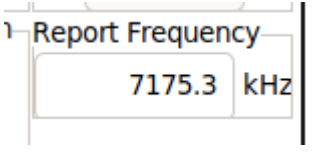
FreeDV Programmänderung

- die Frequenz u.a. wird nur nach Start angezeigt, keine Änderung wenn die Trx Freq. verstellt wird



- Hilfen
  - Debugausgaben im Quellcode

```
if (g_verbose) fprintf(stderr, "update freq and mode ....\n");
```

  - Go to References (VS Code)

```
21  }
22
23  int Hamlib::update_frequency_and_mode(void)
24  {
25      if (m_rig == nullptr)
26      {
27          return RIG_EARG; //
28      }
29
30      rmode_t mode = RIG_MODE
31      pbwidth_t passband = 0;
32      int result = rig_get_mo
```

Go to Definition F12
Go to Declaration
Go to Type Definition
Go to References Shift+F12
Peek
Find All References Alt+Shift+F12

- Quellcode-Einführung FreeDV-GUI (21.07.2022)
  - Eventhandler Start/Stop

```
1517
1518  //---------------------------------------------------------------------------
1519  // OnTogBtnOnOff()
1520  //---------------------------------------------------------------------------
1521  void MainFrame::OnTogBtnOnOff(wxCommandEvent& event)
1522  {
1523      wxString startStop = m_togBtnOnOff->GetLabel();
1524
1525      // we are attempting to start
1526
1527      if (startStop.IsSameAs("&Start"))
1528      {
1529          if (g_verbose) fprintf(stderr, "Start .....\n");
1530          g_queueResync = false;
1531          endingTx = false;
1532
1533          //
1534          // Start Running ------------------------------------------------
1535          //
```

- Quellcode-Einführung FreeDV-GUI (21.07.2022)
  - Timer
    - Definition (main.h)

```
427  #ifdef _USE_TIMER
428      wxTimer                 m_plotTimer;
429
430      // Not sure why we have the option to disable timers. TBD?
431      wxTimer                 m_pskReporterTimer;
```

    - Timerbinding + ID Setzen (main.cpp)

```
558
559  #ifdef _USE_TIMER
560      Bind(wxEVT_TIMER, &MainFrame::OnTimer, this);       // ID_MY_WINDOW);
561      m_plotTimer.SetOwner(this, ID_TIMER_WATERFALL);
562      m_pskReporterTimer.SetOwner(this, ID_TIMER_PSKREPORTER);
```

- Quellcode-Einführung FreeDV-GUI (21.07.2022)
  - Timer
    - Timerevent (main.cpp)

```
//---------------------------------------------------------------
// OnTimer()
//
// when the timer fires every DT seconds we update the GUI displays.
// the tabs only the plot that is visible actually gets updated, this
// keeps CPU load reasonable
//---------------------------------------------------------------
void MainFrame::OnTimer(wxTimerEvent &evt)
{
    if (evt.GetTimer().GetId() == ID_TIMER_PSKREPORTER)
    {
        // PSK Reporter timer fired; send in-progress packet.
        wxGetApp().m_pskReporter->send();
```

- Ergänzungen für Update Freq.
  - Definition Timer für Update Freq.
  - Neue Timer ID definieren
  - Timer ID setzen
  - Timer starten/einstellen/stoppen
  - Timer Eventhandler ergänzen

- Definition Timer für Update Freq. (main.h)

```
425
426 #ifdef _USE_TIMER
427         wxTimer                 m_plotTimer;
428
429         // Not sure why we have the option to disable time
430         wxTimer                 m_pskReporterTimer;
431 #endif
432
433     void destroy_fifos(void);
434
```

```
426
427 #ifdef _USE_TIMER
428         wxTimer                 m_plotTimer;
429
430         // Not sure why we have the option to disable time
431         wxTimer                 m_pskReporterTimer;
432+        wxTimer                 m_updFreqStatusTimer; //[U
433 #endif
434
435     void destroy_fifos(void);
436
```

- Neue Timer ID definieren (main.h)

```
102
103 enum {
104         ID_START = wxID_HIGHEST,
105         ID_TIMER_WATERFALL,
106         ID_TIMER_SPECTRUM,
107         ID_TIMER_SCATTER,
108         ID_TIMER_SCALAR,
109         ID_TIMER_PSKREPORTER,
110     };
111
```

```
102
103 enum {
104         ID_START = wxID_HIGHEST,
105         ID_TIMER_WATERFALL,
106         ID_TIMER_SPECTRUM,
107         ID_TIMER_SCATTER,
108         ID_TIMER_SCALAR,
109         ID_TIMER_PSKREPORTER,
110+        ID_TIMER_UPD_FREQ
111     };
112
```

HAM group FreeDV

- Timer ID setzen (main.cpp)

```
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Class MainFrame(wxFrame* pa->ent) : TopFrame(parent)
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
MainFrame::MainFrame(wxWindow *parent) : TopFrame(parent,
```

```

#ifdef _USE_TIMER
    Bind(wxEVT_TIMER, &MainFrame::OnTimer, this);       // I
    m_plotTimer.SetOwner(this, ID_TIMER_WATERFALL);
    m_pskReporterTimer.SetOwner(this, ID_TIMER_PSKREPORTER);
    //m_panelWaterfall->Refresh();
#endif
```

```

#ifdef _USE_TIMER
    Bind(wxEVT_TIMER, &MainFrame::OnTimer, this);       // I
    m_plotTimer.SetOwner(this, ID_TIMER_WATERFALL);
    m_pskReporterTimer.SetOwner(this, ID_TIMER_PSKREPORTER);
+   m_updFreqStatusTimer.SetOwner(this,ID_TIMER_UPD_FREQ);
    //m_panelWaterfall->Refresh();
#endif
```

- Timer starten/einstellen/stoppen (main.cpp) in Eventhandler Start/Stop

```
1731                if (m_RxRunning)
1732                {
1733            #ifdef _USE_TIMER
1734                    m_plotTimer.Start(_REFRESH_TIMER_PERIOD,
1735            #endif // _USE_TIMER
1736                }
1737            }
```

```
1741                if (m_RxRunning)
1742                {
1743            #ifdef _USE_TIMER
1744                    m_plotTimer.Start(_REFRESH_TIMER_PERIOD
1745+                   m_updFreqStatusTimer.Start(5*1000); //
1746            #endif // _USE_TIMER
1747                }
1748            }
```

```
1754            //
1755            // Stop Running ------------------------------------
1756            //
1757
1758    #ifdef _USE_TIMER
1759            m_plotTimer.Stop();
1760            m_pskReporterTimer.Stop();
1761    #endif // _USE_TIMER
```

```
1765            //
1766            // Stop Running ------------------------------------
1767            //
1768
1769    #ifdef _USE_TIMER
1770            m_plotTimer.Stop();
1771            m_pskReporterTimer.Stop();
1772+           m_updFreqStatusTimer.Stop(); // [UP]
1773+
1774    #endif // _USE_TIMER
```

- Timer Eventhandler ergänzen (main.cpp)

HAM group FreeDV

```
851 void MainFrame::OnTimer(wxTimerEvent &evt)
852 {
853     if (evt.GetTimer().GetId() == ID_TIMER_PSKREPORTER)
854     {
855         // PSK Reporter timer fired; send in-progress packet
856         wxGetApp().m_pskReporter->send();
857     }
858
859     int r,c;
```

```
852 void MainFrame::OnTimer(wxTimerEvent &evt)
853 {
854     if (evt.GetTimer().GetId() == ID_TIMER_PSKREPORTER)
855     {
856         // PSK Reporter timer fired; send in-progress packet
857         wxGetApp().m_pskReporter->send();
858     }
859
860+    if (evt.GetTimer().GetId() == ID_TIMER_UPD_FREQ)
861+    {
862+    // show freq. and mode [UP]
863+        if (wxGetApp().m_hamlib->isActive()) {
864+        if (g_verbose) fprintf(stderr, "update freq and mode
865+            wxGetApp().m_hamlib->update_frequency_and_mode()
866+        //      wxGetApp().m_hamlib->get_frequency();
867+        }
868+    }
869     int r,c;
```