MSRライブラリ利用時の注意事項について

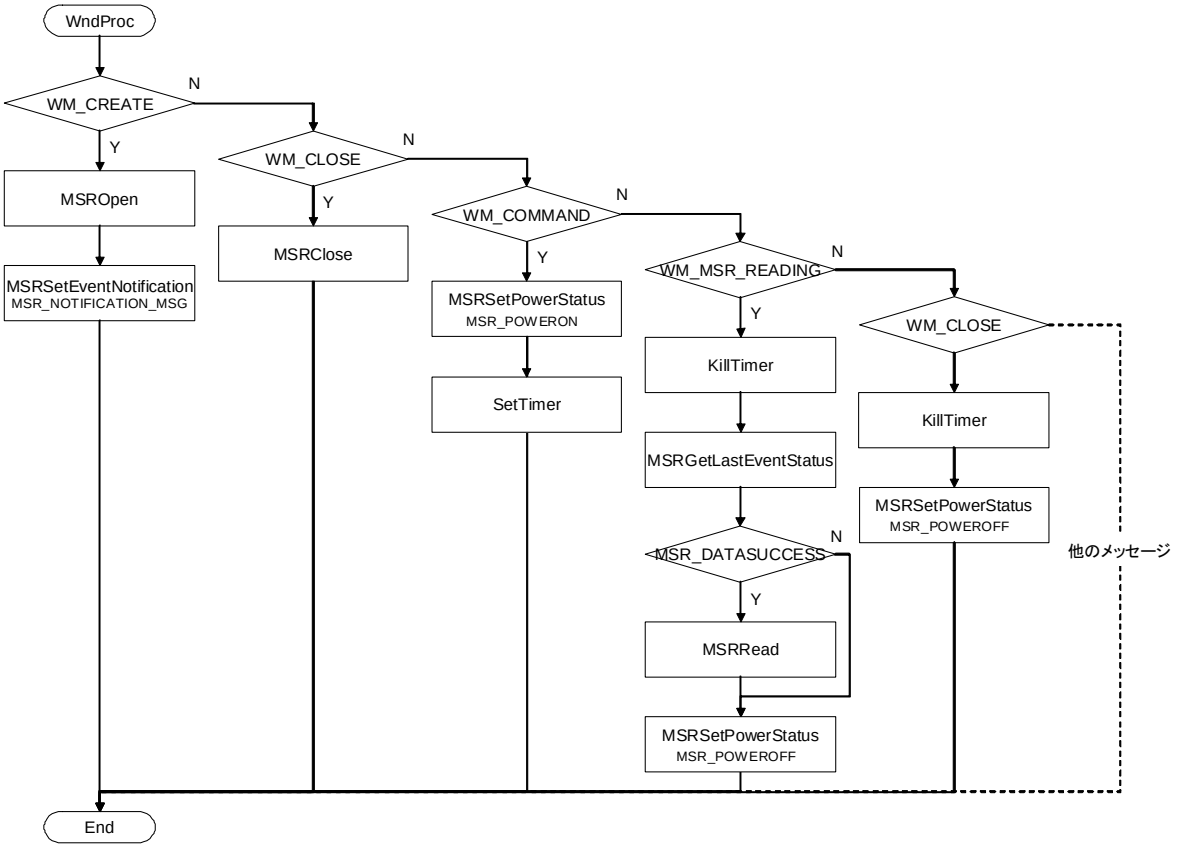
2011/01/06

DT-5300L30SW-MSR モデルにおいて、MSRGetLastEventStatus 関数で取得可能な読取完了ステータスは、MSR_DATASUCCESS、 MSR_NODATA のみとなります。（MSR_READERROR、MSR_BUFFOVER、MSR_PARITYERROR は発生しません）

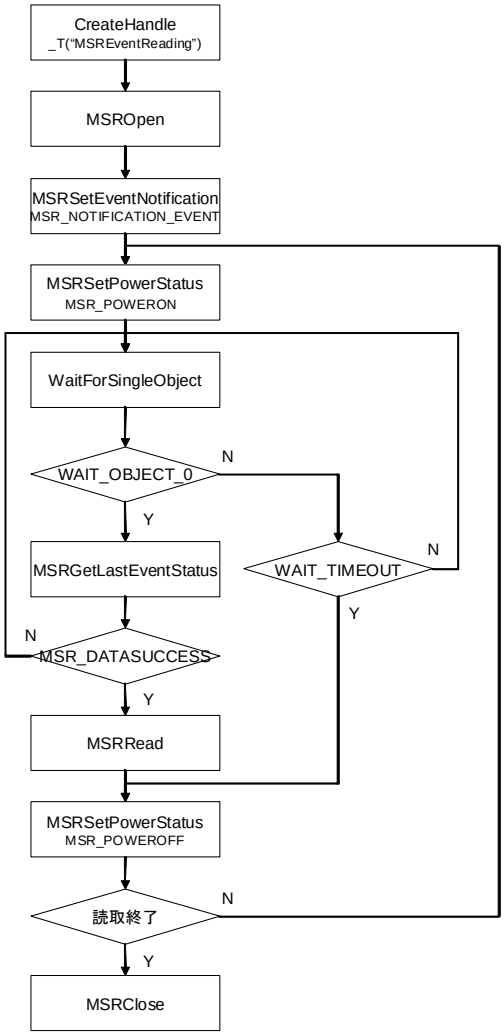
MSR ライブラリを利用してアプリケーションを開発する際は、ウィンドウメッセージ（WM_MSR_READING）またはイベント（MSREventReading）を待機するだけではなく、タイマー処理にて読み取り待ち時間を管理する必要があります。

詳細は下図処理フロー（ウィンドウメッセージ通知を使用する場合、イベント通知を使用する場合）およびサンプルプログラムを参照してください。

【処理フロー】



ウィンドウメッセージ通知を使用する場合



イベント通知を使用する場合

【サンプルプログラム C++】

```
// 読み取り開始処理
void Cxxxxx::OnClicked()
{
  DWORD dwRet;
  // MSR デバイスをオープンします
  dwRet = MSROpen( NULL );
   ・
   ・
   ・
  // MSR デバイスの電源を ON にます
  dwRet = MSRSetPowerStatus( MSR_POWERON );
   ・
   ・
   ・
  // 10 秒間の読み取り待ち時間監視タイマーを稼動します
  m_nTimerID = SetTimer( 1, 10000, NULL );
}

// 読み取りメッセージハンドラ
LRESULT Cxxxxx::OnMSR_READING(WPARAM wParam, LPARAM lParam)
{
  DWORD dwRet;
  DWORD dwGet;
  TCHAR str[30];

  // 読み取り待ち時間監視タイマーを終了します
  KillTimer( m_nTimerID );

  // 読取完了ステータスを取得します
  dwRet = MSRGetLastEventStatus( &dwGet );
  if( dwGet != MSR_DATASUCCESS )
  {
   // エラー内容を表示します
   // DT-5300L30SW-MSR モデルでは dwGet=MSR_NODATA のみとなります
   wsprintf( str, _T("ReadError¥r¥nStatus :0x%08X"), dwGet );
   MessageBox( str, 0, 0 );
  }
  else
  {
   // 読取内容を表示します
   BYTE data[256];
   DWORD len;
   dwRet = MSRRead( data, &len, NULL, NULL );
   if( dwRet == MSR_SUCCESS )
   {
      USES_CONVERSION;
      MessageBox( A2W((char*)data), 0, 0 );
   }
  }

  // MSR デバイスへの電源を OFF にします
  MSRSetPowerStatus( MSR_POWEROFF );
```

```
  return 0;
}

// 読み取り待ち時間監視タイマーハンドラ
void Cxxxxx::OnTimer(UINT_PTR nIDEvent)
{
  // 読み取り待ち時間監視タイマーを終了します
  KillTimer( m_nTimerID );

  // MSR デバイスへの電源を OFF にします
  MSRSetPowerStatus( MSR_POWEROFF );

  CDialog::OnTimer(nIDEvent);
}
```