Box photo and video with the FEZ Spider Mainboard.

AUTHORS:
Group FEZ06 (Chirato Antonio, Di Prima Giuseppe, Zanfino Antonio)

Version 1.0

June, 2014

Introduction.

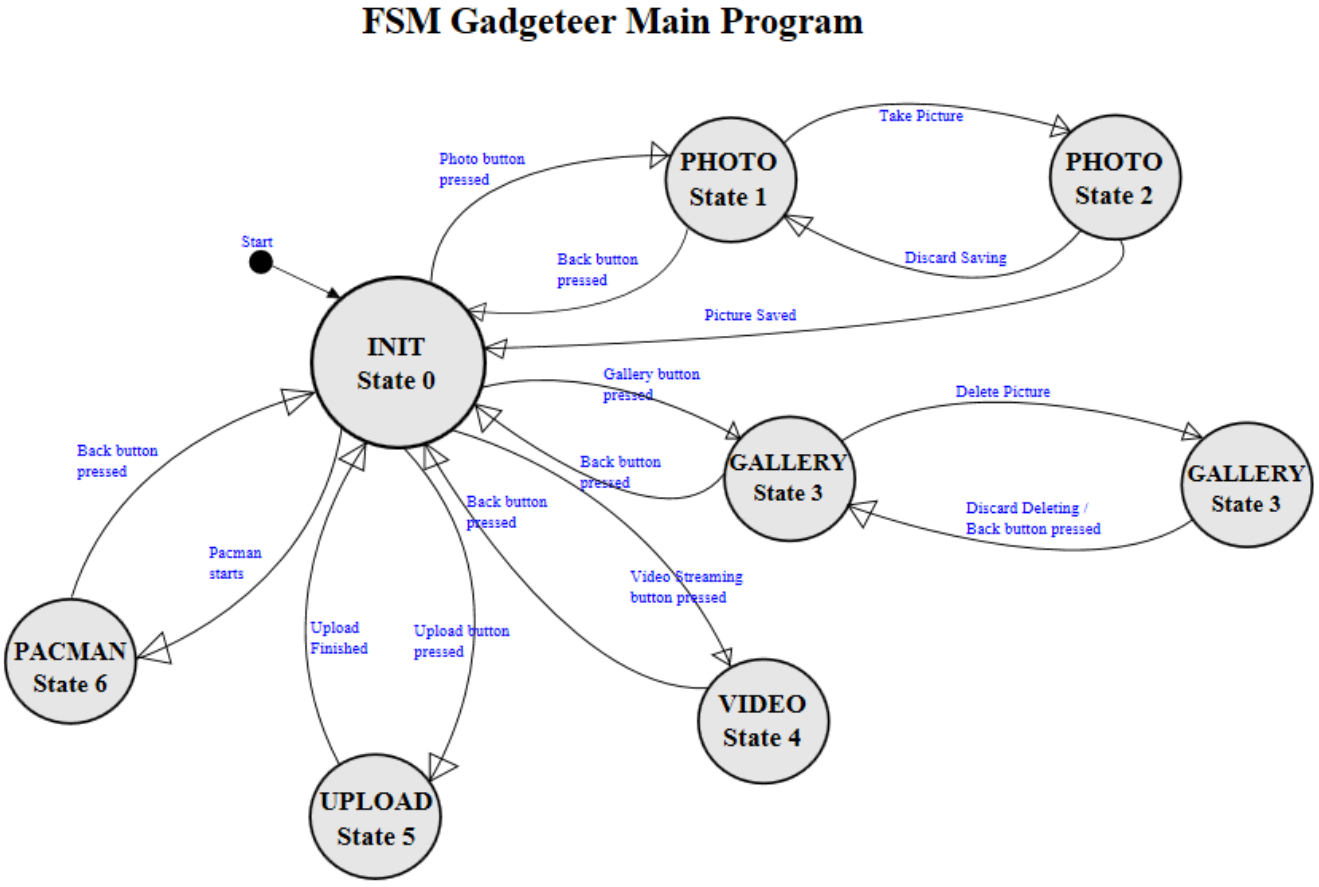
The following project was created for the "FEZ Spider Mainboard" that is a .NET Gadgeteer-compatible mainboard based on GHI Electronics EMX module. In particular, it interacts with the peripherals connected to the mainboard, such as display, camera, joystick, LEDs, Ethernet, SD card and buttons. The project can be divided in two parts:

- **FEZ Spider part**: contains the code's portion loaded on the device to manage its modules;
- **Server part**: the remaining portion of the code is used to create a desktop application that receives data streams from the device through an Ethernet connection.

Below the two portions of the project are described in detail.

FEZ Spider part.

In this part, everything starts when the device is powered. The program is built like a final state machine, in which each different screen represents a different state. When the program starts, Main method runs automatically, initializes the mainboard and its modules, using Gadgeteer and Gadgeteer.Modules library, and then calls the ProgramStarted method.

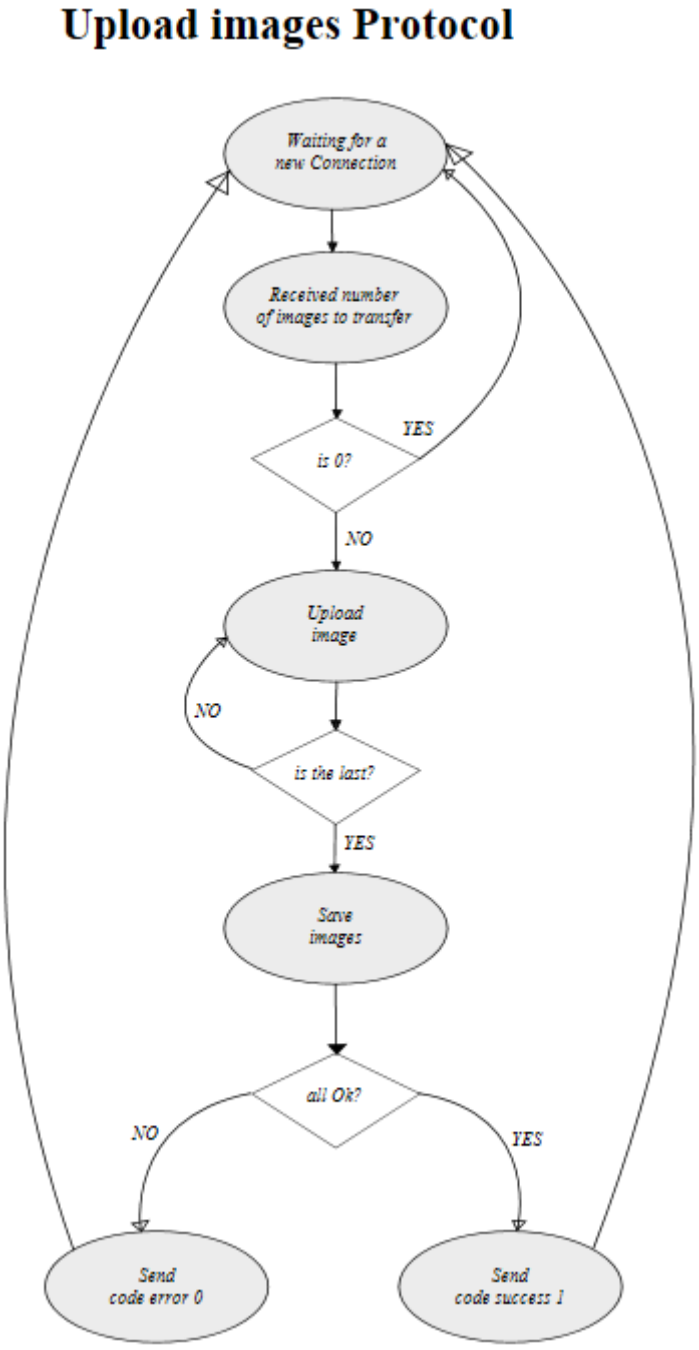


ProgramStarted sets the state as an initial state (0), initializes the display with the main window and creates all others screens that could appear in the various states of the program. Each screen is a StackPanel, composed by other StackPanels; for instance in the main window the buttons displayed are StackPanels with a specific image as background. The Network Interface Connection was forced to a static IP configuration like this `ethernet.UseStaticIP("192.168.137.2", "255.255.255.0", "192.168.137.1")` and, if an Ethernet cable is connected to the device, a connection with the desktop application is established. If there is an SD card in the right slot, it is mounted. The events for SD card, Ethernet, buttons, camera and joystick are created. The highest priority is assigned to the main thread (the graphical one) in order to make more efficient the display refresh.
The two LEDs are associated respectively to Ethernet and SD card. The first one turns green when an Ethernet connection is established; this is verified by the Ethernet network status, if the isNetworkUp property is true, the respective event turns on the led green, otherwise, the event NetworkDown turns on the led red and communicates to the user the lack of Ethernet connection. The second led turns green when an SD card is mounted, while turns red when there is no SD card in the slot; the two events, SDCardMounted and SDCardUnmounted, manage the led's colors.
At this point the device is waiting for the first user's tap on the main screen now visible on the display.
Two events are associated to each StackPanel that represent a different button in the main screen: TouchDown and TouchUp event. The first one only changes the buttons background image and invalidates the main window to make it effective, while the second calls the correct method to launch the action required and sets the program's state to the current state.
The first button in the upper-left corner of the main screen is the photo button; when tapped, the program controls if an SD card is in the slot calling the VerifySDCard method, which returns true if the property IsCardMounted is true. In this case the program enables the camera streaming and the user can take a picture tapping on the screen or clicking the confirm button on the case, otherwise an explanatory picture shows to the user that is not inserted any SD card. The camera's BitmapStreamed event is used to enables the camera. Whenever the user takes a picture, it is captured by the camera and saved in a global variable. The Bitmap saved is shown on the display and the program's state changes again. In the current state, the system asks to the user if the photo has to be saved to the SD card. The save and discard icons will appear on the display, so that the user can choose to save the image tapping the first one, or pressing the confirm button, otherwise discard the image tapping the second or pressing the back button. If the user decides to discard the picture, the system simply returns to the last state where the streaming camera screen is shown while if he decides to save the image, the VerifySDCard method is called again and, on positive outcome, the program starts to save picture on SD card. To prevent the image from its loosing after the saving, caused by a suddenly remotion of the SD, the card is unmounted in order to assert modifications. The Saving operation is done in another thread with lower priority to avoid loss of graphic reactivity; during this operation the SD card's LED turns blue and blinking. At the end of the saving process, the system comes back to initial state; SD card's LED turns green and main screen is shown again.
In the upper-right corner of the main screen there is the photo gallery button; when user presses this button, the system changes its state and verifies again if an SD card is inserted. If so, the user can watch, one at the time, all the photos stored in the SD card. If any photo

is stored in it, an explicative image informs that no images are available. The photos can be switched by using the joystick. On it any event can't be set, but its position, in particular its degree of inclination can be verified iteratively through an event triggered by a timer. If its inclination is greater than 90% or lesser than -90%, in the gallery is shown the next or the previous image. On each image, a trash button is applied; when it is tapped, its TouchDown event modifies its image background while its TouchUp event calls the removeImage method and removes the respective image from the SD card; during removal the SD card's LED turns orange and blinking.
Pressing the back button, the system returns to the main screen and changes its state to initial state.
The lower-left button starts the video recording; when tapped, the pictures that are captured by the camera are sent to the server, where are shown in "real-time". In the button's TouchUp event, if the Ethernet connection is active, in a new thread a TCP connection with the server is created. This thread is waiting for insertion of new Bitmap images in a global shared queue by the main thread. Whenever a picture is ready, it is saved in the queue by the camera's BitmapStreamed event and notified by static AutoResetEvent to the sending thread. The sending thread waits until a picture is available on the queue and, when notified, sends it and waits again. This process ends when the user presses the back button and the system come back to the initial state. While the streaming is sent, the Ethernet LED turns orange and blinking, at the end it turns green if the video is sent correctly, red otherwise.
The last button is reserved to transfer the images from the SD card to the server. Like the video streaming, this function is activated only when an Ethernet connection is available, but in this case an SD card must be inserted in the slot and at least one image must be present in it; otherwise, nothing happens and the program stays on the same state (initial). If all goes well, a new thread is created; this create a new TCP connection with the server, fetches the images from the SD card and converts them in another particular Bitmap format and finally sends them. The image must be converted because Bitmap objects are different in .NET than in .NETMF. While the sending the Ethernet LED turns orange and blinking, at the end it turns green and the system came back to the initial state. The main screen is again shown on the display.
As a bonus, another event is set on the joystick's central button. When the user presses it, after a few seconds, on the display appears a special screen representing the game Pacman. In this screen the user plays using the joystick and when tired, can press the back button that return to the main screen.

Server part.

The server application is used to make a video stream or to transfer photos from SD card to the server database. Videos and photos uploaded on the server can be also sent by e-mail as attachments if their total dimension doesn't exceeded a prefixed dimension of 25 MB. To do this, we have created a class called ServerForm that extends the Windows.Form class, to make the application user-friendly by giving it a minimum GUI. In the form, on the top left there is a box in which the user can see the video streaming sent by the device, while on the left there is another box containing a photo gallery or a video gallery, depending on which one of the two buttons on the top-screen the user selects. Finally on the bottom left there is the form used to send photos and/or videos by email to a preselected e-mail address.



Everything in the program starts when the ServerForm's constructor is called. In it, all the graphical components are initialized and two secondary threads are created to listen for connections from the device. The first thread runs the StartRecordingServer method that initializes a TCP socket, which listens for video streaming from the device with the support of a TCPListener and a TCPClient objects. When the user taps the video button on the device's main screen, the new connection is accepted and everything is prepared for the upload of the video streaming from the client's camera to the server. In particular the server receives one frame of the video at time, which is saved in a List object; when the last image is added in the list, the video, in avi format, is created using the AsyncVideoStreamWrapper object instantiated by the AviWriter object's AddVideoStream method with an RGBVideoEncoder in Async mode. At this point the video created is added in the video gallery, where the user can play, remove or send it by e-mail. The second thread runs the StartUpdateServer method that initializes a TCP socket which listens for the upload of the images from the device's SD card to the server's database.

When the user taps the corresponding button on the device's main screen, a new connection is accepted by the thread

and starts to receive the images. These are added in a list and at the transfer's end a new SqlConnection is created to insert all the images in the database. The inserts are done in a SqlTransaction to avoid any inconsistencies in the database. The images are also added in the image gallery; in this box, there are many PanelItem objects as many images and each of these contains a picture with its path name and a check box for the selection. The images are taken from the database using the ImageTableAdapter object that provides communication between the application and the database by executing SQL statements. In this case the adapter gets the images from the database and stores them in a DataTable of the MyImageDataSet object; the images are taken from this object and added in panel.
Sending an email with photos and videos as attachments is the last functionality implemented by the server application. This operation is managed by Emailer object, specifically by its SendMessageWithAttachment method that gets as parameters the receiver's e-mail, the subject, the message and an ArrayList of files pathnames that have to be send by e-mail. In our case, the ArrayList contains only a pathname which links to a zip file with all the photos and videos to be sent. The zip file containing all the images selected from the database is created by another thread, which adds them in a folder zipped by the ZipFile.CreateFromDirectory method.

Package ProgettoAlbertengo (on FEZ Spider)

Classes

- class **MyBitmap**
- class **Program**
- class **Resources**

Typedefs

- using **GT** = Gadgeteer
- using **GTM** = Gadgeteer.Modules

Package Server

Namespaces

- package **MyImageDataSetTableAdapters**
- package **Properties**

Classes

- class **Emailer**
- class **MyImageDataSet**
- class **Program**
- class **PanelItem**
- class **ServerForm**

Package Server.MyImageDataSetTableAdapters

Classes

- class **ImagesTableAdapter**
- class **TableAdapterManager**

Package Server.Properties

Classes

- class **Resources**
- class **Settings**

Third-part Package Pacman.

A support's package that has allowed us to include a bonus part in this project in which the user can play the famous game Pacman. The game is accessible from the main screen by holding down the center button of the joystick. This package, although adapted and made functional by us, was taken from a network open source project. In this manual is not available a detailed description of it.

ProgettoAlbertengo.MyBitmap Class Reference

Public Member Functions

- **MyBitmap** (bool last, Bitmap bmp)

Public Attributes

- Bitmap **bitmap**
- bool **last**

Detailed Description

Internal class used to send frame by frame the video streaming.

Definition at line 1115 of file Program.cs.

Constructor & Destructor Documentation

ProgettoAlbertengo.MyBitmap.MyBitmap (bool *last*, Bitmap *bmp*)

MyBitmap class constructor

Parameters:

last	(bool) if true the bitmap associated is the last of the video stream.
bmp	Bitmap that represents a frame of the video stream.

Definition at line 1125 of file Program.cs.

ProgettoAlbertengo.Program Class Reference

Inherits Program.

Public Member Functions

- void **SetupNet** ()
- void **connectStreamSocket** ()
- void **connectUploadSocket** ()
- void **startUpload** ()
- void **startVideoStreaming** ()
- bool **VerifySDCard** ()
- void **showGallery** ()

Static Public Member Functions

- static void **Main** ()

Properties

- static GHIElectronics.Gadgeteer.FEZSpider **Mainboard** [get, set]

  This property provides access to the Mainboard API. This is normally not necessary for an end user program.

Private Member Functions

- void **ProgramStarted** ()
- void **joystick_JoystickPressed** (Joystick sender, Joystick.JoystickState state)
- void **ethernet_NetworkUp** (GTM.Module.NetworkModule sender, GTM.Module.NetworkModule.NetworkState state)
- void **ethernet_NetworkDown** (GTM.Module.NetworkModule sender, GTM.Module.NetworkModule.NetworkState state)
- void **sdCard_Mounted** (SDCard sender, GT.StorageDevice SDCard)
- void **sdCard_Unmounted** (SDCard sender)
- void **camera_BitmapStreamed** (Camera sender, Bitmap bitmap)
- void **backButton_Pressed** (Button sender, Button.ButtonState state)
- void **confirmButton_Pressed** (Button sender, Button.ButtonState state)
- void **mainWindow_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **gallery_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **gallery_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **pc_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **pc_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **videoStreaming_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **videoStreaming_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **takePhoto_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **takePhoto_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **cancelSave_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **cancelSave_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)

- void **confirmSave_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **confirmSave_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **trashPanel_TouchDown** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **trashPanel_TouchUp** (object sender, Microsoft.SPOT.Input.TouchEventArgs e)
- void **timer_Tick** (GT.Timer timer)
- void **connectEthernet** ()
- void **SetupDisplay** ()
- void **SetupTrashPanel** ()
- void **SetupSavePhotoWindow** ()
- void **MakeTrasparentImg** ()
- void **SetupInitWindow** ()
- void **HideInitButtons** ()
- void **ShowInitButtons** ()
- void **HideSavePhotoButtons** ()
- void **ShowSavePhotoButtons** ()
- void **removeImage** ()
- void **savePicture** ()
- Color **chooseColor** (int j)
- void **InitializeModules** ()

Detailed Description

Main class of software on the Fez Spider. The program has been designed as a finite state machine, each screen on the display corresponds to a different state. Each tap on the display has a different function depending on the screen in which the user is located.

Definition at line 30 of file Program.cs.

Member Function Documentation

void ProgettoAlbertengo.Program.connectStreamSocket ()

Connects the mainboard with the server to trasmit the stream of the camera.

Definition at line 534 of file Program.cs.

void ProgettoAlbertengo.Program.connectUploadSocket ()

Connects the mainboard with the server to move the picture in the SD card to the database's server.

Definition at line 601 of file Program.cs.

static void ProgettoAlbertengo.Program.Main ()[static]

This method runs automatically when the device is powered, and calls ProgramStarted.

Definition at line 62 of file Program.generated.cs.

void ProgettoAlbertengo.Program.showGallery ()

Shows the photo gallery on the display if there is a SD card mounted, otherwise shows a message "Insert SD card". This method is called when the user tap the gallery button on the display.

Definition at line 1036 of file Program.cs.

void ProgettoAlbertengo.Program.ProgramStarted ()[private]

This method is run when the mainboard is powered up or reset. It initializes the display with the main window and creates the events on the SD card and the Ethernet network.

Definition at line 90 of file Program.cs.

void ProgettoAlbertengo.Program.removeImage ()[private]

Removes the picture actually displayed from the SD card. This method is called when the user taps the trash icon on the screen.

Definition at line 931 of file Program.cs.

void ProgettoAlbertengo.Program.savePicture ()[private]

Saves the last picture which is taken whit the camera in the SD card if it is correctly mounted.

Definition at line 960 of file Program.cs.

void ProgettoAlbertengo.Program.SetupNet ()

Initializes with a static IP configuration the Ethernet.

Definition at line 521 of file Program.cs.

void ProgettoAlbertengo.Program.startUpload ()

Starts to move the pictures from the SD card to the server using the connection created previously. This method is called when the user taps the upload button on the main window.

Definition at line 702 of file Program.cs.

void ProgettoAlbertengo.Program.startVideoStreaming ()

Starts to send video streaming to the server using the connection created previously. This method is called when the user taps the video button on the main window.

Definition at line 742 of file Program.cs.

bool ProgettoAlbertengo.Program.VerifySDCard ()

Verify if the SD card is mounted.

Returns:
(bool) true if the SD card is mounted, false otherwise.
Definition at line 1008 of file Program.cs.

Member Data Documentation

Gadgeteer.Modules.GHIElectronics.Button ProgettoAlbertengo.Program.backButton[private]

The Button module using socket 8 of the mainboard.

Definition at line 43 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.Camera ProgettoAlbertengo.Program.camera[private]

The Camera (Premium) module using socket 3 of the mainboard.

Definition at line 22 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.Button ProgettoAlbertengo.Program.confirmButton[private]

The Button module using socket 11 of the mainboard.

Definition at line 40 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.Display_T35 ProgettoAlbertengo.Program.display[private]

The Display_T35 module using sockets 14, 13, 12 and 10 of the mainboard.

Definition at line 19 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.Ethernet_J11D ProgettoAlbertengo.Program.ethernet[private]

The Ethernet_J11D (Premium) module using socket 7 of the mainboard.

Definition at line 37 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.Joystick ProgettoAlbertengo.Program.joystick[private]

The Joystick module using socket 9 of the mainboard.

Definition at line 28 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.MulticolorLed ProgettoAlbertengo.Program.ledNet[private]

The MulticolorLed module using socket 4 of the mainboard.

Definition at line 46 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.MulticolorLed ProgettoAlbertengo.Program.ledSd[private]

The MulticolorLed module using socket * of ledNet.

Definition at line 49 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.SDCard ProgettoAlbertengo.Program.sdCard[private]

The SDCard module using socket 5 of the mainboard.

Definition at line 31 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.UsbClientDP ProgettoAlbertengo.Program.usbClientDP[private]

The UsbClientDP module using socket 1 of the mainboard.

Definition at line 25 of file Program.generated.cs.

Gadgeteer.Modules.GHIElectronics.WiFi_RS21 ProgettoAlbertengo.Program.wifi[private]

The WiFi_RS21 (Premium) module using socket 6 of the mainboard.

Definition at line 34 of file Program.generated.cs.

Property Documentation

new GHIElectronics.Gadgeteer.FEZSpider ProgettoAlbertengo.Program.Mainboard[static], [get], [set], [protected]

This property provides access to the Mainboard API. This is normally not necessary for an end user program.

Definition at line 52 of file Program.generated.cs.

ProgettoAlbertengo.Resources Class Reference

Package Types

- enum **BinaryResources** : short
- enum **FontResources** : short

Static Package Functions

- static Microsoft.SPOT.Font **GetFont** (Resources.FontResources id)
- static byte[] **GetBytes** (Resources.BinaryResources id)

Properties

- static System.Resources.ResourceManager **ResourceManager** [get]

Detailed Description

Definition at line 14 of file Resources.Designer.cs.

Server.ServerForm Class Reference

Inherits Form.

Public Member Functions

- **ServerForm** ()
- void **StartUpdateServer** ()
- void **StartRecordingServer** ()
- void **ClientConnection** (object obj, bool streaming)
- void **UploadImages** (Object obj)
- void **StreamingVideo** (object obj)
- byte[] **imageToByteArray** (System.Drawing.Image imageIn)
- string **createZipFile** ()
- void **btnSend_Click** (object sender, EventArgs e)
- void **checkBox1_CheckedChanged** (object sender, EventArgs e)

Static Public Member Functions

- static Image **byteArrayToImage** (byte[] byteArrayIn)

Protected Member Functions

- override void **Dispose** (bool disposing)
  Frees the resources in use.

Private Member Functions

- string **GetNextFileName** ()
- void **GetByteArray** (Bitmap image, byte[] buffer)
- void **showSendButton** ()
- void **hideSendButton** ()
- string **getConvertedTotalAttachmentSize** ()
- long **getItemSize** (string fileName)
- void **btnCancel_Click** (object sender, EventArgs e)
- void **button1_Click** (object sender, EventArgs e)
- void **button5_Click** (object sender, EventArgs e)
- void **button2_Click** (object sender, EventArgs e)
- void **InitializeComponent** ()

Detailed Description

Server side class that controls and manages request coming from the client.

Definition at line 28 of file ServerForm.cs.

Constructor & Destructor Documentation

Server.ServerForm.ServerForm ()

ServerForm's constructor, initialize graphical components and create two threads that manage video recording and upload of images from the client.

Definition at line 46 of file ServerForm.cs.

Member Function Documentation

void Server.ServerForm.btnSend_Click (object *sender*, EventArgs *e*)

Event generated by pressing the send button. It provides to generate a new thread that send an email to the specified address (with or without attachments).

Parameters:

sender	Button "Send" that the user pressed.
e	List of any event data.

Definition at line 510 of file ServerForm.cs.

void Server.ServerForm.checkBox1_CheckedChanged (object *sender*, EventArgs *e*)

Event on the check box. It start when the user checks or unchecks one image or one video in the check box. In the first case, the method inserts its path in the global array "attachment" to include it in the zipped folder that the user would like to send via email, otherwise it provides to remove the same path from the array.

Parameters:

sender	Check box in which the selection was made.
e	List of any event data.

Definition at line 614 of file ServerForm.cs.

void Server.ServerForm.ClientConnection (object *obj*, bool *streaming*)

Recognizes the type of request and starts the correct operation to satisfy it.

Parameters:

obj	TCPClient object that identifies the client connected.
streaming	(bool) if true the established connection is dedicated to video streaming, else to photo uploading.

Definition at line 137 of file ServerForm.cs.

string Server.ServerForm.createZipFile ()

Create a zipped folder that contains the files to be sent via e-mail as attachments. The paths of these files are saved in the global array "attachments".

Returns:

The name of the zipped folder's path as a string.

Definition at line 435 of file ServerForm.cs.

override void Server.ServerForm.Dispose (bool *disposing*)[protected]

Frees the resources in use.

Parameters:

disposing	(bool) if true, resources in use should be eliminated.

Definition at line 14 of file ServerForm.Designer.cs.

void Server.ServerForm.StartRecordingServer ()

Initialize TCP socket that listens for video streaming from the client. With support of a TCPListener and a TCPClient objects preparing the upload of the video streaming from the client's camera to the server's database.

Definition at line 102 of file ServerForm.cs.

void Server.ServerForm.StartUpdateServer ()

Initialize TCP socket that listens for data exchange with the client. With support of a TCPListener and a TCPClient objects preparing the upload of the images from the client's SD card to the server's database.

Definition at line 65 of file ServerForm.cs.

void Server.ServerForm.StreamingVideo (object *obj*)

Manages the upload of the video streaming from the client's camera to the server's database.

Parameters:

obj	NetworkStream that identifies the source of the data flow.

Definition at line 285 of file ServerForm.cs.

void Server.ServerForm.UploadImages (Object *obj*)

Allows the upload of the images from the client's SD card to the server's database.

Parameters:

obj	NetworkStream that identifies the source of the data flow.

Definition at line 157 of file ServerForm.cs.

Server.Properties.Settings Class Reference

Inherits ApplicationSettingsBase.

Properties

- static **Settings Default** [get]
- string **MyDBConnectionString** [get]
- string **MyDBConnectionString1** [get]

Detailed Description

Definition at line 16 of file Settings.Designer.cs.

Server.Emailer Class Reference

Static Public Member Functions

- static string **SendMessage** (string sendTo, string sendFrom, string sendSubject, string sendMessage)
- static string **SendMessageWithAttachment** (string sendTo, string sendFrom, string sendSubject, string sendMessage, ArrayList attachments)
- static bool **ValidateEmailAddress** (string emailAddress)

Detailed Description

Definition at line 16 of file Emailer.cs.

Member Function Documentation

static string Server.Emailer.SendMessage (string *sendTo*, string *sendFrom*, string *sendSubject*, string *sendMessage*)[static]

Transmit an email message to a recipient without any attachments

Parameters:

sendTo	Recipient Email Address
sendFrom	Sender Email Address
sendSubject	Subject Line Describing Message
sendMessage	The Email Message Body

Returns:

Status Message as String

Definition at line 28 of file Emailer.cs.

static string Server.Emailer.SendMessageWithAttachment (string *sendTo*, string *sendFrom*, string *sendSubject*, string *sendMessage*, ArrayList *attachments*)[static]

Transmit an email message with attachments

Parameters:

sendTo	Recipient Email Address
sendFrom	Sender Email Address
sendSubject	Subject Line Describing Message
sendMessage	The Email Message Body
attachments	A string array pointing to the location of each attachment

Returns:

Status Message as String

Definition at line 77 of file Emailer.cs.

static bool Server.Emailer.ValidateEmailAddress (string *emailAddress*)[static]

Confirms that an email address is valid in format

Parameters:

emailAddress	Full email address to validate

Returns:

True if email address is valid

Definition at line 131 of file Emailer.cs.

Server.MyImageDataSet Class Reference

Inherits DataSet.

Classes

- class **ImagesDataTable**
- class **ImagesRow**
- class **ImagesRowChangeEvent**

Row event argument class

Public Member Functions

- override global::System.Data.DataSet **Clone** ()
- delegate void **ImagesRowChangeEventHandler** (object sender, **ImagesRowChangeEvent** e)

Static Public Member Functions

- static global::System.Xml.Schema.XmlSchemaComplexType **GetTypedDataSetSchema** (global::System.Xml.Schema.XmlSchemaSet xs)

Protected Member Functions

- **MyImageDataSet** (global::System.Runtime.Serialization.SerializationInfo info, global::System.Runtime.Serialization.StreamingContext context)
- override void **InitializeDerivedDataSet** ()
- override bool **ShouldSerializeTables** ()
- override bool **ShouldSerializeRelations** ()
- override void **ReadXmlSerializable** (global::System.Xml.XmlReader reader)
- override
- global::System.Xml.Schema.XmlSchema **GetSchemaSerializable** ()

Package Functions

- void **InitVars** ()
- void **InitVars** (bool initTable)

Properties

- **ImagesDataTable Images** [get]
- override global::System.Data.SchemaSerializationMode **SchemaSerializationMode** [get, set]
- new global::System.Data.DataTableCollection **Tables** [get]
- new global::System.Data.DataRelationCollection **Relations** [get]

Private Member Functions

- void **InitClass** ()
- bool **ShouldSerializeImages** ()
- void **SchemaChanged** (object sender, global::System.ComponentModel.CollectionChangeEventArgs e)

Detailed Description

Represents a strongly typed in-memory cache of data.

Definition at line 25 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSet.ImagesDataTable Class Reference

Inherits TypedTableBase< ImagesRow >.

Public Member Functions

- void **AddImagesRow** (**ImagesRow** row)
- **ImagesRow AddImagesRow** (byte[] image)
- **ImagesRow FindByid** (long id)
- override
- global::System.Data.DataTable **Clone** ()
- **ImagesRow NewImagesRow** ()
- void **RemoveImagesRow** (**ImagesRow** row)

Static Public Member Functions

- static
- global::System.Xml.Schema.XmlSchemaComplexType **GetTypedTableSchema** (global::System.Xml.Schema.XmlSchemaSet xs)

Protected Member Functions

- **ImagesDataTable** (global::System.Runtime.Serialization.SerializationInfo info, global::System.Runtime.Serialization.StreamingContext context)
- override
- global::System.Data.DataTable **CreateInstance** ()
- override
- global::System.Data.DataRow **NewRowFromBuilder** (global::System.Data.DataRowBuilder builder)
- override global::System.Type **GetRowType** ()
- override void **OnRowChanged** (global::System.Data.DataRowChangeEventArgs e)
- override void **OnRowChanging** (global::System.Data.DataRowChangeEventArgs e)
- override void **OnRowDeleted** (global::System.Data.DataRowChangeEventArgs e)
- override void **OnRowDeleting** (global::System.Data.DataRowChangeEventArgs e)

Package Functions

- **ImagesDataTable** (global::System.Data.DataTable table)
- void **InitVars** ()

Properties

- global::System.Data.DataColumn **idColumn** [get]
- global::System.Data.DataColumn **imageColumn** [get]
- int **Count** [get]
- **ImagesRow this[int index]** [get]

Events

- ImagesRowChangeEventHandler **ImagesRowChanging**
- ImagesRowChangeEventHandler **ImagesRowChanged**

- ImagesRowChangeEventHandler **ImagesRowDeleting**
- ImagesRowChangeEventHandler **ImagesRowDeleted**

Private Member Functions

- void **InitClass** ()

Detailed Description

Represents the strongly named DataTable class.

Definition at line 280 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSet.ImagesRow Class Reference

Inherits DataRow.

Package Functions

- **ImagesRow** (global::System.Data.DataRowBuilder rb)

Properties

- long **id** [get, set]
- byte[] **image** [get, set]

Detailed Description

Represents strongly named DataRow class.

Definition at line 555 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSetTableAdapters.ImagesTableAdapter Class Reference

Inherits Component.

Public Member Functions

- virtual int **Fill** (**MyImageDataSet.ImagesDataTable** dataTable)
- virtual **MyImageDataSet.ImagesDataTable GetData** ()
- virtual **MyImageDataSet.ImagesDataTable GetImageById** (long id)
- virtual int **Update** (**MyImageDataSet.ImagesDataTable** dataTable)
- virtual int **Update** (**MyImageDataSet** dataSet)
- virtual int **Update** (global::System.Data.DataRow dataRow)
- virtual int **Update** (global::System.Data.DataRow[] dataRows)
- virtual int **Delete** (long Original_id)
- virtual int **Insert** (byte[] image)
- virtual int **Update** (byte[] image, long Original_id, long id)
- virtual int **Update** (byte[] image, long Original_id)

Properties

- global::System.Data.SqlClient.SqlDataAdapter **Adapter** [get]
- global::System.Data.SqlClient.SqlConnection **Connection** [get, set]
- global::System.Data.SqlClient.SqlTransaction **Transaction** [get, set]
- global::System.Data.SqlClient.SqlCommand[] **CommandCollection** [get]
- bool **ClearBeforeFill** [get, set]

Private Member Functions

- void **InitAdapter** ()
- void **InitConnection** ()
- void **InitCommandCollection** ()

Detailed Description

Represents the connection and commands used to retrieve and save data.

Definition at line 636 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSetTableAdapters.TableAdapterManager Class Reference

Inherits Component.

Classes

- class **SelfReferenceComparer**

Used to sort self-referenced table's rows Public Types

- enum **UpdateOrderOption** { **InsertUpdateDelete** = 0, **UpdateInsertDelete** = 1 }

Update Order Option Public Member Functions

- virtual int **UpdateAll** (**MyImageDataSet** dataSet)
  Update all changes to the dataset.

Protected Member Functions

- virtual void **SortSelfReferenceRows** (global::System.Data.DataRow[] rows, global::System.Data.DataRelation relation, bool childFirst)
- virtual bool **MatchTableAdapterConnection** (global::System.Data.IDbConnection inputConnection)

Properties

- **UpdateOrderOption UpdateOrder** [get, set]
- **ImagesTableAdapter ImagesTableAdapter** [get, set]
- bool **BackupDataSetBeforeUpdate** [get, set]
- global::System.Data.IDbConnection **Connection** [get, set]
- int **TableAdapterInstanceCount** [get]

Private Member Functions

- int **UpdateUpdatedRows** (**MyImageDataSet** dataSet, global::System.Collections.Generic.List< global::System.Data.DataRow > allChangedRows, global::System.Collections.Generic.List< global::System.Data.DataRow > allAddedRows)
  Update rows in top-down order.
- int **UpdateInsertedRows** (**MyImageDataSet** dataSet, global::System.Collections.Generic.List< global::System.Data.DataRow > allAddedRows)
  Insert rows in top-down order.
- int **UpdateDeletedRows** (**MyImageDataSet** dataSet, global::System.Collections.Generic.List< global::System.Data.DataRow > allChangedRows)
  Delete rows in bottom-up order.
- global::System.Data.DataRow[] **GetRealUpdatedRows** (global::System.Data.DataRow[] updatedRows, global::System.Collections.Generic.List< global::System.Data.DataRow > allAddedRows)
  Remove inserted rows that become updated rows after calling TableAdapter.Update(inserted rows) first

Detailed Description

TableAdapterManager is used to coordinate TableAdapters in the dataset to enable Hierarchical Update scenarios

Definition at line 954 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSetTableAdapters.TableAdapterManager.SelfReferenceComparer Class Reference

Inherits object, and IComparer< global.System.Data.DataRow >.

Public Member Functions

- int **Compare** (global::System.Data.DataRow row1, global::System.Data.DataRow row2)

Package Functions

- **SelfReferenceComparer** (global::System.Data.DataRelation relation, bool childFirst)

Private Member Functions

- global::System.Data.DataRow **GetRoot** (global::System.Data.DataRow row, out int distance)

Detailed Description

Used to sort self-referenced table's rows

Definition at line 1281 of file MyImageDataSet.Designer.cs.

Server.MyImageDataSet.ImagesRowChangeEvent Class Reference

Inherits SystemEventArgs.

Public Member Functions

- **ImagesRowChangeEvent** (**ImagesRow** row, global::System.Data.DataRowAction action)

Properties

- **ImagesRow Row** [get]
- global::System.Data.DataRowAction **Action** [get]

Detailed Description

Row event argument class

Definition at line 593 of file MyImageDataSet.Designer.cs.

Server.PanelItem Class Reference

Inherits Panel.

Public Member Functions

- **PanelItem** (string text, Image bmp)

Properties

- CheckBox **checkBox1** [get, set]
- Label **label1** [get, set]
- Button **button1** [get, set]
- Button **button2** [get, set]

Private Member Functions

- void **InitializeComponent** ()
- void **showButton** (object sender, EventArgs e)
- void **hideButton** (object sender, EventArgs e)
- void **button1_Click** (object sender, EventArgs e)

Detailed Description

Definition at line 13 of file PanelItem.cs.

Server.Program Class Reference

Static Private Member Functions

- static void **Main** ()

Detailed Description

Definition at line 9 of file Program.cs.

Member Function Documentation

static void Server.Program.Main ()[static], [private]

Principal program's access point.

Definition at line 15 of file Program.cs.

Server.Properties.Resources Class Reference

Properties

- global::System.Resources.ResourceManager **ResourceManager** [get]
- global::System.Globalization.CultureInfo **Culture** [get, set]

Detailed Description

Strongly typed resource class for finding localized strings.

Definition at line 25 of file Resources.Designer.cs.

Property Documentation

global.System.Globalization.CultureInfo Server.Properties.Resources.Culture[static], [get], [set], [package]

Overrides current thread's CurrentUICulture property for all resource searches performed using this strongly typed resource class.

Definition at line 54 of file Resources.Designer.cs.

global.System.Resources.ResourceManager Server.Properties.Resources.ResourceManager[static], [get], [package]

Returns the cached ResourceManager instance used by this class.

Definition at line 39 of file Resources.Designer.cs.

Index

INDEX

Introduction 1
FEZ Spider part 1
Server part 4
Packages 6

Class List

ProgettoAlbertengo.MyBitmap 7
ProgettoAlbertengo.Program 8
ProgettoAlbertengo.Resources 14
Server.ServerForm 15
Server.Properties.Settings 19
Server.Emailer 20
Server.MyImageDataSet 22
Server.PanelItem 32
Server.Program 33
Server.Properties.Resources 34