardware and Graph
    characters generator

This a text
hardware generate
ABCDEFGHIJKL...

This a text
graph generate
ABCDEFGHIJKL...
```



```
#04-Graph text may be
    LINE,SOLID,INVERTED

This a Text LINE
This a Text SOLID Normal
This a Text SOLID Inverte
```



```
#05-Graph Text
    with Scales

ABC
ABC
ABC
ABC
```

Text generated by hardware and Text generated Software

Graphics Text Only Line, Solid Inverted

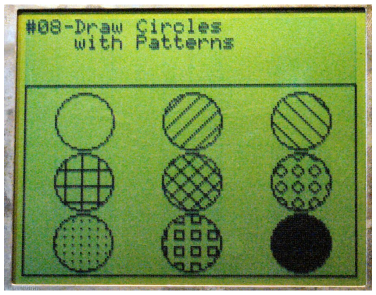
text graphic on the scales 1,2,3 and 4

T6963C Display Drive Version 1:41

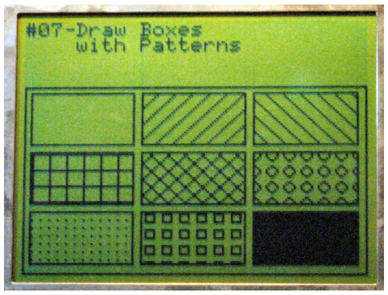
- **String Functions display that accept CR and LF** - facilitating the use of these functions (putc, puts, PutSF, GPutC, GPutS, GPutSF) since it can generate a single String and write in one command into two lineslarge.
- **BitMap of Use -** Functions for handling bitmap memory until 32kbyte sizes are admitted into this drive even if the graphic display is smaller than the Bitmap and may even draw bitmaps that are not fully inserted in the display screen, allowing animations where a bitmap enters through the right side, left, above or below the display-.

maximumsize BitMap 32Kbytes

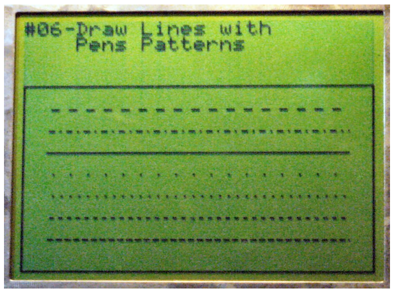
- **Line Drawing, squares and circles** These functions have the ability to be filled with patterns (squares and circles) with or without borders and types of penalties for drawing lines. All these functions were very optimized when it comes to speed.



Filled circles with patterns



squares filled with patterns



lines drawn with types of feathers

- **Quick Horizontal Line Drawing**.- The design of horizontal lines uses hardware feature that allows your stroke fastest way possible-.
- **Drawing in defined window** Drawing functions will be drawn only within the defined window, allowing the isolation of the graphic part to a particular region of the screen.

T6963C Display Drive Version 1:41

Implemented features.

Class class_GraphBasic

private functions

void Initialize (void)
Initializes and graphic ingenuity.

Void UnInitialize(void)
UnInitialization of graphic device.

Void PlotP8 (int xx, int yy, int x, int y, Pen pen, pixel pixel)
function used internally by DrawCircle function to draw on the screen 8 symmetrically or fill with horizontal lines.

void DrawBoxFull(int xi, yi int, int xf, yf int, Pen pen)
Draws a filled by pen square. Starting at the coordinate origin xi, yi and ending in xf, yf.

Public Functions.

Class_GraphBasic (int Rows, int Cols, int GraphWidth, int Initializes)
GraphHeight the graphics engine providing the numbers of hardware lines defined by pin (Rows) the number of columns defined by hardware (Cols) the width in pixels of the graph (GraphWidth) and height in pixels of the graph (GraphHeight).

virtual void DrawHorzLine (int y, int xs, int xE, Pen pen = Pen :: Black)
Draw a horizontal line beginning at the Y ordinate and abscissa xS initial and final abscissa xE with Pena (pen) if not provided by the function with the pen black.This function is virtual, so it will be implemented in the T6963C Drive (derived class)

virtual void SetPixel (int x, int y, Pixel pixel = Draws)
1)Set pixel at x, y coordinate to pixel color (0 or 1) and (1) = black, if not supplied this parameter is assumed pixel = 1. This function is virtual, so it will be implemented in the T6963C Drive (derived Virtual

class)Pixel GetPixel (int x, int y)

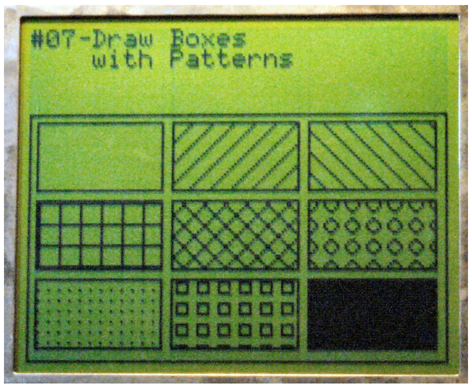
Le pixel status at coordinates display x, y, returning a pixel (0.1) if (1) = black. This function is virtual, so it will be implemented in T6963C Drive (derived class).

void DrawVertLine(int x, int y i, int yf, Pen pen = Pen :: Black)

Draw a vertical line starting at the abscissa x initial and final ordinate yi yf with pen (Pen) which is not provided assumes worth = black.

void DrawBox(int xi, yi int, int xf, yf int, Pen pen = Pen :: Black, bool border = true)

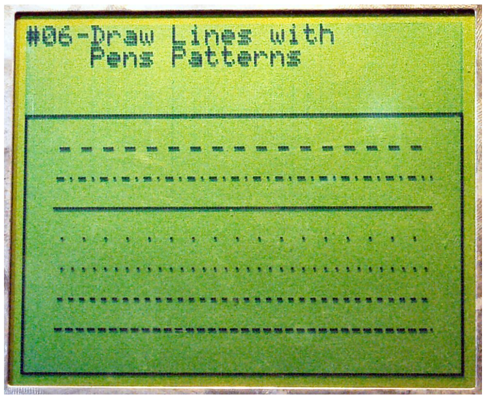
Draw a square starting at the initial coordinates xi, yi and ending the XF coordinates yf with the pen (pen) or not supplied pen = black bordered to "border = true", the DrawBox and modified by SetDrawMode function that can set the fill square if DrawMode = Solid will be drawn a square with black or white fill and DrawMode = Pattern will be drawn a square filled with a pattern that is set to the default SetDrawPatternType.This function can be set by the user. See image below.



DrawBox design standards

void DrawLine(int xi, yi int, int xf, yf int, Pen pen = Pen :: Black)

Draws a line starting at the coordinates xi, yi and ending the coordinates xf, yf if (pen) is not provided will be used = black. The pen can be defined by the user.



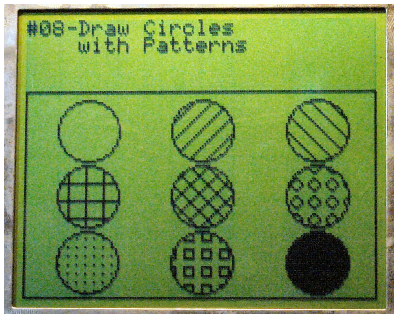
Pen types used by the function DrawLine

void DrawLineTo (int x, int y, Pen pen = Pen :: Black)

Does the same as the DrawLine function except that the initial coordinate will be the last that was set by a command for drawing x, y defines the final coordinated the line will be drawn with the pen (pen) or omitted will be used to pen = black.

void DrawCircle(int x, int y, int radius, Pen Pen Pen :: :: Black, Pixel pixel = 1, bool border = true)

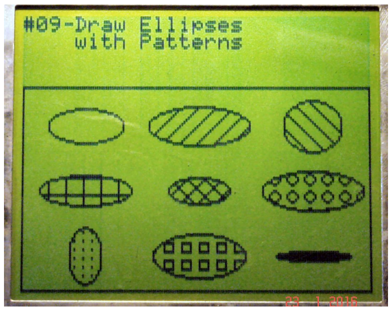
Draws a circle centered on the initial coordinates x, y with a radius (radius) with the pen (pen) or if not provided with penalty = black edge is "border = true "the DrawCircle and modified by the function SetDrawMode you can set the circle filling if DrawMode = Solid will be drawn a square with filling pen (pen) and DrawMode = Pattern will be drawn a square filled with a pattern that is defined with the SetDrawPatternType.O standard function can be set by the user. See image below.



Fills Standards for DrawCircle function

void DrawEllipse (int x, int y, int rx, ry int, Pen Pen Pen :: :: Black, Pixel pixel = 1, bool border = true)

Draws a circle centered at the coordinates Initial x, y with a radius (radius) with the pen (pen) or not supplied pen = black bordered to "border = true, the DrawCircle and modified by the function SetDrawMode you can set the circle filling if DrawMode = Solid is designed to fill a ellipse pen (pen) and DrawMode = Pattern will be drawn a ellipse filled with a pattern that is set to the default SetDrawPatternType.O function can be set by the user. See image below:.



Function fills Standards for DrawEllipse

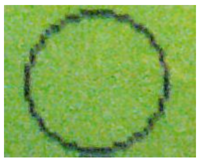
void SetDrawMode (DrawMode DrawMode)

Sets the drawing mode for DrawBox and DrawCircle functions you

DrawMode :: Line

In this mode is only drawn the outline DrawBox the boundary will be defined by type Pen (Pen) and DrawCircle will be defined by the pixel (0,1).

rectangle contour

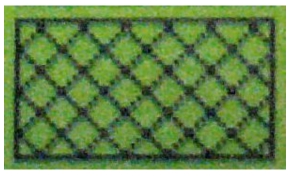
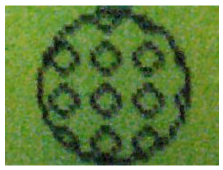


circle contour

DrawMode :: Solid
In this mode is designed completely filled by pixel (0.1)

rectangle filled circle filled

DrawMode :: Pattern
In this mode is designed completely filled by a pattern (Pattern) which may be a pattern already set or a new user-defined.

retando with pattern circle with standard

DrawMode GetDrawMode (void)
Returns the current drawing mode.

uint8_t VersionMajor Returns
(void)version value drive major change.

uint8_t VersionMinor (void)
Return value version of the drive change smaller.

class class_T6963C_Drive_V141

private functions

bool InterfaceInitialize (uint8_t hardwareRows, uint8_t hardwareCols, uint8_t displayCols, _T6963C_ScanMode scanMode)

Initialize the T6963C interface and returns true if initialization is successful.HardwareRows and hardwareCols are preset at the factory pins and represents the amount of rows and columns that the display in question has usually these values are the width and height in display in pixels in question divided by 8 in this demo in question use a display of 160x128 pixels that is 160/8 = 20 columns and 128/8 = 16 lines. displayCols represents the number of columns that will be displayed on the screen may differ from hardwareCols. scanMode is defined by manufactures also can be SINGLE or DUAL, since some controllers use only one (single) controller for the entire display, and others may use up to 2 (DUAL).

uint8_t InterfaceRead (void)

Prepare the T6963C interface for reading and reads the data interface.

void InterfaceWrite (void)

prepares the T6963C interface for recording and writes the data interface

uint8_t InterfaceStatusRead(void)

Read current state T6963C

uint8_t InterfaceDataRead (void)

Read the data of T6963C

void InterfaceCmdWrite (uint8_t commandToBeWritten)

Write command on T6963C

void InterfaceDataWrite (uint8_t dataToBeWritten)

Write data in T6963C

void CanReadWrite (void)

Waits T6963C enter the state where writes and reads are allowed.

void CanAutoWrite (void)

Waits T6963C enter the state where fast recordings are allowed.

void CanAutoRead (void)
Waits T6963C enter the state where fast reads are allowed.

void Reset (void)
Resets the T6963C chip.

bool CheckRAMPosition (uint16_t RAMAddr)
Check T6963C RAM recording of presence at the address RAMAddr the 0x00,0x55,0xaa values and waits the return in the same sequence. Returns true if the data was recorded and returned ok.

Uint16_t GetComputedRAMSize (void)
Returns the amount of RAM present in T6963C interface

void DrawPatternWithMask (uint8_t mask, uint8_t pattern, Pen pen, bool autoWrite = Writes=false)
Write the RAM one pattern is DrawMode = Pattern or Pen is DrawMode = SOLID and performs a read and write using a mask (mask) where the bits = 1 of the mask represents the bits will be changed and if true autoWrite will be a fast writing, in this case SetModeAutoWrite function should be performed before calling this function.

Public Functions

class_T6963C_Drive_V141 (uint8_t hardwareRow, uint8_t hardwareCols, uint8_t displayCols, _T6963C_ScanMode scanMode)
See InterfaceInitialize function.

void SetUpdateCursorPositionState (bool updateState)
If updateState = true the putc function, puts, PutSF after character sented to display the text cursor is placed in this position where the character was recorded.

bool Present (void)
This function returns true if the interface is initialized ok.

void DataWrite (uint8_t *dataToBeWritten)*
This the master who actually writes in chip T6963C, always use this function to send data to the interface.

uint8_t DataRead(void)

This the master who actually reads a given the T6963C chip, always use this function to read data to the interface.

void CmdWrite (uint8_t commandToBeWritten)

This the master who actually writes a command on the T6963C chip, always use this function to send commands to the interface.

void DataAutoWrite (uint8_t dataToBeWritten)

This function allows you to write in T6963C more quickly, for that call SetModeAutoWrite function before and after its execution only DataAutoWrite ,SetModeAutoReset commands will be allowed. Always after using a DataAutoWrite command block at the end call SetModeAutoReset to return the interface to normal operating mode.

Uint8_t DataAutoRead(void)

This functions is similar to DataAutoWrite in operation changing only that it is used for quick reading.

Void SetTextAddress (uint16_t textAddress)

Set memory address where the character buffer that will be shown on the display in text mode. The codes written to this buffer is not ASCII, it is actually moved from 32 bytes starting with " " (space) 0x00 code.

Void SetTextArea (uint8_t displayCols)

Defines how many columns are visible in the display in text mode. Be very careful using this function, please read the datasheet of the T6963C for more details.

Void SetGraphAddres (uint16_t graphAddr)

Set memory address where the graphics buffer bytes will be shown on the display in graphical mode. 0xff recorded in a position represents a row of 8 pixels black.

Void SetGraphArea (uint8_t displayCols)

Defines how many columns are visible on the display in graphical mode. Be very careful using this function, please read the datasheet of the T6963C for more details.

Void SetAddressPointer (uint16_t addressPointer)
Places the pointer at the address = addressPointer, this address after set will be the memory position where data write or reading.

void SetDisplayMode (_T6963C_DisplayModes displayMode, bool This function
status)enables or disables the display modes, the modes são:

_T6963C_DisplayModes::_GRAPH
_T6963C_DisplayModes::_TEXT
_T6963C_DisplayModes::_CURSOR
_T6963C_DisplayModes::_BLINK

Example: if you want to disable the graphics display execute the following command

```
SetDisplayMode (_T6963C_DisplayModes :: _ GRAPH, false);
```

void SetCursorPos (uint8_t Row, uint8_t Col)
Sets the position where the text mode cursor appears, the cursor is visible only if the SetDisplayMode :: _ CURSOR is enabled.

void SetCGRAMOffset (uint16_t cgramAddress)
Sets the address T6963C of external memory which starts the external character generator, the cgramAddress are the bits 11-15 of the address, look at the T6963C manual for details.

void SetModeAutoWrite(void)
Puts the T6963C in the fast recording mode after execution of this command only the DataAutoWrite and SetModeAutoReset functions will be aceitos.Sempre at the end of the function call SetModeAutoReset for the T6963C interface accepts other commands.

void SetModeAutoReset(void)
Restores normal operation of the T6963C interface after execution of commands SetModeAutoWrite, SetModeAutoRead.

void SetModeAutoRead(void)
Puts the T6963C in fast read mode after execution of this command only DataAutoRead and SetModeAutoReset functions will be aceitos.Sempre at the end of the function call to SetModeAutoReset the T6963C interface accepts other commands.

void SetCursorSize (uint8_t CursorSize)
Sets the cursor size in number of lines starting from the character base to the top.

void FillRam (uint16_t startAddress, uint16_t fillSize, uint8_t Fills
patternByte)memory region started in address = startAddress size = filSize with byte = patternByte.

void SetClsTextChar (char character)
defines the character that will fill the screen when the execution of the function ClsText

void ClsText (void)
Clears the text mode screen with the defined character in SetClsTextChar

void ClsGraph (void)
Clears the Graph Mode Screen.

void DataWriteInc (uint8_t DataValue)
Write data to the external memory of T6963C and increments the pointer after recording.

void DataWriteDec (uint8_t DataValue)
Write data to the external memory of T6963C and decreases the pointer after recording.

void DataWriteNoMove (uint8_t DataValue)
Write data to the external memory of T6963C and does not change the pointer.

void DataReadInc (uint8_t DataValue)
Read the external memory of T6963C and increments the pointer after reading.

void DataRead (uint8_t DataValue)
Read in external memory of T6963C and decreases the pointer after reading.

void DataReadNoMove (uint8_t DataValue)
Read the external memory of T6963C and does not change the pointer.

void SetOrMode(void)
Set logic to be executed between the text and part display spelling with loggia OR.

void SetEXorMode(void)
Set logic to be executed between the part text and part display spelling with loggia EXOR.

void SetAndMode(void)
Set logic to be executed between the text and part spelling of the display with loggia DNA.

void SetInternalCGROM(void)
Set T6963C of the character generator to the internal ROM.

void SetExternalCGRAM(void)
Set T6963C of the character generator for Foreign RAM.

uint8_t inline VersionMajor(void)
Return the Drive Version Number T6963C "Major change".

uint8_t inline VersionMinor(void)
Return Drive Version Number T6963C "Minor change"

bool LocateXY(int x, int y)
Places the pointer in the graphic area defined by the graph coordinates x, y.Case x, y is outside the current window (see SetWindow, GetWindow) returns false.

virtual void SetPixel (int x, int y, Pixel pixel = 1)
on or off pixel (1 = black) at coordinates x, y. If none is specified pixel is assumed value 1

Pixel GetPixel (int x, int y)
Read pixel coordinates x, y. Returning 1 or 0.

bool LocateRC (int8_t Row, Col int8_t)
Positions the pointer in the Text area defined by coordinates Line = Row, Column = Col. If Row, Col is outside the current window returns false.

Void putc (char charASCII)
Send to display text area an ASCII character

Void puts (char const * charString)
Send to display the text area a character string ASCII originating in RAM.

void PutSF(const char * charString)
Send to display the text area an ASCII character string originating in Flash. Can used as

PutSF (PSTR ("ABCD"));

Send string "ABCD" of Flash to display the text area

int inline GraphWidth(void)
Return width in pixels of the graphical display area

int inline GraphHeight(void)
Return height in pixels of the graphical display area.

int inline GraphWidth(void)
Return width in pixels of the graphical display area.

int inline GraphHeight(void)
Return Height in pixels of the graphical display area.

uint8_t inline TextWidth ()
Returns number of columns in characters Display the text area.

uint8_t inline TextHeight ()
returns number of rows in characters Display the text area.

uint16_t inline GetTextAddress(void)
Return memory address where it starts the Text 1 buffer text area .

uint16_t inline GetText2Address(void)
Return memory address where it starts the Text 2 buffer text area.
This value will be valid only if more than one (1) page of text.

uint16_t inline GetGraphAddress(void)
Return memory address where it starts the graphic buffer 1 of the graphic area.

uint16_t inlineGetGraph2Address(void)
Return memory address where it starts the graphic buffer 2 of the drawing area.
This value will be valid only if more than one (1) graphic page.

uint16_t inline GetCGRAMAddress ()
Returns memory address where it starts the external character generator.

uint16_t inline GetTextPages ()
returns the number of pages of text.

uint16_t inline GetGraphPages ()
returns the number of graphic pages.

void SetTextActivePage (uint8_t ActivePage)
Sets the text page which will be active, where the text of commands will be executed.

void SetTextVisiblePage (uint8_t visiblePage)
Sets the text page that is visible on the display.

void SetGraphActivePage (uint8_t ActivePage)
Sets the graphical page that is active where the graphics commands are executed.

void SetGraphVisiblePage (uint8_t visiblePage)
Sets the graphical page that is visible on the display.

void BlinkLEDError (uint16_t blinkTimeMs)
routine that will flash the LED on blinkTimeMs time if the display is not initialized properly.

void DrawHorzLine (int y, int xs, int xE, Pen pen = Pen ::Black)

Draw Horizontal Line in fast mode started on the ordinate Y and initial abscissa xS and final xE.

void GetDrawPattern (_T6963C_Pattern &pattern)

Return Pattern actually used in graph mode. Returned pattern may be:

EMPYT,
SLASH,
BACKSLASH,
GRID,
DIAGONAL_GRID,
CIRCLE,
POINT,
BOX,
SOLID

void SetDrawPattern (_T6963C_Pattern &pattern)

Set a new the graph pattern.The patterns may be:

EMPYT,
SLASH,
BACKSLASH,
GRID,
DIAGONAL_GRID,
CIRCLE,
POINT,
BOX,
SOLID

void GetDrawPatternType(_T6963C_Patterns patternType, _T6963C_Pattern & patternToGet)

Return a graphic pattern provided by patternType.

void SetDrawPatternType (_T6963C_Patterns patternType)

Define a new graphic pattern provided by patternType. Patterns may be;

EMPYT,
SLASH,
BACKSLASH,
GRID,
DIAGONAL_GRID,
CIRCLE,
POINT,
BOX,
SOLID

void SetWindow (int xMin, int yMin, int xMax, int yMax)

Set Graphic window where the graphics commands are displayed, started at coordinates xMin, yMin and finished xMax, yMax. If the supplied values are higher or lower than the default coordinates Display the default values are assumed.

Void GetWindow (int & xMin, int & yMin, int & xMax, int &yMax)

Return the window coordinates currently in use.

Bool DrawBitMap (int x , int y, BitMap *bmp)

Draw BitMap the initial x, y coordinates representing the upper left of the bitmap on the display. It accepts lower or higher coordinates defined in SetWindows. If the coordinate is partially off the screen, only the visible part of BitMap will be drawn. Returns true valid if BitMap drawed ok. A _BMP_To_Hex name app is provided for converting images .bmp, .jpg, .pgn, .gif for the BitMap type in C located on the main page Drive.

```
void    GetBMPParms(BitMap        *bmp,
                                  uint16_t    &bmpHeader,
                                  int16_t     &bitDeep,
                                  int16_t     &widthPixel,
                                  int16_t     &heightPixel,
                                  uint16_t    &widthByte,
                                  uint16_t    &sizeByte);
```

Retorn the bmp parameters provided, which
bmpHeader = "BM"
bitDeep = 1
widthPixel = BitMap width in pixels
heightPixel = BitMap height in pixels
widthByte = BitMap width in bytes
sizeByte = BitMap size in bytes

int GetBMPWidth (BitMap * Returns
bmp)width BitMap = bmp in pixels.

int GetBMPHeight (BitMap *bmp)
Return height BitMap = bmp in pixels.

Pixel GetBMPPixel (BitMap * bmp, int x, int y)
Get pixel at coordinates x, y BitMap = bmp

void GPutC (char c, pixel pixel = 1)
Send character c = for the graphic part of the display. If not given pixel is assumed the value 1 (black).

void GPutS(const char * charString, Pixel pixel = 1)
Send chair character = charString originating from the RAM to the graphic part of the display. if not given pixel is assumed 1.

void GPutSF(const char * charString, Pixel pixel = 1)
Send chair character = charString originating from the FLASH to the graphic part of the display. if not given pixel is assumed 1.

uint8_t inline GetTextCharWidth(void)
Return width of the character in text mode in pixels.

uint8_t inline GetTextCharHeight(void)
Return character height in text mode in pixels.

uint8_t inline GetGraphCharWidth(void)
Return width character in the graphics mode in pixels.

uint8_t inline GetGraphCharHeigh(void)
Return character height in pixels in graphics mode.

void inline SetGraphTextScale (uint8_t newScale)
Defines graphic scale for the text graphics mode. 1,2,3 only integer values. etc.

uint8_t inline GetGraphTextScale(void)
Return current graphics scale graphic text mode.