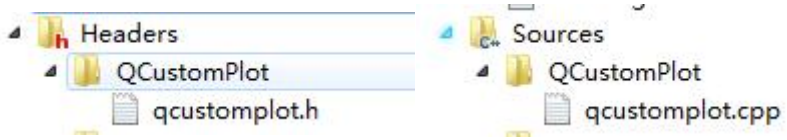
QCustomPlot 是 QT 下一个方便易用的绘图工具，只有两个文件 qcustomplot.h 和 qcustomplot.cpp 组成。源文件和使用文档可从官方网站下载。

官方网站：http://www.qcustomplot.com/

下面介绍下基本使用：

1、将 qcustomplot.cpp 和 qcustomplot.h 拷贝到工程目录下，并在工程中添加文件。



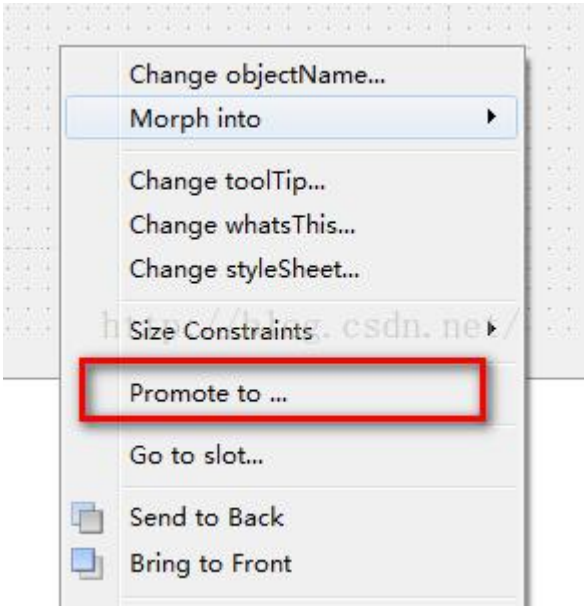
并在工程的 pro 文件添加 printsupport

[cpp] view plain copy

```
1.  greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport
```

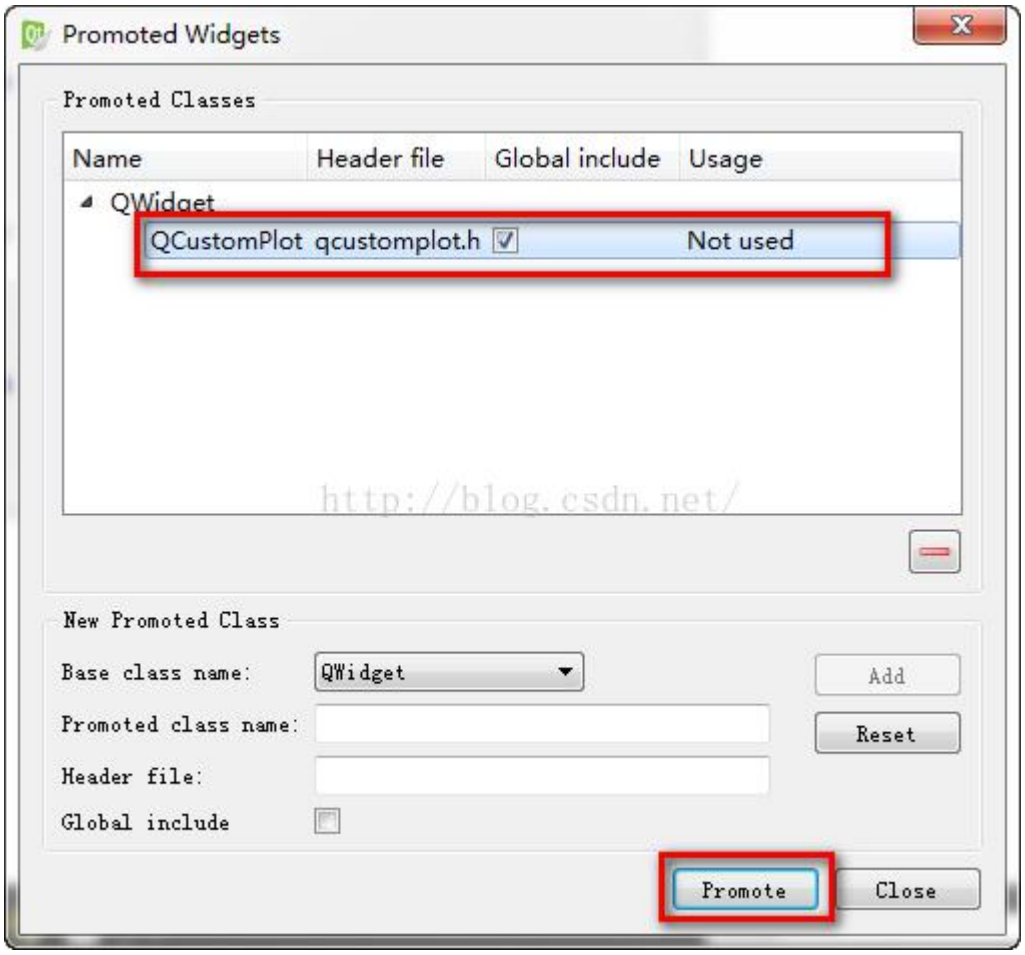
```
7 QT       += core gui
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widget
10
```

2、在 QT 里面添加一个 Widget 命名为 qcustomplotWidget，对这个 Widget 右击，点击 Promote to...



提升类名为 QCustomPlot，并点击 Add：

最后选中点击 Promote：



然后就可以在工程中通过 ui->qcustomplotWidget 直接使用了。

3、通过本人项目来展示相关代码，实际参考了官网上下载的文档和例程：

设置 x,y 轴：

[cpp] view plain copy

```
1.     ui->qcustomplot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::
   iSelectAxes |    //设置交互方式
2.                                      QCP::iSelectLegend | QCP::iSelectPlott
   ables);
3.  //    ui->qcustomplot->axisRect()->setupFullAxesBox();
4.      QBrush qBrush(QColor(255,255,255));//设置背景色
5.      ui->qcustomplot->setBackground(qBrush);
```

```
6.
7.      ui->qcustomplot->legend->setVisible(true);
8.      ui->qcustomplot->xAxis->setLabel("Time Axis (t/s)");//设置 x 轴
9.      ui->qcustomplot->xAxis->setTicks(false);
10.
11.     ui->qcustomplot->yAxis->setLabel("EEG Channel");//设置 y 轴
12.     ui->qcustomplot->yAxis->setAutoTicks(true);
13.     ui->qcustomplot->yAxis->setAutoTickStep(true);
14.     ui->qcustomplot->yAxis->setAutoSubTicks(true);
15.     ui->qcustomplot->yAxis->setRange(8000.0,10000.0);
```

x,y 轴更多设置可参考 QCPAxis Class。

添加图层：

[cpp] view plain copy

```
1.  <pre name="code" class="cpp">    graph1 = ui->qcustomplot->addGraph();//增加一条曲线图层
2.      graph2 = ui->qcustomplot->addGraph();//增加一条曲线图层
3.
4.
5.  //QCPScatterStyle QCPcs1(QCPScatterStyle::ssSquare, QColor(255,0,0),QColor(255,0,0),3);//设置折线图的点的形状及颜色
6.      QPen qPen1(QColor(255,0,0));
7.  //   graph_1->setScatterStyle(QCPcs1);
8.      graph1->setPen(qPen1);//设置画笔颜色
9.      graph1->setData(x,y);
10.     graph1->setName(QString("F3"));
11.
12.
13. //QCPScatterStyle QCPcs2(QCPScatterStyle::ssCircle, QColor(0,255,0),QColor(0,255,0),3);//设置折线图的点的形状及颜色
14.     QPen qPen2(QColor(0,255,0));
15. //   graph2->setScatterStyle(QCPcs2);
16.     graph2->setPen(qPen2);//设置画笔颜色
17.     graph2->setData(x,y);
18.     graph2->setName(QString("F4"));
```

图层更多使用情况可参考 QCustomPlot Class。

添加数据：

可用 graph1->addData()函数，具体函数参数为

[cpp] view plain copy

```cpp
void  addData (const QCPDataMap &dataMap)
void  addData (const QCPData &data)
void  addData (double key, double value)
void  addData (const QVector< double > &keys, const QVector< double > &values)
```

4、如果要实现对图的缩放移动，可以添加一下槽函数：

[cpp] view plain copy

```cpp
/**
 * @brief MainWindow::mousePress
 *  鼠标点击
 */
void MainWindow::mousePress()
{
  // if an axis is selected, only allow the direction of that axis to be dragged
  // if no axis is selected, both directions may be dragged
  if (ui->qcustomplotWidget->xAxis->selectedParts().testFlag(QCPAxis::spAxis))
  {
    ui->qcustomplotWidget->axisRect()->setRangeDrag(ui->qcustomplotWidget->xAxis->orientation());
  }
  else if (ui->qcustomplotWidget->yAxis->selectedParts().testFlag(QCPAxis::spAxis))
  {
    ui->qcustomplotWidget->axisRect()->setRangeDrag(ui->qcustomplotWidget->yAxis->orientation());
  }
  else
  {
    ui->qcustomplotWidget->axisRect()->setRangeDrag(Qt::Horizontal|Qt::Vertical);
  }
}

```

```
/**
 * @brief MainWindow::mouseWheel
 *  鼠标滚轮
 */
void MainWindow::mouseWheel()
{
  // if an axis is selected, only allow the direction of that axis to be zoomed
  // if no axis is selected, both directions may be zoomed
  if (ui->qcustomplotWidget->xAxis->selectedParts().testFlag(QCPAxis::spAxis))
  {
    ui->qcustomplotWidget->axisRect()->setRangeZoom(ui->qcustomplotWidget->xAxis->orientation());
  }
  else if (ui->qcustomplotWidget->yAxis->selectedParts().testFlag(QCPAxis::spAxis))
  {
    ui->qcustomplotWidget->axisRect()->setRangeZoom(ui->qcustomplotWidget->yAxis->orientation());
  }
  else
  {
    ui->qcustomplotWidget->axisRect()->setRangeZoom(Qt::Horizontal|Qt::Vertical);
  }
}

/**
 * @brief MainWindow::selectionChanged
 * 曲线选择
 */
void MainWindow::selectionChanged()
{
  /*
   normally, axis base line, axis tick labels and axis labels are selectable separately, but we want
   the user only to be able to select the axis as a whole, so we tie the selected states of the tick labels
   and the axis base line together. However, the axis label shall be selectable individually.

   The selection state of the left and right axes shall be synchronized as well as the state of the
```

```
57.    bottom and top axes.
58.
59.    Further, we want to synchronize the selection of the graphs with the sele
       ction state of the respective
60.    legend item belonging to that graph. So the user can select a graph by ei
       ther clicking on the graph itself
61.    or on its legend item.
62.   */
63.
64.   // make top and bottom axes be selected synchronously, and handle axis and
       tick labels as one selectable object:
65.   if (ui->qcustomplotWidget->xAxis->selectedParts().testFlag(QCPAxis::spAxis)
       || ui->qcustomplotWidget->xAxis->selectedParts().testFlag(QCPAxis::spTickLab
       els) ||
66.       ui->qcustomplotWidget->xAxis2->selectedParts().testFlag(QCPAxis::spAxi
       s) || ui->qcustomplotWidget->xAxis2->selectedParts().testFlag(QCPAxis::spTick
       Labels))
67.   {
68.     ui->qcustomplotWidget->xAxis2->setSelectedParts(QCPAxis::spAxis|QCPAxis::
       spTickLabels);
69.     ui->qcustomplotWidget->xAxis->setSelectedParts(QCPAxis::spAxis|QCPAxis::
       spTickLabels);
70.   }
71.   // make left and right axes be selected synchronously, and handle axis and
       tick labels as one selectable object:
72.   if (ui->qcustomplotWidget->yAxis->selectedParts().testFlag(QCPAxis::spAxis)
       || ui->qcustomplotWidget->yAxis->selectedParts().testFlag(QCPAxis::spTickLab
       els) ||
73.       ui->qcustomplotWidget->yAxis2->selectedParts().testFlag(QCPAxis::spAxi
       s) || ui->qcustomplotWidget->yAxis2->selectedParts().testFlag(QCPAxis::spTick
       Labels))
74.   {
75.     ui->qcustomplotWidget->yAxis2->setSelectedParts(QCPAxis::spAxis|QCPAxis::
       spTickLabels);
76.     ui->qcustomplotWidget->yAxis->setSelectedParts(QCPAxis::spAxis|QCPAxis::
       spTickLabels);
77.   }
78.
79.   // synchronize selection of graphs with selection of corresponding legend
       items:
80.   for (int i=0; i<ui->qcustomplotWidget->graphCount(); ++i)
81.   {
82.     QCPGraph *graph = ui->qcustomplotWidget->graph(i);
```

```
83.     QCPPlottableLegendItem *item = ui->qcustomplotWidget->legend->itemWithPl
    ottable(graph);
84.     if (item->selected() || graph->selected())
85.     {
86.       item->setSelected(true);
87.       graph->setSelected(true);
88.     }
89.   }
90. }
```

并连接信号：

[cpp] view plain copy

```
1.  connect(ui->qcustomplotWidget, SIGNAL(mousePress(QMouseEvent*)), this, SLOT(
    mousePress()));//连接鼠标点击信号和槽
2.  connect(ui->qcustomplotWidget, SIGNAL(mouseWheel(QWheelEvent*)), this, SLOT(
    mouseWheel()));//连接鼠标滚轮信号和槽
3.  connect(ui->qcustomplotWidget, SIGNAL(selectionChangedByUser()), this, SLOT(
    selectionChanged()));//连接曲线选择信号和槽
```

最后效果图：