

# Projekt-Archiv: Yaesu CAT-Synchronisation (FT-1000MP & FTdx-5000)

**Ziel des Projekts:** Eine synchrone, bidirektionale Steuerung von Frequenz und Betriebsart (inklusive digitaler Sub-Modi) zwischen einem modernen Yaesu FTdx-5000 und einem klassischen FT-1000MP via Liberty Basic.

---

## Die 5 goldenen Regeln der alten Yaesu-CAT-Steuerung (FT-1000MP Ära)

Wer alte Yaesu-Transceiver per Software steuern will, verzweifelt oft an der lückenhaften Dokumentation. Diese fünf Erkenntnisse aus der Praxis sparen wochenlange Frustration:

1. **Traue niemals blind dem Handbuch:** Yaesu hat in den Dokumentationen oft die Parameter-Bytes vertauscht. Wenn das Handbuch sagt, der Modus-Code muss ins erste Byte (P1), will das Funkgerät ihn in der Realität oft im vierten Byte (P4) vor dem Opcode haben.
  2. **Lesen  $\neq$  Schreiben:** Die Hex-Codes, die das Funkgerät beim *Auslesen* sendet, sind oft komplett andere als die Codes, die es zum *Setzen* einer Betriebsart erwartet. (Beispiel: Code 4 ist beim Lesen FM, führt beim Senden aber zu AM!). Man muss die "Setz-Tabelle" selbst austesten.
  3. **Das Seitenband-Rätsel (Sub-Modi):** Der Haupt-Code (z.B. in Byte 8) verrät nur die grundlegende Betriebsart (CW, RTTY, PKT). Ob das Gerät im USB- oder LSB-Seitenband läuft, versteckt sich oft in einem völlig anderen Byte (beim FT-1000MP ist es Byte 9).
  4. **Alte Hardware braucht Atempausen:** Werden Frequenz und Betriebsart im Millisekunden-Takt gleichzeitig gesendet, verschluckt sich der alte Prozessor. Man muss dem Funkgerät kleine Pausen (z. B. 150 ms) gönnen, damit die Relais schalten können und die PLL einrastet, bevor der nächste Befehl kommt.
  5. **Nicht alles ist per CAT steuerbar:** Manche Unterscheidungen, wie etwa zwischen PKT-LSB und PKT-USB, existieren im CAT-Protokoll alter Geräte schlichtweg nicht. Hier hilft nur: Den Standard-PKT-Befehl senden und das gewünschte Seitenband im internen Menü (beim FT-1000MP: Menü 8-6) festlegen.
- 

## Die entschlüsselten Mode-Tabellen (FT-1000MP)

**Auslesen des FT-1000MP (Die Wahrheit aus Byte 8 und 9):**

- **CW-LSB:** B8 = 2 | B9 = 196
- **CW-USB (Reverse):** B8 = 2 | B9 = 68
- **RTTY-LSB:** B8 = 5 | B9 = 17
- **RTTY-USB:** B8 = 5 | B9 = 145
- **PKT-LSB:** B8 = 6 | B9 = 17

- **PKT-FM:** B8 = 6 | B9 = 145

### Senden an den FT-1000MP (Die funktionierenden P4-Codes):

- 0 = LSB
- 1 = USB
- 2 = CW
- 3 = CW Reverse
- 4 = AM
- 6 = FM
- 8 = RTTY-LSB
- 9 = RTTY-USB
- 10 = PKT (Standard/LSB)
- 11 = PKT-FM

## Das finale Liberty Basic Skript

Dieses Skript setzt alle obigen Regeln um, wertet die Sub-Modi aus und synchronisiert beide Funkgeräte fehlerfrei.

Basic

```
' --- CAT SYNC MANAGER: FT-1000MP <-> FTdx-5000 V1.0 ---
nomainwin

' =====
' SICHERHEITSABFRAGE (START-FENSTER)
' =====
WindowWidth = 350 : WindowHeight = 300
UpperLeftX = (DisplayWidth-WindowWidth)/2
UpperLeftY = (DisplayHeight-WindowHeight)/2

' Warntexte
statictext #warn.t1, "!!! A C H T U N G !!!", 100, 20, 200, 20
statictext #warn.t2, "Bitte beide Transceiver auf", 100, 60, 200, 20
statictext #warn.t3, "kleinste Leistung einstellen", 100, 80, 200, 20
statictext #warn.t4, "und nicht senden!!!", 100, 100, 200, 20

' Formatierung für die Anführungszeichen in Liberty Basic: ""
statictext #warn.t5, "FTdx-5000 -> BK ""AUS"", PWR ""10W""", 50, 140, 250, 20
statictext #warn.t6, "FT-1000MP -> VOX ""AUS"", PWR ""0W""", 50, 160, 250, 20

statictext #warn.t7, "alles gemacht?", 125, 200, 100, 20

button #warn.btn, "weiter", [startMain], UL, 115, 225, 100, 30
```

```

open "Sicherheitswarnung" for window as #warn
print #warn.t1, "!font Arial 12 bold"
print #warn.t5, "!font Arial 10 bold"
print #warn.t6, "!font Arial 10 bold"
print #warn, "trapclose [quitProg]"

```

```

wait ' <<< Hier stoppt das Programm und wartet auf Klick oder Schließen

```

```

[quitProg]
close #warn : end

```

```

[startMain]
close #warn ' Schließe das Warnfenster

```

```

' GUI-Setup
WindowWidth = 450 : WindowHeight = 320
UpperLeftX = (DisplayWidth-WindowWidth)/2
UpperLeftY = (DisplayHeight-WindowHeight)/2

```

```

button #main.btn1, "MASTER: FT-1000MP -> 5000", [setMPto5k], UL, 50, 20, 330, 40
button #main.btn2, "MASTER: FTdx-5000 -> MP", [set5ktoMP], UL, 50, 70, 330, 40
button #main.btn3, "Synchronisation AUS", [setOff], UL, 50, 130, 330, 40

```

```

statictext #main.status, "Status: Synchronisation AUS", 50, 200, 300, 20
statictext #main.debug, "Warte auf Daten...", 50, 230, 380, 20

```

```

open "sync FTdx5000 <-> FT1000MP V1.0 / DD2NU" for window as #main
print #main, "trapclose [beenden]"

```

```

' --- INI DATEI LESEN ---
iniFile$ = DefaultDir$ + "\ini.txt"

```

```

if fileExists(DefaultDir$, "ini.txt") then
    open iniFile$ for input as #ini
    line input #ini, comPortFT5k$
    line input #ini, baudFT5k$
    line input #ini, comPortFT1k$
    line input #ini, baudFT1k$
    close #ini
    comFT5k$ = comPortFT5k$ + ":" + baudFT5k$ + ",n,8,2"
    comFT1k$ = comPortFT1k$ + ":" + baudFT1k$ + ",n,8,2"
else
    comFT5k$ = "COM17:38400,n,8,2"
    comFT1k$ = "COM22:4800,n,8,2"
end if

```

```

' Ports oeffnen
on error goto [errorPort]

```

```
open comFT5k$ for random as #dx5k
open comFT1k$ for random as #mp1k
```

```
on error goto 0
```

```
' FTdx-5000 "Auto Info" ON
print #dx5k, "AI1;"
```

```
' <<< NEU: lastSubModeMP hinzugefügt >>>
global syncMode, lastFreq, lastModeMP, lastSubModeMP, lastMode5k$
syncMode = 0
lastFreq = 0
lastModeMP = -1
lastSubModeMP = -1
lastMode5k$ = ""
```

```
' --- Start der Hauptschleife ---
```

```
[mainLoop]
```

```
scan
```

```
' =====
```

```
' MODUS 1: FT-1000MP -> FTdx-5000
```

```
' =====
```

```
if syncMode = 1 then
```

```
if lof(#mp1k) > 0 then dummy$ = input$(#mp1k, lof(#mp1k))
```

```
print #mp1k, chr$(0); chr$(0); chr$(0); chr$(3); chr$(16);
```

```
t = 0
```

```
while lof(#mp1k) < 32 and t < 50
```

```
scan
```

```
timer 10, [tick] : wait : [tick] : timer 0
```

```
t = t + 1
```

```
wend
```

```
if lof(#mp1k) >= 32 then
```

```
vfoData$ = input$(#mp1k, 32)
```

```
b2 = asc(mid$(vfoData$, 2, 1))
```

```
b3 = asc(mid$(vfoData$, 3, 1))
```

```
b4 = asc(mid$(vfoData$, 4, 1))
```

```
b5 = asc(mid$(vfoData$, 5, 1))
```

```
rawVal = (b2 * 16777216) + (b3 * 65536) + (b4 * 256) + b5
```

```
currentFreqHz = int(rawVal * 0.625) - 5
```

```
if currentFreqHz <> lastFreq then
```

```
call SyncToDX5k currentFreqHz
```

```
print #main.debug, "Frequenz: "; currentFreqHz; " Hz"
```

```
lastFreq = currentFreqHz
```

```
end if
```

```

' <<< MODE SYNC MP -> 5000 (MIT BYTE 8 UND 9) >>>
m8 = asc(mid$(vfoData$, 8, 1))
m9 = asc(mid$(vfoData$, 9, 1))

if m8 <> lastModeMP or m9 <> lastSubModeMP then
  md5k$ = ""
  select case m8
    case 0 : md5k$ = "01" ' LSB
    case 1 : md5k$ = "02" ' USB
    case 3 : md5k$ = "05" ' AM
    case 4 : md5k$ = "04" ' FM

    case 2 ' CW
      if m9 = 196 then md5k$ = "07" ' CW-LSB
      if m9 = 68 then md5k$ = "03" ' CW-USB (Reverse)
      if md5k$ = "" then md5k$ = "07" ' Fallback

    case 5 ' RTTY
      if m9 = 17 then md5k$ = "06" ' RTTY-LSB
      if m9 = 145 then md5k$ = "09" ' RTTY-USB
      if md5k$ = "" then md5k$ = "06" ' Fallback

    case 6 ' PKT
      if m9 = 17 then md5k$ = "08" ' PKT-LSB
      if m9 = 145 then md5k$ = "0A" ' PKT-FM
      if md5k$ = "" then md5k$ = "08" ' Fallback
  end select

  if md5k$ <> "" then
    print #dx5k, "MD"; md5k$; ";";
  end if
  lastModeMP = m8
  lastSubModeMP = m9
end if
end if
end if

' =====
' MODUS 2: FTdx-5000 -> FT-1000MP
' =====

if syncMode = 2 then
  nDX = lof(#dx5k)
  if nDX > 0 then
    resDX$ = input$(#dx5k, nDX)

    ' ZUERST DEN MODE SENDEN
    pMD = instr(resDX$, "MD")
    if pMD > 0 then
      mdStr$ = mid$(resDX$, pMD + 2, 2)

      if mdStr$ <> lastMode5k$ and mdStr$ <> "" then
        mdMP = -1
      end if
    end if
  end if
end if

```

```

select case mdStr$
  case "01" : mdMP = 0 ' LSB
  case "02" : mdMP = 1 ' USB
  case "03" : mdMP = 2 ' CW
  case "07" : mdMP = 3 ' CW Reverse
  case "05" : mdMP = 4 ' AM
  case "04" : mdMP = 6 ' FM
  case "06" : mdMP = 8 ' RTTY-LSB
  case "09" : mdMP = 9 ' RTTY-USB
  case "08" : mdMP = 10 ' DATA-LSB -> PKT
  case "0A" : mdMP = 11 ' DATA-FM -> PKT-FM
  case "0C" : mdMP = 10 ' DATA-USB -> MP kann nur normales PKT (10)
end select

if mdMP >= 0 then
  print #mp1k, chr$(0); chr$(0); chr$(0); chr$(mdMP); chr$(12);
  print #main.debug, "Mode gesendet (Code: "; mdMP; ")"

  timer 150, [waitMP] : wait : [waitMP] : timer 0
end if
lastMode5k$ = mdStr$
end if
end if

' DANACH ERST DIE FREQUENZ SENDEN
posFA = 0
for i = 1 to len(resDX$) - 2
  if mid$(resDX$, i, 2) = "FA" then posFA = i
next i

if posFA > 0 then
  raw$ = ""
  for i = posFA + 2 to len(resDX$)
    c$ = mid$(resDX$, i, 1)
    if instr("0123456789", c$) > 0 then raw$ = raw$ + c$
    if c$ = ";" then exit for
  next i
  newFreq = val(raw$)

  if newFreq > 100000 and newFreq <> lastFreq then
    call SyncToMP newFreq
    print #main.debug, "Frequenz: "; newFreq; " Hz"
    lastFreq = newFreq
  end if
end if

end if
end if

timer 20, [p] : wait : [p] : timer 0
goto [mainLoop]

```

```

' --- BUTTON EVENTS ---
[setMPto5k]
    syncMode = 1 : lastFreq = 0 : lastModeMP = -1 : lastSubModeMP = -1
    print #main.status, "Status: MP -> 5000 (Aktiv)" : goto [mainLoop]

[set5ktoMP]
    syncMode = 2 : lastFreq = 0 : lastMode5k$ = ""
    print #main.status, "Status: 5000 -> MP (Aktiv)" : goto [mainLoop]

[setOff]
    syncMode = 0 : print #main.status, "Status: Synchronisation AUS" : goto [mainLoop]

[beenden]
    close #dx5k : close #mp1k : close #main : end

[errorPort]
    notice "Port-Fehler! Pruefe COMx Ports." : end

' =====
' SUBROUTINEN
' =====

sub SyncToMP freq
    fHz = int(freq / 10)
    s$ = right$("00000000" + str$(fHz), 8)

    b4 = decToHex(val(mid$(s$, 1, 2)))
    b3 = decToHex(val(mid$(s$, 3, 2)))
    b2 = decToHex(val(mid$(s$, 5, 2)))
    b1 = decToHex(val(mid$(s$, 7, 2)))

    print #mp1k, chr$(b1); chr$(b2); chr$(b3); chr$(b4); chr$(10);
end sub

sub SyncToDX5k freq
    f$ = str$(freq)
    while len(f$) < 8 : f$ = "0" + f$ : wend
    if len(f$) > 8 then f$ = right$(f$, 8)

    cmd$ = "FA" + f$ + ";"
    print #dx5k, cmd$;
end sub

function decToHex(d)
    decToHex = (int(d / 10) * 16) + (d mod 10)
end function

function fileExists(path$, filename$)
    files path$, filename$, info$()
    fileExists = val(info$(0, 0)) > 0
end function

```