# DataGridView动态生成

DataGridViewEdit^ mDataGridView;

Void GF\_Main::initDataGridView(void) {

mDataGridView = gcnew DataGridViewEdit(toolStripButton\_Hex->Checked);

mDataGridView->Dock = DockStyle::Fill;

mDataGridView->EditingControlShowing += gcnew DataGridViewEditingControlShowingEventHandler(this, &GF\_Main::dataGridView1\_EditingControlShowing);

mDataGridView->OnRegisterValueChanged += gcnew DataGridViewEdit::registerValueChangeHandler(this, &GF\_Main::dataGridView\_RegisterAdjust);

mDataGridView->CellEndEdit += gcnew DataGridViewCellEventHandler(this, &GF\_Main::dataGridView1\_CellEndEdit);

mDataGridView->CellClick += gcnew DataGridViewCellEventHandler(this, &GF\_Main::dataGridView1\_CellClick);

mDataGridView->CellValueChanged += gcnew DataGridViewCellEventHandler(this, &GF\_Main::dataGridView1\_CellValueChanged);

mDataGridView->CellMouseEnter += gcnew DataGridViewCellEventHandler(this, &GF\_Main::dataGridView1\_CellMouseEnter);

panel\_DataGridView->Controls->Add(mDataGridView);

mDataTable = gcnew DataTable;

// automatically generate the DataGridView columns.

mDataGridView->AutoGenerateColumns = true;

// full row select

// mDataGridView->SelectionMode = DataGridViewSelectionMode::FullRowSelect;

// set up the data source.

bindingSource1->DataSource = mDataTable;

mDataGridView->DataSource = bindingSource1;

bindingNavigator1->BindingSource = bindingSource1;

// automatically resize the visible rows.

mDataGridView->AutoSizeRowsMode = DataGridViewAutoSizeRowsMode::DisplayedCells;

mDataGridView->AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode::Fill;

// disable edit by click

// mDataGridView->EditMode = DataGridViewEditMode::EditProgrammatically;

// only vertical scroll bar

mDataGridView->ScrollBars = ScrollBars::Vertical;

// hide row header

mDataGridView->RowHeadersVisible = false;

// forbid add new rows

mDataGridView->AllowUserToAddRows = false;

array<Object^, 2>^ columns = gcnew array<Object^, 2>{

// Name Type Visible

{ COL\_NAME, String::typeid, true },

{ COL\_VALUE, String::typeid, true },

{ COL\_WIDTH, String::typeid, true },

{ COL\_R\_W, String::typeid, true },

{ COL\_REMARK, String::typeid, true },

{ COL\_SELECT, String::typeid, true },

{ COL\_PAGENAME, String::typeid, true },

{ COL\_INDEX, Int32::typeid, false },

};

Table::makeDataTable(mDataGridView, mDataTable, columns);

for each (DataGridViewColumn^ column in mDataGridView->Columns) {

if (column->Name == COL\_VALUE) {

mDataGridView->Columns[COL\_VALUE]->CellTemplate->Style->BackColor = Color::Wheat;

} else {

column->ReadOnly = true;

}

}

}

# 获取选择单元格的行

//下面是获取当前行

DataGridViewRow^ viewRow = mDataGridView->CurrentCell->OwningRow;

或者

DataGridViewRow^ viewRow = dataGridView->Rows[e->RowIndex];

//下面是读取指定行中某列

String^ pageName = viewRow->Cells[COL\_PAGENAME]->Value->ToString();

// 下面是判断是不是数值（应该是十六进制）

String^ sValue = viewRow->Cells[COL\_VALUE]->Value->ToString();

UInt32 value;

if (mDataGridView->hexStyle) {

value = UInt32::Parse(sValue, NumberStyles::HexNumber);

} else {

value = UInt32::Parse(sValue);

}

# 数据表合并：

DataTable^ dtAddresses = gcnew DataTable(REG\_ADDR\_TABLE);

array<Object^, 2>^ columnsAddresses = gcnew array<Object^, 2>{

{ REG\_ADDR\_REG, String::typeid },

{ REG\_ADDR\_VALUE, String::typeid },

{ REG\_ADDR\_BIT\_HIGH, String::typeid },

{ REG\_ADDR\_BIT\_LOW, String::typeid },

};

Table::makeDataTable(dtAddresses, columnsAddresses);

ds->Tables->Add(dtAddresses);

//下面是创建数据表之间的关系

// create relation

DataRelation^ drPageRule = gcnew DataRelation("", dtPages->Columns[PAGE\_NAME],

dtRules->Columns[RULES\_PAGE]);

drPageRule->Nested = true;

ds->Relations->Add(drPageRule);

另外还有：

Table::autoResize(mDataGridView, true);//这个用于读操作的时候使用

# 打开/关闭子窗体前进行判断是否已打开

Void GF\_Main::tunerSettingToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

ToolStripMenuItem^ menuItem = (ToolStripMenuItem^) sender;

menuItem->Checked = !menuItem->Checked;

if (menuItem->Checked) {

formTunerOpen();

} else {

formTunerClose();

}

}

Void GF\_Main::formTunerOpen(void) {

ConsoleU::writeLine("open tuner setting form", ConsoleU::Level::Normal);

if (!isFormOpen(mFormTuner)) {

mDockTuner->position = DockU::Position::MiddleRight;

mFormTuner = gcnew GF\_Tuner(mTuner, mDockTuner);

mFormTuner->Closed += gcnew EventHandler(this, &GF\_Main::subformClosed);

this->AddOwnedForm(mFormTuner);

}

mFormTuner->Show();

tunerSettingToolStripMenuItem->Checked = true;

}

Void GF\_Main::formTunerClose(void) {

if (isFormOpen(mFormTuner)) {

mFormTuner->Close();

this->RemoveOwnedForm(mFormTuner);

ConsoleU::writeLine("close tuner setting form", ConsoleU::Level::Normal);

}

}

Void GF\_Main::subformClosed(System::Object^ sender, System::EventArgs^ e) {

if (sender == mFormTuner) {

tunerSettingToolStripMenuItem->Checked = false;

}

}

# XML写入

Void GF\_Main::toolStripMenuItem\_SaveAs\_Click(System::Object^ sender, System::EventArgs^ e) {

Chip^ chip = mChips->current;

if (chip == nullptr) return;

saveFileDialog1->InitialDirectory = String::Format("{0}/{1}",

Directory::GetCurrentDirectory(), PATH\_CONFIG\_CHIPS);

saveFileDialog1->FileName = String::Format("{0}-{1}",

Path::GetFileNameWithoutExtension(chip->config), DateTime::Now.ToString("yyyy-MM-dd-HH-mm-ss"));

if (saveFileDialog1->ShowDialog() == ::DialogResult::OK) {

chip->save(saveFileDialog1->FileName);

}

}

Int32 H2Xml::save(String^ fileName) {

// save to XML file

XmlU::Write(mDataSet, fileName);//这个在Gsoft里面

ConsoleU::writeLine(String::Format("h2xml file: {0}", fileName), ConsoleU::Level::Info);

return 0;

}

# XML读取

Int32 Chips::load(String^ fileName) {

String^ fileExtension = Path::GetExtension(fileName);

if (fileExtension != ".xml") return -1;

if (!File::Exists(fileName)) {

return -2;

}

DataSet^ ds = gcnew DataSet;

ds->ReadXml(fileName);

DataTable^ dtChips = ds->Tables[CHIP\_TABLE\_CHIP];

if (dtChips == nullptr) return -3;

DataRowCollection^ rowsChips = dtChips->Rows;

if (rowsChips == nullptr) return -4;

// clear previous

mChipCollection->Clear();

// collect new ones

for each (DataRow^ row in rowsChips) {

Chip^ chip = gcnew Chip(mUsb);

chip->name = row[CHIP\_NAME]->ToString();

try {

chip->id = Int32::Parse(row[CHIP\_ID]->ToString(), NumberStyles::HexNumber);

} catch (FormatException^) {}

try {

chip->address = Int32::Parse(row[CHIP\_ADDRESS]->ToString(), NumberStyles::HexNumber);

} catch (FormatException^) {}

chip->usb = row[CHIP\_USB]->ToString();

try {

chip->visible = Boolean::Parse(row[CHIP\_VISIBLE]->ToString());

} catch (FormatException^) {}

chip->config = String::Format("{0}/{1}/{2}",

Directory::GetCurrentDirectory(), PATH\_CONFIG\_CHIPS, row[CHIP\_CONFIG]);

try {

chip->autoDownload = Boolean::Parse(row[CHIP\_AUTODOWNLOAD]->ToString());

} catch (FormatException^) {}

mChipCollection->Add(chip);

}

ConsoleU::writeLine(String::Format("chips loaded. -> {0}", count), ConsoleU::Level::Normal);

return 0;

}

# 初始化时候手动选择XML文件

Void GF\_Main::toolStripMenuItem\_Load\_Click(System::Object^ sender, System::EventArgs^ e) {

Chip^ chip = mChips->current;

if (chip == nullptr) return;

openFileDialog1->InitialDirectory = String::Format("{0}/{1}",

Directory::GetCurrentDirectory(), PATH\_CONFIG\_CHIPS);

openFileDialog1->FileName = Path::GetFileName(chip->config);

if (openFileDialog1->ShowDialog() == ::DialogResult::OK) {

chip->config = openFileDialog1->FileName;

loadChipsTree();

// reload pages & registers

loadNode();

}

}

//下面是绑定树

Void GF\_Main::loadChipsTree(void) {

treeView\_Chips->Nodes->Clear();

if (mChips->count > 0) {

for each (Chip^ chip in mChips->items) {

if (!chip->visible) {

continue;

}

TreeNode^ node = gcnew TreeNode(chip->name);

for each (Page^ page in chip->pages->items) {

node->Nodes->Add(page->name);

}

treeView\_Chips->Nodes->Add(node);

}

}

// expand all tree view

treeView\_Chips->ExpandAll();

}

//下面是绑定到DataGridView中

Void GF\_Main::loadNode(void) {

if (mCurrentNode == nullptr) return;

mDataTable->Clear();

if (mCurrentNode->Level == 0) { // select chip

Chip^ chip = mChips->getByName(mCurrentNode->Text);

if (chip == nullptr) return;

// show all pages

for each (Page^ page in chip->pages->items) {

importPage(page);

}

// set current chip

mChips->current = mChips->getByName(mCurrentNode->Text);

} else if (mCurrentNode->Level == 1) { // select page

Chip^ chip = mChips->getByName(mCurrentNode->Parent->Text);

if (chip == nullptr) return;

// set current chip

mChips->current = chip;

// show page selected

Page^ page = chip->pages->getByName(mCurrentNode->Text);

if (page == nullptr) return;

// set current page

mChips->current->pages->current = page;

importPage(page);

} else {

// reserved, noting to do

}

// set chip slave address to usbIox

mUsb->slaveAddress = mChips->current->address;

}

Void GF\_Main::importPage(Page^ page) {

List<Register^>^ regs = page->registers->items;

if (regs == nullptr || regs->Count <= 0) {

ConsoleU::writeLine("page is empty, noting to load", ConsoleU::Level::Warning);

return;

}

Int32 index = 0;

for each (Register^ reg in regs) {

reg->onValueChanged += gcnew Register::valueChangedHandler(this, &GF\_Main::registerValueChanged);

try {

DataRow^ row = mDataTable->NewRow();

row[COL\_INDEX] = index++;

row[COL\_NAME] = reg->name;

if (mDataGridView->hexStyle) {

row[COL\_VALUE] = reg->value.ToString("X");

} else {

row[COL\_VALUE] = reg->value;

}

row[COL\_REMARK] = reg->remark;

row[COL\_PAGENAME] = page->name;

mDataTable->Rows->Add(row);

} catch (FormatException^) {

}

}

Table::autoResize(mDataGridView, true);

}

手动打开控制台

Void GF\_Main::debugToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

ToolStripMenuItem^ menuItem = (ToolStripMenuItem^) sender;

menuItem->Checked = !menuItem->Checked;

if (menuItem->Checked) {

::createConsole();

} else {

ConsoleU::releaseConsole();

}

}

# 不明的地方

//我感觉应该是如果打开了“控制台”返回true否则返回false，仅仅是猜测

debugToolStripMenuItem->Checked = ConsoleU::hasConsole();