PICSIM

DEVELOPER'S GUIDE

2007

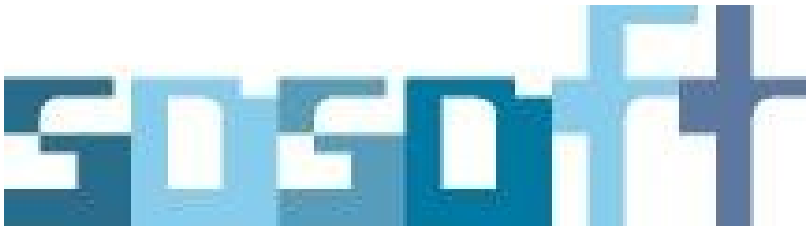

TABLE OF CONTENTS

EDITOR CLASS

Editor Class : **public** QTextEdit

PicSim has all the necessary features of a simple text editor. The editor class has the following members:

public members:

Editor(QWidget *parent = 0, const char *name = 0)
void newFile()
bool open()
bool openFile(const QString &fileName)
bool save()
bool saveAs()
QSize sizeHint() const
QString getcurFile(void)
QString strippedName(const QString &fullFileName)

signals:

void message(const QString &fileName, int delay)

protected methods:

void closeEvent(QCloseEvent *event)

private methods:

bool maybeSave()
void saveFile(const QString &fileName)
void setCurrentFile(const QString &fileName)
bool readFile(const QString &fileName)
bool writeFile(const QString &fileName)

private member data:

QString curFile
bool isUntitled
QString fileFilters

Detailed Description of Editor Class:

Editor::Editor(QWidget *parent, const char *name)

: QTextEdit(parent, name)

Constructs an editor as the child widget of the widget *parent*, with name *name*. Derived from QtextEdit class.

Editor::newFile()

Opens an empty file named document*x*.c where x is the order of the new file opened.

Editor::open()

Opens an “open file dialog” and calls openFile.

Editor::openFile(const QString &fileName)

Opens the file with the given fileName.

Editor::save()

If the file is named, saves that file. Else opens saveAs dialog.

Editor::saveAs()

Opens saveAs dialog and saves the file with the name user inserts.

Editor::closeEvent(QCloseEvent *event)

Closes the file.

Editor::maybeSave()

Checks if the file is changed before closing and asks if the user wants to save it.

Editor::saveFile(const QString &fileName)

Writes into the file with the file name and called by save and saveAs methods.

Editor::setCurrentFile(const QString &fileName)

Sets the file last clicked as the current file.

Editor::sizeHint() const

Gives the size of current file.

Editor::readFile(const QString &fileName)

Helps to read the file fileName.

Editor::writeFile(const QString &fileName)

Helps to write into the file fileName.

Editor::strippedName(const QString &fullFileName)

Returns the strippedName of the file which is given with its full path as fullFileName.

Editor::getcurFile(void)

Returns the name of current file.

MAIN WINDOW CLASS

MainWindow Class: QMainWindow

This class has all the features of a main form which are needed in PicSim such as menubar and toolbar.

Public methods:

MainWindow(QWidget *parent = 0, const char *name = 0)

Public slots:

void newFile()

void openFile(const QString &fileName)

Protected methods:

void closeEvent(QCloseEvent *event)

Private slots:

void open()

void save()

void saveAs()

void cut()

void copy()

void paste()

void del()

void about()

void updateMenus()

void activateWindow(int param)

void copyAvailable(bool available)

void updateModIndicator()

void compileWindowOpen()

void compileWindowClose()

void simulateWindowOpen()

void debugWindowOpen()

void uploadToBoard()

void uploadToBoardClose()

void deleteBoard()

void deleteBoardClose()

void readfromstdout()

void readfromstdoutodyssey()

void readfromstdoutodysseyupload()

Private methods:

void createActions()

void createMenus()

void createWindowsMenu()

void createToolBars()

void createStatusBar()

Editor *createEditor()

Editor *activeEditor()

Private member data:

QString directory

QWorkspace *workspace

QLabel *readyLabel

QLabel *modLabel

QWidgetList windows

QProcess *dir

QProcess *sdcc

QProcess *odysseyrem

QProcess *odysseywri

QProcess *debugprocess

QMessageBox *compmess

QMessageBox *dellmess

QMessageBox *upmess

QMessageBox *debugmess

QPopupMenu *fileMenu

QPopupMenu *editMenu

QPopupMenu *windowsMenu

QPopupMenu *helpMenu

QPopupMenu *projectMenu

QPopupMenu *debugMenu

QToolBar *fileToolBar

QToolBar *editToolBar

Private Actions:

QAction *newAct

QAction *openAct

QAction *saveAct

QAction *saveAsAct

QAction *exitAct

QAction *cutAct

QAction *copyAct

QAction *pasteAct

QAction *deleteAct

QAction *closeAct

QAction *closeAllAct

QAction *tileAct

QAction *cascadeAct

QAction *nextAct

QAction *previousAct

QAction *aboutAct

QAction *aboutQtAct

QAction *compileAct

QAction *simulateAct

QAction *debugAct

QAction *uploadAct

QAction *deleteBoardAct

Detailed Description of MainWindow Class:

MainWindow(QWidget *parent = 0, const char *name = 0)

Constructs the mainwindow and connects necessary signals to corresponding slots.

void newFile()

Calls the necessary functions of the editor to create a new file.

void openFile(const QString &fileName)

Calls the functions of editor to open an existing file.

void closeEvent(QCloseEvent *event)

Closes PicSim.

void open()

Creates the first empty editor and opens the empty file.

void save()

Calls editor class’s save function.

void saveAs()

Calls editor class's saveAs function.

void cut()

Cuts the selected text and coppies it to the clipboard.

void copy()

Copies the selected text to the clipboard.

void paste()

Pastes the text on the clipboard starting from the cursor.

void del()

Deletes the selected text.

void about()

Displays about message of PicSim.

void updateMenus()

Sets the menus of PicSim as enabled.

void activateWindow(int param)

Activates the editor window with the id *param.*

void copyAvailable(bool available)

Sets the copy and cut actions as enabled if some text is selected.

void updateModIndicator()

Updates the current mode of active editor.

void compileWindowOpen()

Compiles the c source code written in the active editor. Uses sdcc as a qprocess and writes the standard output of sdcc into the *compile message* message box.

void compileWindowClose()

Closes the compilation message box if *ok* button is pressed.

void simulateWindowOpen()

Opens the simulation window, which is a child widget of main window.

void debugWindowOpen()

Opens debugger watch window,which is a child widget of the main window.

void uploadToBoard()

Uploads the hex file created by the compiler to the PicDev board using odyssey as a qprocess. Uses parallel port and shows the standard output of odyssey in a message box.

void uploadToBoardClose()

Closes the *upload message* message box when *ok* button is pressed.

void deleteBoard()

Erases the PicDev board and shows the message of odyssey in a message box.

void deleteBoardClose()

Closes the *delete board* message box when *ok* button pressed.

void readfromstdout()

Reads the message of sdcc from standard output.

void readfromstdoutodyssey()

Reads the delete message of odyssey from standard output.

void readfromstdoutodysseyupload()

Reads the programming message of odyssey from standard output.

void createActions()

Allocates memory for new, open, save, saveAs, exit, cut, copy, paste, delete, close, closeAll, tile, cascade, next, previous, about, aboutqt, compile, debug, upload and simulate actions and connect them to the corresponding slots.

void createMenus()

Allocates memory for file, windows, edit, project, help menus.

void createWindowsMenu()

Creates windows menu.

void createToolBars()

Creates the toolbar menu.

void createStatusBar()

Creates the status bar.

Editor *createEditor()

Constructs an editor class object.

Editor *activeEditor()

Returns the editor which is currently active.

Private Actions:

The actions which send the necessary signals to activate the corresponding slots.

SIMULATOR CLASS

Simulator Class: **public** QDialog

PicSim has all the necessary features of a simple simulator. The simulator class has the following members:

Public members:

simulator(QWidget* parent = 0, const char* name = 0, bool modal = FALSE, WFlags fl = 0);

void sevensegment()

void leds()

void startsimulation()

KLed* kLedX(*)

QPushButton* pushButtonX(*)

QTextEdit* textEdit1

bool ispoweron

Public slots:

void poweron()

Protected:

QVBoxLayout* layout19

QHBoxLayout* layoutX(*)

Protected slots:

virtual void languageChange()

(*) stands for similar variable names are defined for different extensions.

e.g. KLed* kLed3;

KLed* kLed5; etc.

Detailed Description of Simulator Class:

simulator::simulator(QWidget* parent, const char* name, bool modal, WFlags fl)

: QDialog(parent, name, modal, fl)

Constructs a simulator as the child widget of the widget *parent*, with name *name*. Derived from QDialog class.

simulator::languageChange()

Sets the strings of the subwidgets using the current language.

simulator::poweron()

Checks whether power button is pressed.

simulator::leds()

Turns on /off the leds depending on the 2nd bit of PORTA and bits of PORTD

simulator::sevensegment()

Changes the status of the seven segment displays depending on the first three bits of PORTE and bits of PORTD

simulator::startsimulation()

Starts simulation when the power button is pressed and calls the needed functions (sevensegment(), leds(),etc.).

DATA MODULE

Regs Class:

It has two public members, where ‘name’ is the name of the registers in the banks and ‘value’ keeps the data at that register.

There are two arrays called “bank01” and “bank23” of type ‘regs’, which corresponds to our 4 banks in 16F877 register file map.

void initregs()

Initiates all values of registers to 0 and assigns the names of the special function registers.

int addresstoint(char a, char b)

Takes a hexadecimal number as if two characters and returns to the corresponding decimal number to be used as array index in bank applications.

void tobinary(char *result, int value)

Takes a decimal number and converts it into a binary number, stores the result in a character array, whose pointer to its first element is given as an argument to the function, ‘result’

void tobinary2(int value)

Takes a decimal number and converts it into a binary number, stores the result in a character array, ‘binary1’.

int returnarrayindex()

Gives the index of the array ‘return_array’ where the value at that index is -1.

int bintodec(char * a)

Takes a character pointer of where the binary number is stored, and returns its decimal equivalent.

int hextodec(char a,char b)

Takes a hexadecimal number as two characters and returns its decimal equivalent.

int gotohextodec(char a,char b,char c)

Takes 3 digit hexadecimal number and returns its decimal form (where the first bit –from left- of a’s binary equivalent is discarded) with the help of hextodec function. This function is used to determine which instruction comes next when a goto instruction is executed.

int callhextodec(char a,char b,char c)

Takes 3 digit hexadecimal number and returns its decimal form with the help of hextodec function. This function is used to determine which instruction comes next when a call instruction is executed.

HEX MODULE

void hex()

Takes all instructions from the Hex file (i.e. except for the first line and the last line and the other characters not corresponding an instruction) and keeps them in an integer array.

void hexfunc(int k)

Takes the index of hexdata as an argument, k, and generates the instruction defined in the Hex file, i.e. makes the necessary changes in the content of the registers. This function will work in parallel with the simulation module.

void callerfunc(int k)

Takes the index of the hexdata array as an argument, if the banks are not initiated it calls the initregs function, then if hexdata is not constructed yet it calls hex function and finally it calls hexfunc() to be ready for simulation. This function is called in startsimulation function of the simulator.

WATCH WINDOW CLASS

class watchwindow : public Qwidget

This class has all the necessary functions to construct a watch table for the user to see the content of the special function registers. The watch window class has the following members:

Public members:

watchwindow(QWidget* parent = 0, const char* name = 0, WFlags fl = 0)

QPushButton* closebutton

QPushButton* continuebutton

QPushButton* next

QTable* watchtable

Public slots:

void gotoend()

void stepbystep()

Protected slots:

virtual void languageChange()

Detailed Description of Watch Window Class:

watchwindow::watchwindow(QWidget* parent, const char* name, WFlags fl)

: QWidget(parent, name, fl)

Constructs the watchwindow as a child of ‘parent ’ widget, with the name ‘name’ and widget flags set to ‘f’.

void watchwindow::stepbystep()

This function is used to show the register content instruction by instruction.

void watchwindow::gotoend()

This function is used to show the register content continuously.

void watchwindow::languageChange()

Sets the strings of the subwidgets using the current language.

QString watchwindow::tobinarydebug(int value)

This function is used to convert integer value to its binary equivaent, to be used in further implementations for showing the register content in binary form.

QString watchwindow::hexecevir(int value)

This function is used to convert integer value to its hexadecimal equivaent, to be used in further implementations for showing the register content in hexadecimal form.

void watchwindow::regcontent(void)

This function is used to show the register content in orm.