Quick Reference Guide

September 2011

Graphics Quick Reference Guide

www.microchip.com/graphics

Display Controller Solutions – Graphical Displays

Microchip offers varying levels of solutions to drive everything from simple monochrome LCDs to full color WVGA user interfaces.

Graphics support includes the following approaches:

- PIC24F DA integrated graphics controller
- PIC32 controllerless graphics
- Support for PIC MCU with external graphics controllers

The silicon offering is complemented with powerful, free and easy to use graphics library, display designer GUI and hardware development kits with flexible interface to various glass sizes.

Supported Screen Sizes and Colors

Microchip graphics solutions support various screen sizes and colors ranging from small monochrome OLED displays up to WVGA displays with vivid color. The table below shows the bits per pixel required to represent color.

Display Representation	Color Examples	Color Depth (bits per pixel)
Mono	Black and White	1
Grayscale	4 shades 16 shades	2 4
Color	256 colors 65K colors 16 million colors	8 16 24

As the color depth and display resolution increase, the frame buffer grows. Depending on the size, the frame buffer can be stored in the microcontroller RAM, in external SRAM or integrated into an external graphics controller.

The table below shows examples of the frame buffer sizes required for some popular resolution and color depths.

- PIC24 DA family supports up to 96 KB on chip
- PIC32 MCUs support up to 128 KB on chip
- External SRAM can be used for larger frame buffers
- For advanced graphics, external graphics controllers have additional frame buffer storage

Display Resolution Typical Sizes			Color Depth/ Memory Requirement in (bytes)			
			1 bpp (Mono)	2 bpp (4 shades)	8 bpp (256 colors)	16 bpp (65K colors)
WVGA	800x480	7″	48,000	96,000	384,000	768,000
VGA	640x480	5.7″	38,400	76,800	307,200	614,400
WQVGA	480x272	4.3″	16,320	32,640	130,560	261,120
QVGA	320x240	3.2″	9,600	19,200	76,800	153,600
Common for OLED	128x64	1″-2.7″	1,024	2,048	8,192	16,384

Internal SRAM on PIC24DA or PIC32 MCU External SRAM

Target Applications

Applications that benefit from attractive and easy to use graphical displays include:

Consumer: Thermostats, Cordless Phones, Remote Controls

Home Appliance: Coffee Makers, Washing Machines, Ovens

Industrial: Digital Instrument Gauges, Storage Controls, Remote Terminals

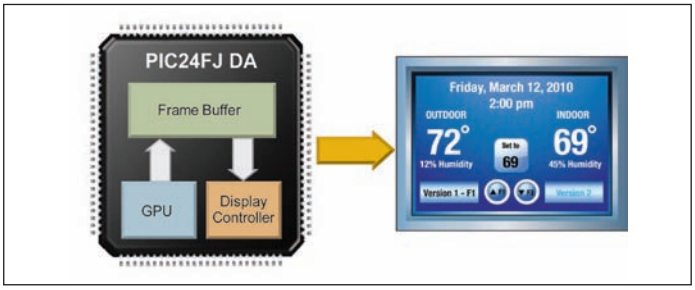
Portable Medical: Glucometers, Blood-Pressure Monitors, Portable ECGs

	PIC24 DA Integrated Graphics Controller	PIC32 Controllerless Graphics	External Solomon Systech Graphics Controller SSD1926	External Epson Graphics Controller S1D13517
Display*	WQVGA 480x272	WQVGA 480x272	WQVGA 480x272	WVGA 800x480
Graphics	HW Acceleration: Rectangles, Characters, Images	DMA on PIC32 + <5 MIPS	HW Acceleration, SD card, I/F, JPEG engine	SDRAM, I/F, Alpha-blending, Picture-in-picture
Frame Buffer	Color Lookup Table + 96 KB on MCU + Ext SRAM	128 KB on MCU + Ext SRAM	256 KB on Solomon Systech Controller	Ext SDRAM
Core MIPS	16	80	–	–
Power	Better	Good	Good	Good
Cost	$	$	$$	$$$

*Max size at 16 bpp, 60 Hz

Graphical Display Configurations

PIC24F with Integrated Graphics Controller: Low Cost, Easy to Use

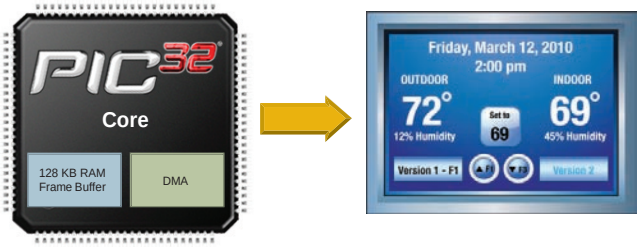


The PIC24F DA family makes it easy and cost-effective to add advanced graphics to your application by eliminating the need for external frame buffers or display controllers.

- Dedicated graphics clock for a continuous, clean display
- On-chip display controller provides direct interface to TFT, STN and OLED displays
- Easy to use Graphics Processing Units for hardware acceleration
  - Move and copy rectangles with smooth, fast memory transfers
  - Decompress images without CPU intervention
  - Render text without CPU intervention
- Color look-up table and 96 KB frame buffer to support multiple colors
  - Supports QVGA 8 bpp with internal frame buffer
  - Supports WQVGA 16 bpp with external frame buffer using PMP (Parallel Master Port)

With the hardware acceleration, this family is able to process and render graphics without using any MCU MIPS. The dedicated graphics engine is able to continuously drive a display without being shared with any other function.

PIC32 Low Cost Controllerless Graphics: 32-bit Performance, Flexibility, Integration



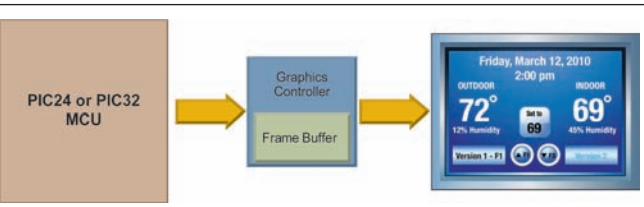
Microchip's PIC32 line of 32-bit microcontrollers offers 80 MIPS and high performance DMA to render graphics directly to displays. This enables PIC32 devices to drive a display without an external graphics controller.

- Uses <5 MIPS and DMA to render graphics
  - Direct interface to STN, TFT displays
- Integrated 128 KB frame buffer
  - Supports QVGA 8 bpp with internal frame buffer
  - Supports WQVGA 16 bpp with external frame buffer using PMP (Parallel Master Port)
- Works with any PIC32 80 MIPS 32-bit microcontroller

With devices offering up to 512 KB Flash and 128 KB RAM, developers have plenty of space for application code, communications stacks and data buffering. In addition to the graphics capabilities, PIC32 MCUs also have integrated peripherals for USB, CAN, Ethernet and capacitive touch sensing.

External Graphics Controller: PIC24 or PIC32 with Parallel Master Port (PMP)

PIC24 and PIC32 MCUs can also work with an external graphics controller to support larger screen sizes or more advanced graphical features.



The Solomon Systech SSD1926 Graphics Controller has hardware graphics acceleration to free up the MIPS of the PIC MCU. This controller includes a SD Card interface and JPEG decode engine as well as 256 KB RAM. The Graphics PICtail™ Plus SSD1926 Board (AC164127-5) includes serial Flash for data storage and interfaces to either Explorer 16 or PIC32 Starter Kits.

The Epson S1D13517 Graphics Controller includes alpha blending, picture-in-picture and supports up to WVGA (800x480) at 24 bpp. This controller has an SDRAM interface for connection to low cost external memory. The Graphics Controller PICtail Plus Epson S1D13517 Board (AC164127-7) includes 128 Mb SDRAM frame buffer and 16 Mb serial Flash and interfaces to either Explorer 16 or PIC32 Starter Kits.

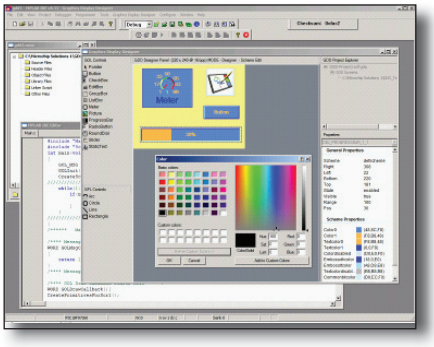
Graphics Library, Designer and App Notes

FREE Microchip Graphics Library

The Microchip Graphics Library is highly modular and is optimized for Microchip's 16- and 32-bit microcontrollers. It is easy to use and has an open documented interface for driver or controller support. The library supports the following features:

- Pre-made graphics objects
- Multiple fonts and languages
- User interface for mTouch™ sensing
- Includes buttons, charts, check boxes, scroll bars, list boxes, images and basic animation

FREE Microchip Graphics Display Designer



The Microchip Graphics Display Designer (GDD) is a visual design tool that provides customers with a quick and easy way of creating Graphical User Interface (GUI) screens for graphical interface applications on Microchip MCUs.

It provides the following advantages to the developers:

- Simplifies coding for the GUI screens with an ability to draw, resize and delete screen objects
- Eliminates the need to manually calculate the X/Y coordinates for on-screen object placements
- Generates output source files
- Ability to import various graphical resources, including custom fonts and bitmap images

Application Notes for Graphical Displays

- *How to Use Widgets in Microchip Graphics Library*, AN1136
- *Fonts in the Microchip Graphics Library*, AN1182
- *How to Create Widgets in Microchip Graphics Library*, AN1246
- *Using a Keyboard with the Microchip Graphics Library*, AN1227
- *Developing Graphics Applications using MCU with Integrated Controller*, AN1368
- *Using PIC32 MCUs to Develop Low-Cost Controllers (LCC) Graphics Solutions*, AN1387

Development Tools for Graphical Display Controllers

Family	Display Boards Supported				
	QVGA 3.2″ Graphics Display Truly 240x320 Board (AC164127-4)	WQVGA 4.3″ Graphics Display Powertip 480x272 Board (AC164127-6)	VGA 5.7″ Graphics Display Truly 640x480 Board (AC164127-8)	WVGA 7″ Graphics Display Truly 800x480 Board (AC164127-9)	Prototype Boards Connect Your Glass (AC164139)
PIC24 DA Family	PIC24FJ256DA210 Board (DM240312) + Display Board		8 bpp or 30 Hz		Yes
PIC32 LCC Graphics	PIC32 Starter Kit (DM320001 or DM320003-2) + LCC Graphics Board (AC164144) + Display Board		8 bpp or 30 Hz		Yes
PIC24 + Solomon Systech SSD1926	Explorer 16 (DM240001) + Solomon Systech GFX Board (AC164127-5) + Display Board				Yes
PIC32 + Solomon Systech SSD1926	PIC32 Starter Kit (DM320001 or DM320003-2) + Multimedia Expansion Board (DM320005) with Integrated Display				No
PIC32 + Solomon Systech SSD1926	PIC32 Starter Kit (DM320001 or DM320003-2) + Solomon Systech GFX Board (AC164127-5) + Display Board				Yes
PIC24 + Epson S1D13517	Explorer 16 (DM240001) + Epson GFX Board (AC164127-7) + Display Board				Yes
PIC32 + Epson S1D13517	PIC32 Starter Kit (DM320001 or DM320003-2 or DM320004) + Epson GFX Board (AC164127-7) + Display Board				Yes

Note: Recommendations based on 16 bpp, 60 Hz performance on PIC MCU LCD Controller.

Tools for Designing Graphical Displays

Microchip Graphics Solutions

This table shows the out-of-the-box support for the following development boards and kits. With proper software and hardware configuration compatibility of certain hardware combinations and other PIC devices can be achieved.

PIC32 Starter Kit Based	PIC32 Starter Kit (DM320001)	PIC32 USB Starter Kit II (DM320003-2)	PIC32 Ethernet Starter Kit (DM320004)	PIC24E USB Starter Kit (DM240012)	dsPIC33E USB Starter Kit (DM330012)	Graphics Display Truly 3.2" 240x320 Board (AC164127-4)	Graphics Display Powertip 4.3" 480x272 Board (AC164127-6)	Graphics Display Truly 5.7" 640x480 Board (AC164127-8)	Graphics Display Truly 7" 800x480 Board (AC164127-9)	Graphics Display Prototype Board (AC164139)
Graphics LCD Controller PICtail™ Plus SSD1926 Board (AC164127-5)	✓	✓	–	✓+	✓+	✓	✓	– (1)	– (1)	✓ (2)
Graphics Controller PICtail™ Plus Epson S1D13517 Board (AC164127-7)	✓	✓	✓	✓+	✓+	✓	✓	✓	✓	✓ (2)
Low-Cost Controllerless (LCC) Graphics PICtail™ Plus Daughter Board (AC164144)	✓	✓	✓	✓+	✓+	✓	✓	✓ (3)	–	✓ (2)
Multi Media Expansion Board (DM320005)	✓	✓	✓	✓	✓	–	–	–	–	–

Development Boards	Explorer 16 Development Board (DM240001) + Plug-In Modules (PIMs)		Graphics Display Truly 3.2" 240x320 Board (AC164127-4)	Graphics Display Powertip 4.3" 480x272 Board (AC164127-6)	Graphics Display Truly 5.7" 640x480 Board (AC164127-8)	Graphics Display Truly 7" 800x480 Board (AC164127-9)	Graphics Display Prototype Board (AC164139)
PIC24FJ256DA210 Development Board (DM240312)			✓ (4)	✓ (5)	✓ (3)	–	✓ (2)
Graphics LCD Controller PICtail™ Plus SSD1926 Board (AC164127-5)	PIC24F PIMs	✓	✓	✓	–	–	✓ (2)
	PIC32MX PIMs	✓	✓	✓	–	–	✓ (2)
	PIC24EP & dsPIC33EP PIMs	✓+	✓	✓	–	–	✓ (2)
	PIC24H & dsPIC33F PIMs	✓	✓	✓	–	–	✓ (2)
Graphics Controller PICtail™ Plus Epson S1D13517 Board (AC164127-7)	PIC24F PIMs	✓	✓	✓	✓	–	✓ (2)
	PIC32MX PIMs	✓	✓	✓	✓	✓	✓ (2)
	PIC24EP & dsPIC33EP PIMs	✓+	✓	✓	✓	✓	✓ (2)
	PIC24H & dsPIC33F PIMs	✓+	✓+	✓+	✓+	✓+	✓ (2)
Low-Cost Controllerless (LCC) Graphics PICtail™ Plus Daughter Board (AC164144)	PIC24F PIMs	–	–	–	–	–	–
	PIC32MX PIMs	✓	–	–	✓ (3)	–	✓ (2)
	PIC24EP & dsPIC33EP PIMs	✓+	✓+	✓+	✓+ (3)	–	✓+ (2)
	PIC24H & dsPIC33F PIMs	–	–	–	–	–	–

Stand Alone Development Boards	
MPLAB® Starter Kit for PIC24H MCUs (DM240021)	Stand Alone Development Board with a built-in display.
MPLAB® Starter Kit for PIC24F MCUs (DM240011)	Stand Alone Development Board with a built-in display.
PIC24F PIMs	PIC24FJ128GA010 PIM (MA240011) PIC24FJ256GA110 PIM (MA240015) PIC24FJ256GB110 PIM (MA240014) PIC24FJ256GB210 PIM (MA240021)
PIC32MX PIMs	PIC32MX360F512L PIM (MA320001) PIC32MX460F512L PIM (MA320002) PIC32MX795F512L PIM (MA320003)
PIC24EP & dsPIC33EP PIMs	dsPIC33EP512MU810 PIM (MA330025-1) PIC24EP512GU810 (MA240025-1)
PIC24H & dsPIC33F PIMs	PIC24HJ128GP504 PIM (MA240016-2) dsPIC33FJ128GP804 PIM (MA330019-2)

✓ = Compatible (out of the box)

✓+ = Compatible (will need firmware modification)

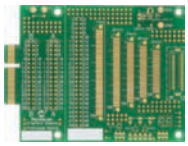
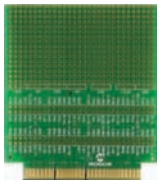
– = Incompatible

Notes:

1. SSD1926 supports up to WQVGA (480x272) displays.
2. Manually assemble chosen display panel to the protoyping board.
3. Run at 8 bpp with external memory.
4. 8 bpp or less using internal memory, 8 bpp or 16 bpp using external memory.
5. 8 bpp or 16 bpp with external memory.

Tools for Designing Graphical Displays

Development Tool	Part Number	Tool Image
PIC24FJ256DA210 Development Board	DM240312	
Explorer 16 Development Board	DM240001	
Multi Media Expansion Board	DM320005	
Graphics LCD Controller PICtail™ Plus SSD1926 Board	AC164127-5	
Graphics Controller PICtail™ Plus Epson S1D13517 Board	AC164127-7	
Low-Cost Controllerless (LCC) Graphics PICtail™ Plus Daughter Board	AC164144	
Graphics Display Truly 3.2″ 240x320 Board	AC164127-4	
Graphics Display Powertip 4.3″ 480x272 Board	AC164127-6	
Graphics Display Truly 5.7″ 640x480 Board	AC164127-8	
Graphics Display Truly 7″ 800x480 Board	AC164127-9	

Development Tool	Part Number	Tool Image
PIC32 Starter Kit	DM320001	
PIC32 USB Starter Kit II	DM320003-2	
PIC32 Ethernet Starter Kit	DM320004	
PIC24E USB Starter Kit	DM240012	
dsPIC33E USB Starter Kit	DM330012	
Graphics Display Prototype Board	AC164139	
Prototype PICtail™ Plus Daughter Board	AC164126	
MPLAB® Starter Kit for PIC24H MCUs	DM240021	
MPLAB® Starter Kit for PIC24F	DM240011	

Microchip Graphics Library Quick Reference Guide

Microchip Graphics Library Quick Reference Guide

August 22, 2011 – based on Graphics Library Version 3.01

This card contains brief API description. For complete API description, see Graphics Library Help File.

Predefined Types

GFX_COLOR	Data type for color data. Dependent on the COLOR_DEPTH setting. COLOR_DEPTH Data Type 8 typedef BYTE GFX_COLOR; 16 typedef WORD GFX_COLOR; 24 typedef DWORD GFX_COLOR;
XCHAR	Data type for characters. Set by macros in GraphicsConfig.h macro XCHAR Data Type #define USE_MULTIBYTECHAR signed int #define USE_UNSIGNED_XCHAR unsigned char non-of the above signed char

Device Driver Layer

Device Driver Layer	Description
void ResetDevice()	Initialize the display interface and resets the display.
void PutPixel(SHORT x, SHORT y)	Renders the pixel located on the given x,y position with the current set color.
GFX_COLOR GetPixel(SHORT x, SHORT y)	Returns the pixel color located on the given x,y position.
void DisplayBrightness(WORD level)	Sets the brightness of the display (level: 0-100).
WORD IsDeviceBusy()	Returns a non-zero if display controller is busy with the previous rendering operation. Zero if idle.
WORD GetMaxX()	Returns the maximum horizontal coordinate.
WORD GetMaxY()	Returns the maximum vertical coordinate.
void SetClip(BYTE control)	Enables/disables clipping. control definition CLIP_DISABLE Disable clipping CLIP_ENABLE Enable clipping
void SetClipRgn(SHORT left, SHORT top, SHORT right, SHORT bottom)	Sets the clipping region with the given coordinates.
WORD GetClipLeft()	Returns left clipping border.
WORD GetClipRight()	Returns right clipping border.
WORD GetClipTop()	Returns top clipping border.
WORD GetClipBottom()	Returns bottom clipping border.
void SetColor(GFX_COLOR color)	Sets the current drawing color.
GFX_COLOR GetColor()	Returns the current drawing color.
void TransparentColorEnable(GFX_COLOR color)	Set the transparent color and enable the transparent color feature for PutImage(). Graphics Config: USE_TRANSPARENT_COLOR
void TransparentColorDisable()	Disable the transparent color feature for PutImage(). Graphics Config: USE_TRANSPARENT_COLOR
WORD GetTransparentColorStatus()	Returns the current status of the transparent color feature of PutImage(). return value definition TRANSPARENT_COLOR_DISABLE feature is disabled TRANSPARENT_COLOR_ENABLE feature is enabled Graphics Config: USE_TRANSPARENT_COLOR
GFX_COLOR GetTransparentColor()	Returns the current transparent color set by TransparentColorEnable().
void SetActivePage(WORD page)	Set the active page. Active page is the buffer used for rendering.(1)

Device Driver Layer	Description
void SetVisualPage(WORD page)	Set the visible page. Visual page (display buffer) is the page shown on the display.(1)
WORD CopyWindow(DWORD srcAddr, DWORD dstAddr, WORD srcX, WORD srcY, WORD dstX, WORD dstY, WORD width, WORD height)	Copies the contents of a window located in the address defined by srcAddr and the pixel offset set by srcX and srcY to an address location defined by dstAddr and the pixel offset set by dstX and dstY. Size of the copied window is set by the given width and height.(2)
WORD CopyPageWindow(BYTE srcPage, BYTE dstPage, WORD srcX, WORD srcY, WORD dstX, WORD dstY, WORD width, WORD height);	Copies the contents of a window located in the page defined by srcPage and the pixel offset set by srcX and srcY to another page defined by dstPage and the pixel offset set by dstX and dstY. Size of the copied window is set by the given width and height.(1, 2)
WORD CopyBlock(DWORD srcAddr, DWORD dstAddr, DWORD srcOffset, DWORD dstOffset, WORD width, WORD height);	Copies a block of pixels from the given source address srcAddr with an offset srcOffset to a destination address given by dstAddr with a given offset dstOffset. Size of the copied block is set by the given width and height.(2)
void SwitchOnDoubleBuffering()	Turn on the automatic management of double buffering. All rendering will be performed on the currently set draw buffer. Graphics Config: USE_DOUBLE_BUFFERING
void SwitchOffDoubleBuffering()	Turn off the automatic management of double buffering. All rendering will be performed on the currently set display buffer. Graphics Config: USE_DOUBLE_BUFFERING
BYTE IsDisplayUpdatePending()	Returns the status of the display update set by RequestDisplayUpdate(). Graphics Config: USE_DOUBLE_BUFFERING
void RequestDisplayUpdate(void)	Schedule synchronization of the contents of the draw buffer and the display buffer at the next vertical blanking. Graphics Config: USE_DOUBLE_BUFFERING
void InvalidateAll()	Marks the whole screen area as invalidated. Graphics Config: USE_DOUBLE_BUFFERING
void InvalidateRectangle(WORD left, WORD top, WORD right, WORD bottom);	Invalidates the defined rectangular area in the display buffer. On the next synchronization of the draw and display buffer, the invalidated rectangle will be refreshed with the data in the draw buffer. Graphics Config: USE_DOUBLE_BUFFERING
void UpdateDisplayNow()	Synchronizes the draw and display buffer immediately. Graphics Config: USE_DOUBLE_BUFFERING

Notes:

1. The API is enabled only if the display controller supports multiple display buffers (or pages).
2. This is an optional feature and is implemented only if the display controller supports addressing of a block of pixel data.

Microchip Graphics Library Quick Reference Guide

Primitive Layer

Primitive Layer	Description
void InitGraph()	Initialize the display controller, sets the screen to BLACK, sets current color to WHITE, sets the cursor at (0,0), sets the line type to SOLID_LINE and sets the active page and visual page to 0.
void ClearDevice()	Clears the screen with the current color and sets the cursor at (0,0).
SHORT GetX()	Returns the graphic cursor x coordinates.
SHORT GetY()	Returns the graphic cursor y coordinates.
void MoveTo(**SHORT x,** **SHORT y)**	Moves the graphic cursor to the new x,y location.
void MovetoRel(SHORT x, SHORT y)	Moves the graphic cursor relative to the current location.
void SetFont(void *pFont)	Sets the current font to be used when rendering characters.
void SetFontOrientation(WORD orient)	Sets the font orientation to vertical or horizontal orientation. orient definition ORIENT_HOR horizontal ORIENT_VER vertical
BYTE GetFontOrientation()	Returns the current font orientation (vertical or horizontal). See SetFontOrientation().
WORD OutChar(XCHAR ch)	Renders the character represented by the character ID ch on the current graphic cursor position using the current set font and color.
WORD OutText(XCHAR *pString)	Renders a string of characters pointed to by pString on the current graphic cursor location using the current set font and color.
WORD OutTextXY(**SHORT x,** **SHORT y,** **XCHAR *pString)**	Renders the specified string pointed to by pString on the location specified by x,y using the current set font and color.
SHORT GetTextHeight(void *pFont)	Returns the height of the specified font.
SHORT GetTextWidth(**XCHAR pString,** **void *pFont)**	Returns the width of the specified string for the specified font.
void SetLineType(WORD lnType)	Sets the line type to render. lnType definition SOLID_LINE Solid line DASHED_LINE Dashed line
void SetLineThickness(WORD lnThickness)	Sets the line thickness to render. lnThickness definition NORMAL_LINE 1 pixel thick THICK_LINE 3 pixel thick
WORD Line(**SHORT x1,** **SHORT y1,** **SHORT x2,** **SHORT y2)**	Renders a line using the current set line type, line thickness and color from x1,y1 to x2,y2.
WORD LineRel(**SHORT x,** **SHORT y)**	Renders a line using the current set line type, line thickness and color from the current graphic cursor position to the relative position specified by the given x and y displacement.
WORD LineTo(**SHORT x,** **SHORT y)**	Renders a line using the current set line type, line thickness and color from the current graphic cursor position to the given x,y position.

Primitive Layer	Description
WORD Circle(**SHORT x,** **SHORT y,** **SHORT r)**	Renders a circle using the current set line type, line size and color located at center on x,y with a radius r.
WORD FillCircle(**SHORT x,** **SHORT y,** **SHORT r)**	Renders a filled circle using the current set color located at center on x,y with a radius r.
WORD DrawPoly(**SHORT numPoints,** **SHORT *polyPoints)**	Renders a polygon with the current line type, line size and color where the number of points is given by numPoints and the polygon points is given by the array pointed to by polyPoints. PolyPoints[n] = x0,y0,x1,y1,…xn,yn; n = numPoints
WORD Arc(**SHORT xL,** **SHORT yR,** **SHORT xR,** **SHORT yB,** **SHORT r1,** **SHORT r2,** **BYTE octant)**	Draws the octant arc of the beveled figure with the given centers, radii and octant mask.
WORD PutImage(**SHORT left,** **SHORT top,** **void *pBitmap,** **BYTE stretch)**	Renders the image pointed to by pBitmap starting from left, top position.
SHORT GetImageHeight(**void *pBitmap)**	Returns the image height.
SHORT GetImageWidth(**void *pBitmap)**	Returns the image width.
SHORT GetSineCosine(**SHORT v,** **WORD type)**	Returns the sine or cosine (type = GETSINE or GETCOSINE) values of the given angle v. Return values are normalized to 256.
SHORT Sine(**SHORT v)**	Returns the sine value of the given angle v. Return values are normalized to 256.
SHORT Cosine(**SHORT v)**	Returns the sine value of the given angle v. Return values are normalized to 256.
WORD AlphaBlendWindow(**DWORD foregroundWindowAddr,** **DWORD backgroundWindowAddr,** **DWORD destinationWindowAddr,** **WORD width,** **WORD height,** **BYTE alphaPercentage)**	Alpha blends the foreground and the background images with the dimension specified by width and height and writes the result to the destination. Graphics Config: USE_ALPHA_BLEND(1)
DWORD GFXGetPageXYAddress(**SHORT pageNumber,** **WORD x,** **WORD y)**	Calculates the address of the pixel located in x,y position inside the buffer specified by pageNumber. Graphics Config: USE_ALPHA_BLEND(1)
DWORD GFXGetPageOriginAddress(**SHORT pageNumber)**	Calculates the address of the buffer specified by pageNumber. Graphics Config: USE_ALPHA_BLEND(1)
WORD ExternalMemoryCallback(**GFX_EXTDATA *memory,** **LONG offset,** **WORD nCount,** **void *buffer)**	When using external memory, this function must be implemented in the application code and will contain code to access external memory with the appropriate external memory drivers. Graphics Config: USE_FONT_EXTERNAL or USE_BITMAP_EXTERNAL

Note:
1. This is an optional feature and is implemented only if the display controller supports alpha blending.

Microchip Graphics Library Quick Reference Guide

GOL Layer

GOL Layer	Description
void GOLInit()	This function initializes the object layer, primitive layer and display driver layer. Creates a default style scheme with default settings.
WORD GOLDraw()	Renders all widgets that needs to be rendered on the active list.
WORD GOLDrawCallback()	Function implemented in the application. Function called by GOLDraw() to allow application defined rendering.
void GOLRedraw(OBJ_HEADER pObj)	Sets the drawing states of the widget to be redrawn on the next GOLDraw().
void GOLRedrawRec(SHORT left, SHORT top, SHORT right, SHORT bottom)	Marks all widgets in the active list intersected by the given rectangular area.
WORD IsObjUpdated(OBJ_HEADER pObj)	Tests if the widget is pending to be redrawn. Returns a non zero if pending and zero if not.
void GOLDrawComplete(OBJ_HEADER pObj)	Resets the drawing states of the widget.
void GOLAddObject(OBJ_HEADER pObj)	Adds the widget to the end of the active list.
void GOLFree()	Frees up the memory used by the current active list. Active list becomes empty.
OBJ_HEADER *GOLFindObject(WORD ID)	Returns the pointer to the widget in the active list with the given ID.
BOOL GOLDeleteObject(OBJ_HEADER pObj)	Deletes a widget from the current active list.
BOOL GOLDeleteObjectByID(WORD ID)	Deletes a widget from the current active list.
GOL_OBJ_TYPE GetObjType(OBJ_HEADER pObj)	Returns the widget type of the specified widget.
WORD GetObjID(OBJ_HEADER pObj)	Returns the object ID of the specified widget.
OBJ_HEADER *GetObjNext(OBJ_HEADER pObj)	Returns the pointer to the next widget right after the specified widget in the active list.
void GOLNewList()	Starts a new but empty list of widgets. Use GOLAddObjects() to populate the list.
OBJ_HEADER *GOLGetList()	Returns the pointer to the current active list.
void GOLSetList(OBJ_HEADER pObjList)	Sets the active list to the pointer specified by pObjList.
void GOLSetFocus(OBJ_HEADER pObj)	Sets the widget pObj to be focused.
OBJ_HEADER *GOLGetFocus()	Returns the pointer to the currently focused widget in the active list.
WORD GOLCanBeFocused(OBJ_HEADER pObj)	Returns a non-zero if the widget can be focused.
OBJ_HEADER *GOLGetFocusNext()	Returns the pointer of the next widget that can be focused in the active list.
OBJ_HEADER *GOLGetFocusPrev()	Returns the pointer of the previous widget that can be focused in the active list.

GOL Layer	Description
GOLPanelDraw(SHORT left, SHORT top, SHORT right, SHORT bottom, SHORT radius, GFX_COLOR faceClr, GFX_COLOR embossLtClr, GFX_COLOR embossDkClr, void *pBitmap, WORD embossSize)	Sets up the parameters to draw a panel. Use this API to set up GOLPanelDrawTsk() and GOLTwoTonePanelDrawTask().
WORD GOLPanelDrawTsk()	Renders a panel with the parameters set by GOLPanelDraw().
WORD GOLTwoTonePanelDrawTsk()	Renders a two tone panel with the parameters set by GOLPanelDraw().
WORD GetState(OBJ_HEADER *pObj, WORD stateBits)	Returns the current value of the state bits of the widget.
void SetState(OBJ_HEADER *pObj, WORD stateBits)	Sets the state bit(s) of the widget.
void ClrState(OBJ_HEADER *pObj, WORD stateBits)	Clears the state bit(s) of the widget.
void GOLMsg(GOL_MSG *pMsg)	Process the GOL_MSG and checks which widget is affected by the message. For each affected widget, GOLMsgCallback() is called for the application to have an opportunity to customize the reaction to the message and cancel or call the default reaction to the message.
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER * pObj, GOL_MSG * pMsg)	Function implemented in the application. Function called by GOLMsg() to allow application to implement its own interpretation of messages and customize widget behavior.
GOL_SCHEME * GOLCreateScheme()	Creates a style scheme object with application defined styles or using graphics library assigned default styles.
void GOLSetScheme(OBJ_HEADER *pObj, GOL_SCHEME *pScheme)	Sets the GOL_SCHEME to be used for the widget.
GOL_SCHEME *GOLSetScheme(OBJ_HEADER *pObj)	Gets the GOL_SCHEME used by the widget.
GOL_SCHEME *GOLGetSchemeDefault()	Returns the pointer to the default scheme created when GOLInit() is called.
GFX_COLOR RGBConvert(WORD red, WORD green, WORD blue)	Converts 8-8-8 RGB color data to color format set by COLOR_DEPTH.

Microchip Graphics Library Quick Reference Guide

Widget API

AnalogClock	Button	Chart	Chart2	Chart3
Graphics Config: USE_ANALOGCLOCK	Graphics Config: USE_BUTTON USE_BUTTON_MULTI_LINE	Graphics Config: USE_CHART	void ChHideSeries(CHART *pObj, WORD seriesNum)	XCHAR *ChGetSampleLabel(CHART *pObj)
ANALOGCLOCK * AcCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, SHORT hour, SHORT minute, SHORT radius, BOOL sechand, WORD state, void * pBitmap, GOL_SCHEME * pScheme)	BUTTON * BtnCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, SHORT radius, WORD state, void * pBitmap, XCHAR * pText, GOL_SCHEME * pScheme)	CHART * ChCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, DATASERIES * pData, CHARTPARAM * pParam, GOL_SCHEME * pScheme)	void ChGetShowSeriesCount(CHART * pObj)	WORD ChGetSampleStart(CHART *pObj)
WORD AcDraw(void * pObj)	WORD BtnDraw(void * pObj)	WORD ChDraw(void * pObj)	WORD ChGetShowSeriesStatus(CHART * pObj)	WORD ChGetSampleEnd(CHART *pObj)
WORD AcHandsDraw(ANALOGCLOCK * pObj, SHORT hand, SHORT thickness, WORD color, void * pBitmap)	XCHAR *BtnGetText(BUTTON *pObj)	XCHAR *ChGetTitle(CHART *pObj)	void ChSetValueLabel(CHART* pObj, XCHAR* pValueLabel)	void ChSetPercentRange(CHART * pObj, WORD min, WORD max)
void AcSetHour(ANALOGCLOCK * pObj, SHORT hour)	void BtnSetText(BUTTON *pObj, XCHAR *pString)	void ChSetTitle(CHART *pObj, XCHAR *pTitle)	XCHAR *ChGetValueLabel(CHART* pObj)	WORD ChGetPercentRange(CHART *pObj)
void AcSetMinute(ANALOGCLOCK * pObj, SHORT minute)	void *BtnGetBitmap(BUTTON *pObj)	void ChSetTitleFont(CHART *pObj, void *pFont)	WORD ChGetValueMax(CHART *pObj)	void ChSetSampleRange(CHART * pObj, WORD start, WORD end)
void AcSetSecond(ANALOGCLOCK * pObj, SHORT second)	void BtnSetBitmap(BUTTON *pObj, void *pBitmap)	XCHAR *ChGetTitleFont(CHART *pObj)	WORD ChGetValueMin(CHART *pObj)	WORD ChGetSampleRange(CHART *pObj)
typedef struct { OBJ_HEADER hdr; SHORT radius; SHORT centerx; SHORT centery; SHORT valueS; SHORT prev_valueS; SHORT valueM; SHORT prev_valueM; SHORT valueH; SHORT prev_valueH; void * pBitmap; } ANALOGCLOCK;	void BtnMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)	DATASERIES *ChAddDataSeries(CHART *pObj, WORD nSamples, WORD *pData XCHAR *pNames)	void ChSetValueRange(CHART * pObj, WORD min, WORD max)	WORD ChGetPercentMax(CHART *pObj)
	WORD BtnTranslateMsg(void * pObj, GOL_MSG * pMsg)	void ChRemoveDataSeries(CHART * pObj, WORD number)	WORD ChGetValueRange(CHART* pObj)	WORD ChGetPercentMin(CHART *pObj)
	typedef struct { OBJ_HEADER hdr; SHORT radius; SHORT textWidth; SHORT textHeight; XCHAR * pText; void * pBitmap; } BUTTON;	void ChShowSeries(CHART *pCh, WORD seriesNum)	void ChSetSampleLabel(CHART *pObj, XCHAR *pXLabel)	void ChSetColorTable(CHART* pObj, GFX_COLOR *pColorArray)

Microchip Graphics Library Quick Reference Guide

Widget API (Continued)

Chart4	Round Dial	Check Box	Digital Meter	Edit Box
GFX_COLOR *ChGetColorTable(CHART* pObj)	Graphics Config: USE_ROUNDDIAL	Graphics Config: USE_CHECKBOX	Graphics Config: USE_DIGITALMETER	Graphics Config: USE_EDITBOX
void *ChGetAxisLabelFont(CHART *pObj)	ROUNDDIAL * RdiaCreate(WORD ID, SHORT x, SHORT y, SHORT radius, WORD state, SHORT res, SHORT value, SHORT max, GOL_SCHEME * pScheme)	CHECKBOX * CbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, GOL_SCHEME * pScheme)	DIGITALMETER * DmCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, DWORD Value, BYTE NoOfDigits, BYTE DotPos, GOL_SCHEME * pScheme)	EDITBOX * EbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, WORD charMax, GOL_SCHEME * pScheme)
void ChSetAxisLabelFont(CHART *pObj, void* pFont)	WORD RdiaDraw(void * pObj)	WORD CbDraw(void * pObj)	WORD DmDraw(void * pObj)	WORD EbDraw(void * pObj)
void *ChGetGridLabelFont(CHART *pObj)	void RdiaIncVal(ROUNDDIAL *pObj)	XCHAR *CbGetText(CHECKBOX pObj)	WORD DmGetValue(DIGITALMETER *pObj)	XCHAR *EbGetText(EDITBOX pObj)
void ChSetGridLabelFont(CHART *pObj, void* pFont)	void RdiaDecVal(ROUNDDIAL *pObj)	void CbSetText(CHECKBOX pObj, XCHAR *pText)	void DmSetValue(DIGITALMETER *pObj, WORD value)	void EbSetText(EDITBOX pObj, XCHAR *pText)
void ChFreeDataSeries(void * pObj)	WORD RdiaGetVal(ROUNDDIAL *pObj)	void CbMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)	void DmIncVal(DIGITALMETER *pObj, WORD deltaValue)	void EbAddChar(EDITBOX * pObj, XCHAR ch)
WORD ChTranslateMsg(void * pObj, GOL_MSG * pMsg)	void RdiaSetVal(ROUNDDIAL *pObj, WORD value)	WORD CbTranslateMsg(void * pObj, GOL_MSG * pMsg)	void DmDecVal(DIGITALMETER *pObj, WORD deltaValue)	void EbDeleteChar(EDITBOX * pObj)
typedef struct { OBJ_HEADER hdr; CHARTPARAM prm; DATASERIES * pChData; } CHART;	void RdiaMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)	typedef struct { OBJ_HEADER hdr; SHORT textHeight; XCHAR * pText; } CHECKBOX;	WORD DmTranslateMsg(void * pObj, GOL_MSG * pMsg)	void EbMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)
typedef struct { XCHAR * pSData; WORD samples; BYTE show; WORD * pData; void * pNextData; } DATASERIES;	WORD RdiaTranslateMsg(void * pObj, GOL_MSG * pMsg)		typedef struct { OBJ_HEADER hdr; SHORT textHeight; DWORD Cvalue; DWORD Pvalue; BYTE NoOfDigits; BYTE DotPos; } DIGITALMETER;	WORD EbTranslateMsg(void * pObj, GOL_MSG * pMsg)
typedef struct { XCHAR * pTitle; XCHAR * pSmplLabel; XCHAR * pValLabel; SHORT seriesCount; WORD smplStart; WORD smplEnd; WORD valMax; WORD valMin; WORD perMax; WORD perMin; WORD * pColor; void * pTitleFont; void * pAxisLabelsFont; void * pGridLabelsFont; } CHARTPARAM;	typedef struct { OBJ_HEADER hdr; SHORT xCenter; SHORT yCenter; SHORT radius; SHORT value; WORD max; WORD res; SHORT curr_xPos; SHORT curr_yPos; SHORT new_xPos; SHORT new_yPos; SHORT vAngle; } ROUNDDIAL;			typedef struct { OBJ_HEADER hdr; SHORT textHeight; XCHAR * pBuffer; WORD charMax; WORD length; } EDITBOX;

Microchip Graphics Library Quick Reference Guide

Widget API (Continued)

Grid	Grid2	Group Box	List Box	List Box2
Graphics Config: USE_GRID	void GridSetFocus(GRID * pObj, SHORT column, SHORT row)	Graphics Config: USE_GROUPBOX	Graphics Config: USE_LISTBOX	WORD LbGetVisibleCount(LISTBOX *pObj)
GRID * GridCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, SHORT numColumns, SHORT numRows, SHORT cellWidth, SHORT cellHeight, GOL_SCHEME * pScheme)	void GridMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)	GROUPBOX * GbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, GOL_SCHEME * pScheme)	LISTBOX * LbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, GOL_SCHEME * pScheme)	void LbClrtSel(LISTBOX pObj, LISTITEM *pItem)
WORD GridDraw(void * pObj)	WORD GridTranslateMsg(void * pObj, GOL_MSG * pMsg)	WORD GbDraw(void * pObj)	WORD LbDraw(void * pObj)	void LbSetBitmap(LISTITEM *pItem, void *pBtmap)
void GridClearCellState(GRID * pObj, SHORT column, SHORT row, WORD state)	typedef struct { OBJ_HEADER hdr; SHORT numColumns; SHORT numRows; SHORT cellHeight; SHORT cellWidth; SHORT focusX; SHORT focusY; GRIDITEM * gridObjects; } GRID;	XCHAR *GbGetText(GROUPBOX *pObj)	LISTITEM *LbGetItemList(LISTBOX *pObj)	void *LbGetBitmap(LISTITEM *pItem)
WORD GridGetFocusX(GRID *pObj)	typedef struct { void * data; WORD status; } GRIDITEM;	void GbSetText(GROUPBOX *pObj, XCHAR * pText)	LISTITEM * LbAddItem(LISTBOX * pObj, LISTITEM * pPrevItem, XCHAR * pText, void * pBitmap, WORD status, WORD data)	void LbDelItemsList(void * pObj)
WORD GridGetFocusY(GRID *pObj)		WORD GbTranslateMsg(void * pObj, GOL_MSG * pMsg)	void LbDelItem(LISTBOX * pObj, LISTITEM * pItem)	void LbMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)
void GridFreeItems(void * pObj)		typedef struct { OBJ_HEADER hdr; SHORT textWidth; SHORT textHeight; XCHAR * pText; } GROUPBOX;	void LbChangeSel(LISTBOX * pObj, LISTITEM * pItem)	WORD LbTranslateMsg(void * pObj, GOL_MSG * pMsg)
void * GridGetCell(GRID * pObj, SHORT column, SHORT row, WORD * cellType)			void LbSetSel(LISTBOX *pObj, LISTITEM *pItem)	typedef struct { OBJ_HEADER hdr; LISTITEM * pItemList; LISTITEM * pFocusItem; WORD itemsNumber; SHORT scrollY; SHORT textHeight; } LISTBOX;
WORD GridSetCell(GRID * pObj, SHORT column, SHORT row, WORD state, void * data)			LISTITEM * LbGetSel(LISTBOX * pObj, LISTITEM * pFromItem)	typedef struct { void * pPrevItem; void * pNextItem; WORD status; XCHAR * pText; void * pBitmap; WORD data; } LISTITEM;
void GridSetCellState(GRID * pObj, SHORT column, SHORT row, WORD state)			SHORT LbGetFocusedItem(LISTBOX * pObj)	
			void LbSetFocusedItem(LISTBOX * pObj, SHORT index)	
			WORD LbGetCount(LISTBOX pObj)	

Microchip Graphics Library Quick Reference Guide

Widget API (Continued)

Meter	Picture Control	Progress Bar	Radio Button
Graphics Config: USE_METER	Graphics Config: USE_PICTURE	Graphics Config: USE_PROGRESSBAR	Graphics Config: USE_RADIOBUTTON
METER * MtrCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, SHORT value, SHORT minValue, SHORT maxValue, void * pTitleFont, void * pValueFont, XCHAR * pText, GOL_SCHEME * pScheme)	PICTURE * PictCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, char scale, void * pBitmap, GOL_SCHEME * pScheme)	PROGRESSBAR * PbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, WORD pos, WORD range, GOL_SCHEME * pScheme)	RADIOBUTTON * RbCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, GOL_SCHEME * pScheme)
WORD MtrDraw(void * pObj)	WORD PictDraw(void * pObj)	WORD PbDraw(void * pObj)	WORD RbDraw(void * pObj)
void MtrSetVal(METER * pObj, SHORT newVal)	void PictSetBitmap(PICTURE *pObj, void *pBtmap)	void PbSetRange(PROGRESSBAR * pObj, WORD range)	WORD RbGetCheck(RADIOBUTTON * pObj)
WORD MtrGetVal(METER *pObj)	void *PictSetBitmap(PICTURE *pObj)	WORD PbGetRange(PROGRESSBAR * pObj)	void RbSetCheck(RADIOBUTTON * pObj, WORD ID)
void MtrDecVal(METER *pObj, WORD deltaValue)	WORD PictGetScale(PICTURE *pObj)	void PbSetPos(PROGRESSBAR * pObj, WORD position)	XCHAR *RbGetText(RADIOBUTTON * pObj)
void MtrIncVal(METER *pObj, WORD deltaValue)	void PictSetScale(PICTURE *pObj, WORD scale)	WORD PbGetPos(PROGRESSBAR * pObj)	void RbSetText(RADIOBUTTON * pObj XCHAR *pText)
void MtrSetScaleColors(METER *pObj, WORD arc1, WORD arc2, WORD arc3, WORD arc4, WORD arc5, WORD arc6)	WORD PictTranslateMsg(void * pObj, GOL_MSG * pMsg)	WORD PbTranslateMsg(PROGRESSBAR * pObj, GOL_MSG * pMsg)	void RbMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)
void MtrSetTitleFont(METER *pMtr, void *pFont)	typedef struct { OBJ_HEADER hdr; char scale; void * pBitmap; } PICTURE;	typedef struct { OBJ_HEADER hdr; WORD pos; WORD prevPos; WORD range; } PROGRESSBAR;	WORD RbTranslateMsg(void * pObj, GOL_MSG * pMsg)
void MtrSetValueFont(METER *pMtr, void *pFont)			typedef struct { OBJ_HEADER hdr; OBJ_HEADER * pHead; OBJ_HEADER * pNext; SHORT textHeight; XCHAR * pText; } RADIOBUTTON;
void MtrMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)			
WORD MtrTranslateMsg(void * pObj, GOL_MSG * pMsg)			
typedef struct { OBJ_HEADER hdr; XCHAR * pText; SHORT value; SHORT minValue; SHORT maxValue; SHORT xCenter; SHORT yCenter; SHORT radius; SHORT xPos; SHORT yPos; WORD arcColor6; WORD arcColor5; WORD arcColor4; WORD arcColor3; WORD arcColor2; WORD arcColor1; void * pTitleFont; void * pValueFont; } METER;			

Microchip Graphics Library Quick Reference Guide

Widget API (Continued)

Slider/Scroll Bar	Static Text	Text Entry	Text Entry2	Window
Graphics Config: USE_SLIDER	Graphics Config: USE_STATICTEXT	Graphics Config: USE_TEXTENTRY	void TeDelKeyMembers(void * pObj)	Graphics Config: USE_WINDOW
SLIDER * SldCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, WORD range, WORD page, WORD pos, GOL_SCHEME * pScheme)	STATICTEXT * StCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, XCHAR * pText, GOL_SCHEME * pScheme)	TEXTENTRY * TeCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, SHORT horizontalKeys, SHORT verticalKeys, XCHAR * pText[], void * pBuffer, WORD bufferLength, void * pDisplayFont, GOL_SCHEME * pScheme)	BOOL TeSetKeyText(TEXTENTRY * pObj, WORD index, XCHAR * pText)	WINDOW * WndCreate(WORD ID, SHORT left, SHORT top, SHORT right, SHORT bottom, WORD state, void * pBitmap, XCHAR * pText, GOL_SCHEME * pScheme)
WORD SldDraw(void * pObj)	WORD StDraw(void * pObj)	WORD TeDraw(void * pObj)	void TeMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)	WORD WndDraw(void * pObj)
void SldSetPage(SLIDER * pObj, WORD page)	XCHAR *StGetText(STATICTEXT* pObj)	XCHAR * TeGetBuffer(TEXTENTRY *pObj)	WORD TeTranslateMsg(void * pObj, GOL_MSG * pMsg)	XCHAR *WndGetText(WINDOW* pObj)
WORD SldGetPage(SLIDER * pObj)	void StGetText(STATICTEXT* pObj XCHAR * pText)	void TeSetBuffer(TEXTENTRY *pObj XCHAR * pText)	typedef struct { OBJ_HEADER hdr; SHORT horizontalKeys; SHORT verticalKeys; XCHAR * pTeOutput; WORD CurrentLength; WORD outputLenMax; KEYMEMBER * pActiveKey; KEYMEMBER * pHeadOfList; void * pDisplayFont; } TEXTENTRY;	void *WndSetText(WINDOW* pObj XCHAR * pText)
void SldSetPos(SLIDER * pObj, WORD position)	WORD StTranslateMsg(void * pObj, GOL_MSG * pMsg)	void TeClearBuffer(TEXTENTRY *pOb)	typedef struct { SHORT left; SHORT top; SHORT right; SHORT bottom; SHORT index; WORD state; BOOL update; WORD command; XCHAR * pKeyName; SHORT textWidth; SHORT textHeight; void * pNextKey; } KEYMEMBER;	WORD WndTranslateMsg(void * pObj, GOL_MSG * pMsg)
WORD SldGetPos(SLIDER * pObj)	typedef struct { OBJ_HEADER hdr; SHORT textHeight; XCHAR * pText; } STATICTEXT;	WORD TeGetKeyCommand(TEXTENTRY * pObj, WORD index)		typedef struct { OBJ_HEADER hdr; SHORT textHeight; XCHAR * pText; void * pBitmap; } WINDOW;
void SldSetRange(SLIDER * pObj, WORD range)		BOOL TeSetKeyCommand(TEXTENTRY * pObj, WORD index WORD command)		
WORD SldGetRange(SLIDER * pObj)		KEYMEMBER * TeCreateKeyMembers(TEXTENTRY * pObj, XCHAR * pText[])		
void SldIncPos(SLIDER * pObj)		void TeAddChar(TEXTENTRY * pObj)		
void SldDeccPos(SLIDER * pObj)		BOOL TeIsKeyPressed(TEXTENTRY * pObj, WORD index)		
void SldMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)		void TeSpaceChar(TEXTENTRY * pObj)		
WORD SldTranslateMsg(void * pObj, GOL_MSG * pMsg)				
typedef struct { OBJ_HEADER hdr; WORD currPos; WORD prevPos; WORD range; WORD pos; WORD page; WORD thWidth; WORD thHeight; } SLIDER;				
void SldMsgDefault(WORD translatedMsg, void * pObj, GOL_MSG * pMsg)		void TeSpaceChar(TEXTENTRY * pObj)		
typedef struct { OBJ_HEADER hdr; WORD currPos; WORD prevPos; WORD range; WORD pos; WORD page; WORD thWidth; WORD thHeight; } SLIDER;				

Notes

Support

Microchip is committed to supporting its customers in developing products faster and more efficiently. We maintain a worldwide network of field applications engineers and technical support ready to provide product and system assistance. In addition, the following service areas are available at **www.microchip.com:**

- **Support** link provides a way to get questions answered fast: **http://support.microchip.com**
- **Sample** link offers evaluation samples of any Microchip device: **http://sample.microchip.com**
- **Forum** link provides access to knowledge base and peer help: **http://forum.microchip.com**
- **Buy** link provides locations of Microchip Sales Channel Partners: **www.microchip.com/sales**

Training

If additional training interests you, then Microchip can help. We continue to expand our technical training options, offering a growing list of courses and in-depth curriculum locally, as well as significant online resources – whenever you want to use them.

- Technical Training Centers: **www.microchip.com/training**
- MASTERs Conferences: **www.microchip.com/masters**
- Worldwide Seminars: **www.microchip.com/seminars**
- eLearning: **www.microchip.com/webseminars**
- Resources from our Distribution and Third Party Partners **www.microchip.com/training**

Sales Office Listing

AMERICAS

Atlanta
Tel: 678-957-9614
Boston
Tel: 774-760-0087
Chicago
Tel: 630-285-0071
Cleveland
Tel: 216-447-0464
Dallas
Tel: 972-818-7423
Detroit
Tel: 248-538-2250
Indianapolis
Tel: 317-773-8323
Los Angeles
Tel: 949-462-9523
Santa Clara
Tel: 408-961-6444
Toronto
Mississauga, Ontario
Tel: 905-673-0699

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Denmark - Copenhagen
Tel: 45-4450-2828
France - Paris
Tel: 33-1-69-53-63-20
Germany - Munich
Tel: 49-89-627-144-0
Italy - Milan
Tel: 39-0331-742611
Netherlands - Drunen
Tel: 31-416-690399
Spain - Madrid
Tel: 34-91-708-08-90
UK - Wokingham
Tel: 44-118-921-5869

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
China - Beijing
Tel: 86-10-8569-7000
China - Chengdu
Tel: 86-28-8665-5511
China - Chongqing
Tel: 86-23-8980-9588
China - Hangzhou
Tel: 86-571-2819-3187
China - Hong Kong SAR
Tel: 852-2401-1200
China - Nanjing
Tel: 86-25-8473-2460
China - Qingdao
Tel: 86-532-8502-7355
China - Shanghai
Tel: 86-21-5407-5533
China - Shenyang
Tel: 86-24-2334-2829
China - Shenzhen
Tel: 86-755-8203-2660
China - Wuhan
Tel: 86-27-5980-5300
China - Xiamen
Tel: 86-592-2388138
China - Xian
Tel: 86-29-8833-7252
China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
India - New Delhi
Tel: 91-11-4160-8631
India - Pune
Tel: 91-20-2566-1512
Japan - Yokohama
Tel: 81-45-471- 6166
Korea - Daegu
Tel: 82-53-744-4301
Korea - Seoul
Tel: 82-2-554-7200
Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Malaysia - Penang
Tel: 60-4-227-8870
Philippines - Manila
Tel: 63-2-634-9065
Singapore
Tel: 65-6334-8870
Taiwan - Hsin Chu
Tel: 886-3-5778-366
Taiwan - Kaohsiung
Tel: 886-7-536-4818
Taiwan - Taipei
Tel: 886-2-2500-6610
Thailand - Bangkok
Tel: 66-2-694-1351

8/2/11

Microcontrollers • Digital Signal Controllers • Analog • Memory • Wireless

DS01394A

MICROCHIP

www.microchip.com

Microchip Technology Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199