Ergänzung zum Beitrag in FA 6/23, S. 450 ff. „Web-SDR-Empfang auf Langwelle mit Datendecodierung“

Ergänzend zum Beitrag präsentieren wir hier noch einige Beispiele und Anleitungen, die in der gedruckten Ausgabe keinen Platz mehr fanden. Dazu zählen die Decodierung des Wetternachrichtendiensts DDH47, das Zeitzeichen ALS162 und das auf das Sendesignal des Rundfunksenders BBC Four modulierte Zeitzeichen. Die zugehörigen Processing-Dateien stehen ebenfalls im Download-Bereich auf *www.funkamateur.de* zum Herunterladen bereit.

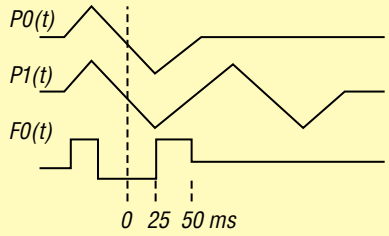

Bild 13: Signalformen der Phasen- und Frequenzmodulation

■ Wetternachrichten von DDH47

Auf 147,3 kHz verbreitet der Sender DDH47 in F1B (RTTY) mit 50 Baud und einem Hub von ±42,5 Hz Wetternachrichten. Die Zeichen werden im Baudot-Code ausgegeben. Bild 14 zeigt die empfangene Phase (untere Spur) und das RTTY-Empfangssignal (obere Spur).
Zum Empfang ist grundsätzlich der gleiche Empfänger wie zum Empfang des Rundsteuerfunks EFR einsetzbar, nur eben für den 5-Bit-Baudot-Code ohne Paritätsbit. Der Empfänger gibt dann beispielsweise den in Ergebnis 6 gezeigten Text aus.

■ TDF (France Inter, ALS162) auf 162 kHz

Auf 162 kHz gibt Télé Diffusion de France über dem Sender ALS162 u.a. phasenmoduliert die Zeitinformation aus. Die Zeitinformation wird durch entsprechende Symbole zum Sekundenbereich übertragen. Die beiden Signalformen für Null und Eins sind in Bild 13 als $P_0(t)$ und $P_1(t)$ dargestellt. Die Signalform $P_1(t)$ entspricht dem zweimaligem Senden von $P_0(t)$ im Abstand von 100 ms.
Da die Phasenverläufe abschnittsweise linear sind kann man die Modulation auch als Frequenzmodulation betrachten. Für das Signall NULL ist der entsprechende Frequenzverlauf in Bild 13 als $F_0(t)$ dargestellt.
In Bild 15 sieht man die Inphasen/Quadraturphasensignale $I(t)$ und $Q(t)$ für den TDF. Der sin/cos-Verlauf entsteht durch den Frequenzoffset des LO. Im kleinen sieht man die schwankende Phase. Nach

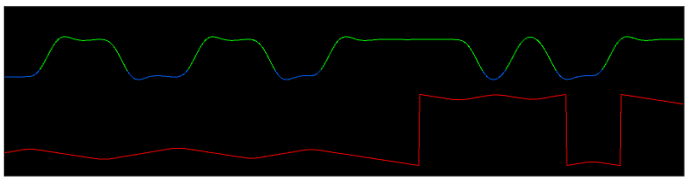

Bild 14: Empfangssignale des Senders DDH47

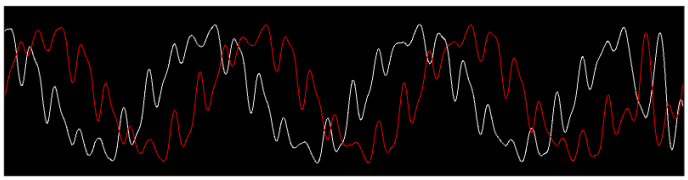

Bild 15: I/Q-Signale des Senders der TDF

Ergebnis 6: Ausgabe des DDH47-Programms

```
SUEDWEST 4, ABNEHMEND 3.
RIGAISCHER MEERBUSEN :
SUEDWEST UM 4.
WINDVORHERSAGE FUER 10M HOHE, WINDSTAERKE IN BEAUFORT,
LUFTDRUCK IN HEKTOPASCAL
=
NNNN
CQ CQ CQ DE DDH47 DDH9 DDH8
FREQUENCIES 147.3 KHZ 11039 KHZ 14467.3 KHZ
RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRY
```

Listung 12: Dekodierung der Frequenz mit Filterung

```
void FMdemod(float II, float QQ) {
  float phi = -sigSign * atan2(II, QQ) ;
  Phase = (byte)( (phi + PI) / PI * 128) ;
  PhaseDifference = byte(Phase - LastPhase) ;
  LastPhase = Phase ;
  FrequencyEstimate = PhaseDifference ;                          // cic lowpass
  FRQintegrator += FrequencyEstimate ;
  FRQcicout = FRQintegrator - FRQfifo[FRQfifoPTR] ;
  FRQfifo[FRQfifoPTR] = FRQintegrator ;                  // advance pointer and bump around
  FRQfifoPTR++ ; if ( FRQfifoPTR == FRQlength ) { FRQfifoPTR = 0 ; }
  Frequency=FRQcicout ;
  x2 = x1 ; x1 = x0 ; x0 = MatchedFilter(-Frequency) ;
```

Listing 13: Programmcode des Matched Filters für den TDF Empfang

```
void advance(int d) {
  ptr += d ;
  if ( ptr >= TDFlength ) { ptr -= TDFlength ; } }
int MatchedFilter(int Frequency) {
  int sum ;                                          // CIC implementation of matched filter
  ptr = TDFptr ;                                  // ptr points to oldest sample in TDFbuffer
  sum = -TDFbuffer[ptr] ;
  advance(12) ;
  sum += 2*TDFbuffer[ptr] ;
  advance(25) ;
  sum -= 2*TDFbuffer[ptr] ;
  advance(12) ;
  sum += TDFbuffer[ptr] ;
  TDFintegrator += Frequency ;                          // replace oldest sample in buffer
  TDFbuffer[TDFptr] = TDFintegrator ;
  TDFptr++ ;
  if ( TDFptr == TDFlength) {
    TDFptr = 0 ; }
  return sum ; }
```

Listing 14: Sampling der TDF-Bits

```
void exeSecondTimer() {
  SecondTimer += (1.00 - 0.0002) ;
  if (SecondTimer >= 500.0) {
    SecondTimer -= 500.0 ; }
  if (((int)SecondTimer) == 50) {
    if ( x1 > THRESHOLD1 ) { TheBit = 1 ; }
    else { TheBit = 0 ; }
    print(TheBit) ;
    if(TheSecond > 61) {
      print(" gap error!") ;
      TheSecond = 0 ; }
    if (TheSecond < 60) {
      TheBits[TheSecond] = TheBit ; }
    TheSecond++ ; } }
```

Listing 15: Bit-Decodierung

```
int getBCD(int BitPos , int BitCount) {
  int value ;
  int k, q ;
  value = 0 ;
  q = 1 ;
  for (k = 0 ; k < BitCount ; k++ ) {
    if ( TheBits[BitPos++] > 0 ) { value += q ; ParityBit ^= 1 ; }
    q=2 * q ; }
  return value ; }
  ParityBit = 0 ;                                  // Anzeige der Minuten
  min1 = getBCD(21, 4) ;
  min10 = getBCD(25, 3) ;
  P1 = ParityBit ^ TheBits[28] ;
  ParityBit = 0 ;                                  // Anzeige der  Stunden
  hrs1 = getBCD(29, 4) ;
  hrs10 = getBCD(33, 2) ;
  P2 = ParityBit ^ TheBits[35] ;
```

Ergebnis 7: Ausgabe der Zeitinformation des Senders der TDF

```
00011 10000 00010 00100 11110 10000 10010 00010 00011 01100 01000 10010 peak found!
#01000100: 12:17 04.06 SATURDAY 2022 p000
00001 10000 00010 00100 10001 10000 10010 00010 00011 01100 01000 10010 peak found!
#01000100: 12:18 04.06 SATURDAY 2022 p000
00011 10000 00010 00100 11001 10010 10010 00010 00011 01100 01000 10010 peak found!
#01000100: 12:19 04.06 SATURDAY 2022 p000
```

den NULL/EINS Impulsen werden proprietäre Informationen übertragen. In der 59. Sekunde findet keine Phasenmodulation statt. Diese Lücke benutzt man zur Synchronisation. Die Routine zur Frequenz-Demodulation sieht man in Listing 12. Die ermittelte Frequenz (*Variable Frequency Estimate*) wird in einem CIC-Filter – einem Tiefpassfilter – gefiltert. Das gefilterte Signal durchläuft anschliessend ein sogenanntes *Matched Filter* zur Detektion von $F_0(t)$. Das *Matched Filter* hat als Stoßantwort die zu detektierende Signalform $F_0(t)$ selbst.

Das *Matched Filter* ist programmiert wie in Listing 13 gezeigt. Das *Matched Filter* wird mit 500 Samples/s getaktet (Sampling-Periode = 2ms). Wenn $P_0(t)$ gesendet wurde, erzeugt das *Matched Filter* einen einzelnen Peak bei $t = 0$. Wird $P_1(t)$ empfangen erzeugt das *Matched Filter* einen Peak bei $t = 0$ und einen weiteren bei $t = 100$ ms, d. h. beim Sample 50. Am Vorhandensein eines Peaks bei $t = 100$ms erkennt man also ob eine Null oder eine Eins gesendet wurde. Die Samples innerhalb einer Sekunde zählt die Variable *SecondTimer*. Das Samplen der empfangenen Bits erfolgt dann wie in Listing 14 gezeigt. Die empfangenen Bits werden in der Variablen *TheBits* gespeichert.

In Bild 16 sieht man entsprechende Signale. Die oberste Spur zeigt das Ausgangssignal des *Matched Filter* in der Nähe des Sekundenbeginns. Die beiden senkrechten Striche markieren die Abtastzeitpunkte $t = 0$ ms und $t = 100$ ms. Die zweite Spur zeigt das Ausgangssignal des Matched Filter über einen längeren Zeitraum. Gut erkenbar ist das Aussetzen der Phasenmodulation in Sekunde 59. Die dritte Spur zeigt das Lückensignal Silencetimer das immer bei einem Puls des Mached Filters zurückgesetzt wird.

In den Pulslücken wird es inkrementiert und mit dem Überschreiten einer Schwelle von 400 haben wir die Lücke zur 59. Sekunde erkannt.

Um die Zeitinformation darzustellen verwendet man das Listing 15. Die Routine *getBCD()* holt Bits aus dem Array *TheBits* und verrechnet sie gemäß dem BCD-Code zu einer Zahl die zurückgegeben wird. Gleichzeitig wird das Paritätsbit in *ParityBit* aktualisiert. In den nachfolgenden vier Zeilen sieht man die Anzeige der Minuten, in den letzten vier Zeilen erfolgt die Anzeige der Stunden. Die weiteren Informationen werden entsprechend angezeigt. Die Ausgabe sieht dann aus wie in Ergebnis 7 dargestellt.

BBC 198 kHz

Auf 198 kHz betreibt die BBC einen AM Rundfunksender. Zusätzlich zur Amplitudenmodulation wird eine Phasenmodulation (PM) verwendet um zusätzliche digitale Daten auszusenden. Die PM beträgt ±22,5° und die Datenrate ist 25Bit/s. In Bild 17 sieht man an der oberen Spur (blau), wie das Inphasensignal im Empfänger aussieht.

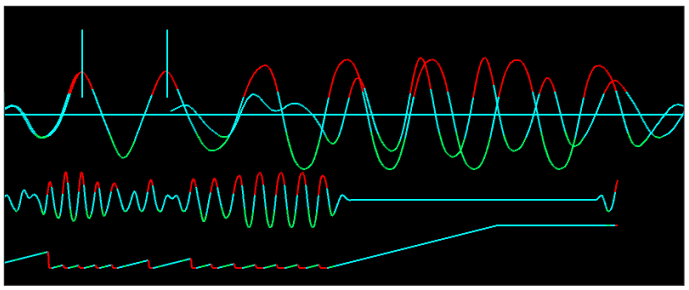

Bild 16: Empfangssignale des Senders der TDF

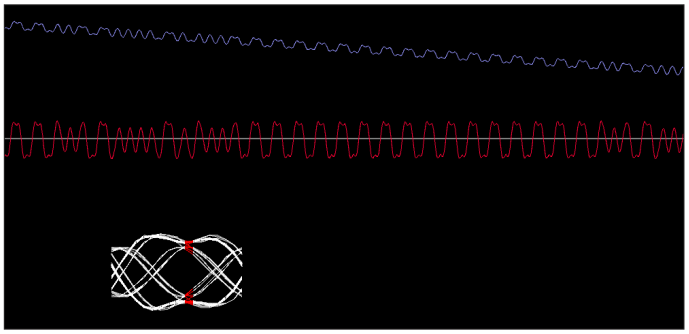

Bild 17: BBC-Empfangssignale

Zur Decodierung verwenden wir nicht die Phase sondern die Augenblicksfrequenz, welche in der zweiten Spur in Bild 17 dargestellt ist (rote Spur) da dann der LO-Offset keine Rolle spielt. Da das Empfangssignal keine spektrale Komponente mit der Frequenz des Bittaktes enthält, können wir keine gewöhnliche PLL zur Bittaktgewinnung einsetzen. Statt dessen kommt eine sogenannte Costas-Loop zum Einsatz. Das Programm der *Costas-Loop* ist in Listing 16 dargestellt. In Bild 17 ist unten (weiß) ein Augendiagramm für den empfangenen Bitstrom dargestellt. Man erkennt die gute Öffnung des Auges welches guten Empfang gewährleistet.

Nach der Bittaktrückgewinnung muss man auch die Blockgrenzen im Bitstrom erkennen. Dies gelingt aufgrund des verwendeten fehlerkorrigierenden Codes. Immer wenn ein fehlerfreies Wort im Bitstrom entdeckt wird, wird die *SyncMessage*-Routine aufgerufen (Listing 17). Diese
detektiert mehrere gleiche Syncmessages und führt die Blocksynchronisation dann entsprechend durch. Immer wenn eine komplette Gruppe fehlerfrei empfangen wurde (Listing 18) wird diese angezeigt. Handelt es sich um Gruppentyp 0x0f wird eine Uhrzeitgruppe decodiert. Die Ausgabe des Programms sieht dann aus wie in Ergebnis 8 dargestellt.

Damit sind wir am Ende unserer Tour über den Langwellenbereich angekommen. Mit dem Web-SDR Twente kann man gut den Empfang dieser Sender durchführen. Mit unseren Programmen kann man dann die enthaltene Information decodieren.

Listing 16: Programmquelltext der Costas-Loop

```
void ClockPLL16(int Signal) {
  if (Signal < 0) { AbsSignal = -Signal ; }
  else { AbsSignal = Signal ; }
  Isum += AbsSignal * cos16[Clk16] ;
  Qsum += AbsSignal * sin16[Clk16] ;
  Count++ ;
  Count &= 63 ;
  if (Count == 0) {
    int ClockPhase = iCordic(Isum / 16 , Qsum / 16 ) ;
    if (ClockPhase > 118 + 15) { ClockCorrection = -1 ; }
    if (ClockPhase < 118 - 15) { ClockCorrection = 1 ; }
    Isum = 0 ;
    Qsum = 0 ; }  }
  if ( ( Clk16 == 14 ) & (ClockCorrection != 0 ) ) {
    Clk16 += ClockCorrection ;
    ClockCorrection = 0 ; }
  if (Clk16 == SAMPLECLK2) {
    if (Signal < 0) { doBBCbit(1) ; }
    else { doBBCbit(0) ; }  }
  Clk16++ ;
  if (Clk16 == 16) { Clk16 = 0 ; }
```

Listing 17: Blocksynchronisation

```
if ( SRsyn == 0x1283 ) { SyncMessage(0) ; }
void SyncMessage(int sm) {
  sm = local_time - sm ;
  if (sm < 0) { sm += 50 ; }
  outBlank() ; outPutc('S') ; DecOut2(sm) ;
  outBlank() ;
  sync_fifo0 = sync_fifo1 ;
  sync_fifo1 = sync_fifo2 ;
  sync_fifo2 = sm ;
  if ( (sync_fifo0 == sync_fifo1) && (sync_fifo1 == sync_fifo2) ) {
    sync_point = sync_fifo0 ; }  }
```

Listing 18: Zeitdecodierung

```
if ( ( synced_time == 0) && (SRsyn == 0x1283) ) {
  outCrlf() ;
  outPutc('+') ;
  DisplayGroup() ;
  if ( ( (SRww1 & 0x8000) == 0) && ( (SRgt&0xf) == 0 )) {
    DisplayTimeGroup() ;
    seconds = 0 ; }  }
```

Ergebnis 8: Ausgabe der Zeitinformation der BBC

```
+0 AAAA AAAA 117D 1000001011001011010001100000110000010111011110011 1 S22
+0 5968 C182 1DE7 time= 12:06:60 week=22 day=04 thursday
1000010101010101010101010101010101010101000101111101 S22
+0 AAAA AAAA 117D 1010111100001010001011000101010100010110100001 1000 S22
+5 E145 8AA2 1A18 1111010011011111111000000100110001001001110010 1110 S22
+E 9BFC 0989 072E 1111010100000110000100000000010000011100001111 1001 S22
+E A0C2 0083 10F9 1111010000000111111000100111000000111010100101 0100 S22
+E 80FC 4E07 0A54 1000010101010101010101010101010101010101000101111101 S22
+0 AAAA AAAA 117D 1000010101010101010101010101010101010101000101111101 S22
+0 AAAA AAAA 117D 1000010101010101010101010101010101010101000101111101 S22
+0 AAAA AAAA 117D 1000010101010101010101010101010101010101000101111101 S22
+0 AAAA AAAA 117D 1000010101010101010101010101010101010101000101111101 S22
```

Tabelle 3: Processing-Dateien aller Beipiele dieses Beitrags

scopeShow1V01.pde	
DCFrx2V01.pde	DCF77-Amplitudenmodulation
DCFprn1V01.pde	DCF77-Phasenmodulation
MSFrx2V01.pde	Empfänger für Zeitzeichen MSF auf 60 kHz
specAna1V01.pde	
EFR129p1rx1V01.pde	EFR-Empfänger für DCF49 auf 129 kHz
EFR135p6rx1V01.pde	EFR-Empfänger für HGA22 auf 135,6 kHz
EFR139rx1V01.pde	EFR-Empfänger für DCF39 auf 139 kHz
DDH147p3rx1V01.pde	RTTY-Wetternachrichten-Empfänger für DDH47 auf 147,3 kHz
TDFrx1V01.pde	Empfänger für Zeitzeichen ALS162 auf 162 kHz, TDF
BBCrx1V01.pde	BBC Four