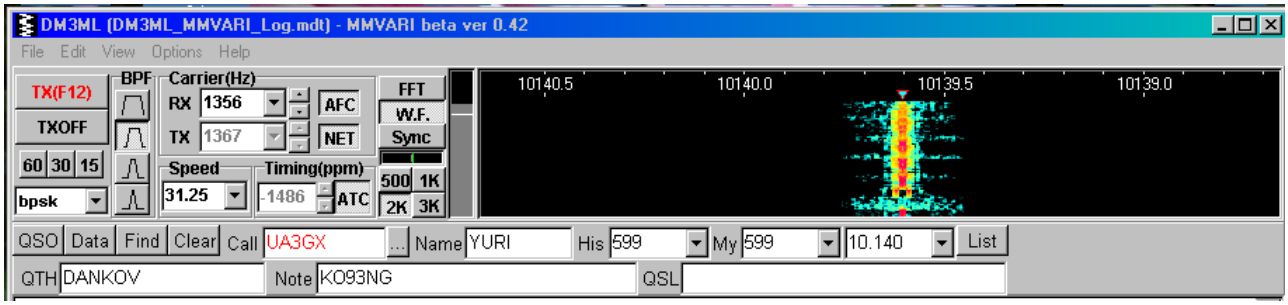
MMVARI-Handbuch

Version Beta 0.42



Zusammengestellt aus den Texten

EPROJECT.txt
EHISTORY.txt
EMMVARI.txt

Übersetzt von Eike, DM3ML, im Oktober 2009

Inhaltsverzeichnis

Vorwort des Übersetzers

Im Oktober 2009 habe ich mir die drei zu MMVARI vorliegenden englischen Texte zur Übersetzung vorgenommen. Es handelt sich um das Projekt (EPROJECT.txt), die Entwicklungsgeschichte (EHISTORY.txt) und die eigentliche Programmbeschreibung (EMMVARI.txt). JE3HHT beschreibt darin vor allem programmiertechnische Probleme. Interessant ist die gegenüber anderen Programmen sehr weit ausgebaute Macro-Programmierung.Sie dürfte aber mehr gestandene Programmierer als den einfachen Nutzer interessieren.
Die eigentliche Bedienung des Programms kommt etwas zu kurz. Ich habe dort, wo sich der Text auf die Programmbedienung bezieht, ein paar erläuternde Bildschirmfotos eingebaut. Diese Bilder sind NICHT Bestandteil der Originaltexte. Weiterhin habe ich einigige als „Zusatz DM3ML“ gekennzeichnete Erweiterungen mit Bildern und Text zur Programmbedienung eingebaut
Gut Funk und 73 de Eike, DM3ML

MMVARI-Projekt (EPROJECT.txt)

January 2, 2004 JE3HHT Makoto Mori
Translated into English by JA7UDE Nobuyuki Oba
Übersetzt ins Deutsche von Eike, DM3ML (Okt 2009)

Vorwort

Ich mache gern QSOs in Japanisch, einfach weil ich ein Japaner bin. Für innerjapanische QSOs verwende ich gern PSK31 mit japanischen Schriftzeichen. Es ist bekannt, dass PSK31 eine sehr gutes Kommunikationsmittel ist. Ich habe mich bemüht, vor allem die ostasiatischen Schriftzeichen einschliesslich Hangul und Chinesisch einzuarbeiten. Das Ziel dieses Projekt ist, eine Lösung für diese Schriftzeichen zu finden.

Zeichenkodierung und Struktur

Die ostasiatischen Sprachen wie Japanisch, Hangul und Chinesisch haben so viele Zeicheh, dass es unmöglich ist, sie in den Raum von 8 Bit (2^8=64 Möglichkeiten) zu bringen. Die aktuelle PSK31-Anwendung benutzt aus diesen Gründen den MultiByte Character Set (MBCS). Um ein MBCS-Teichen zu übertragen, sendet PSK31 zwei Bytes. Es ergeben sich dadurch aber mehrere Nachteile :

- Langsam
- Geht die Phasensynchronisierung verloren, werden eine Weile fehlerhafte Zeichen (Müll) ausgegeben
- Zeichen mit unterschiedlicher Byteanzahl werden gemischt. Dadurch funktioniert der Rückschritt (Backspace) nicht mehr fehlerfrei

Sehen wir nach, wie diese Probleme Schritt für Schritt gelöst werden können :

- Es trifft zu, dass die asiatischen Sprachen zahlreiche Schriftzeichen haben. Das ist aber nicht der eigentliche Grund, dass sie für PSK31 nicht gut geeignet sind. Das erste Byte des MBCS-Kodes ist dem Bereich $81 bis $FE zugewiesen. Davon stehen für Hiragana, den in Japan am häufigsten verwendeten Schriftzeichensatz der Unterbereich $9F bis $F2. In der PSK31-Varicodezeichentabelle sind die Kodes in diesem Bereich dem Längenkode zugeordnet. Zu allem Überfluss wird auch noch eine Lücke (Gap) mit dem Kode 00 zwischen zwei Varicode-Zeichen beim einem MBCS-Zeichensatz eingefügt. Dabei kommen 24 Bits für ein Schriftzeichen zusammen. PSK31 wird dadurch langsamer als ein CW-QSO mit japanischen Schriftzeichen.

- Wenn in einem Datenstrom ein Zeichen gestört ist, verliert der Empfänger die Synchronisation und die Zuordnung der MBCS-Zeichen. Dadurch gibt er Weile einen fehlerhaften Text aus, weil er nicht mehr feststellen kann, was das erste und das zweite Byte des MBCS-Zeichens ist. Glücklicherweise werden die Kombinationen $A0 bis $DF nicht im ersten Byte des japanischen Schriftzeichensatzes verwendet und können daher für die Phasensynchronisation verwendet werden. Andererseits verwenden Hangul und Chinesisch eine größere Anzahl von Schrifzeichen sowohl im ersten als auch im zweiten Byte.

- In einer Anwendung werden ein-Byte- und zwei-Byte-Zeichen gemischt. Die Verwendung des Backspace führt dadurch zu Fehlern. An das Japanische angepasste PSK31-Programme wie HALPSK und WINPSK/J verwenden einen Trick, um das Problem zu lösen, der aber nicht die beste Möglichkeit ist. Die effektivste Möglichkeit ist, die Zeichenabstand zu vergrößern, so dass jedes Zeichen einen eindeutigen Kode bekommt. Theoretisch gestattet der in PSK31 verwendete Varicode so viele verschiedene Zeichen, wie benötigt.

Um den MBCS-Zeichensatz ausschliesslich zu unterstützen, verwendet das Programm ausschliesslich die Zeichen im Bereich $00-$FF und $8140-$FEFF. Es werden keine Zeichen ausserhalb dieses Bereichs

verwendet. Das zweite Byte des MBCS-Satzes ist in jedenfall gleich oder größer als $40. Im Japanischen ist kein Zeichen dem Bereich $A040-$DFFF zugeordnet. In Hangul und Chinesisch gibt es einige Zeichen in diesem Bereich.

Falls jemand fragt. “Wie wäre es mit dem UNICODE?" : Beim UNICODE muss der gesamte Bereich von $0000 bis $FFFF verarbeitet werden. Er umfasst alle weltweit verwendeten Schriftzeichen, aber wenn ich Japanisch schreibe, benötigte ich keine chinesischen Schriftzeichen. Hangul soll später in den UNICODE aufgenommen werden und benötigt wahrscheinlich einen noch längeren Kode. UNICODE wird auch nicht von allen WINDOWS-Versionen unterstützt. Aus diesem Grunde bin ich vom UNICODE abgekommen.

“Und der JIS-Kode?“ : Dieser Kode unterstützt ausschliesslich die japanische Sprache und schied daher aus.

Um den MBCS-Kode in die VARICODE-Bytes $00-$FF und $8140-$FEFF umzusetzen, brauchen wir einen kontinuierlichen Index von $0000-$5F7F. Eine Regel beim VARICODE ist, das jedes Zeichen mit einer 1 anfängt und mit einer 1 endet. Innerhalb des Kodes darf es keine Folgen von zwei oder mehr Nullen geben. Auf der Basis dieser Festlegungen ergibt sich diese Kodetabelle für den VARICODE:

Index	VARICODE
$0000	1
$0001	11
$0002	101
$0003	111
$0004	1011
$0005	1101
$0006	1111
$0007	10101
$0008	10111
$0009	11011
$000A	11101
$000B	11111
:	:
$00FF	101101011011
:	:
$5F7F	110111111111101101101

Zu beachten ist, dass wenn der Index direkt in den Kode umgesetzt wird, selten verwendete Zeichen einen kurzen Kode bekommen und dadurch die Übertragung uneffektiv und verlängert wird. ASCII-Zeichen im Bereich $0000-$007F werden in PSK31-Kodes unter Berücksichtigung des Zeichenhäufigkeit umgesetzt. Um die Kompatibilität zu erhalten, bleibt der PSK31-Kode in diesem Bereich erhalten.

Sehen wir uns die Häufigkeit der japanischen Zeichen vom Gesichtspunkt der MBCS-Kodierung an. Die Tabelle zeigt die Häufigkeit des ersten Bytes in verschiedenen Dokumenten an, die ich geschrieben habe.

$81 : 6.33%	$89 : 0.95%	$91 : 2.44%	$99 : 0.00%
$82 : 51.5%	$8A : 1.68%	$92 : 2.12%	$9A : 0.00%
$83 : 8.99%	$8B : 0.54%	$93 : 2.93%	$9B : 0.00%
$84 : 0.00%	$8C : 1.79%	$94 : 1.74%	$9C : 0.00%
$85 : 0.00%	$8D : 2.66%	$95 : 4.51%	$9D : 0.00%
$86 : 0.00%	$8E : 3.75%	$96 : 1.22%	$9E : 0.00%
$87 : 0.00%	$8F : 1.47%	$97 : 1.38%	$9F : 0.00%
$88 : 1.68%	$90 : 2.04%	$98 : 0.24%	

$E0 : 0.00%	$E8 : 0.00%	$F0 : 0.00%	$F8 : 0.00%
$E1 : 0.00%	$E9 : 0.00%	$F1 : 0.00%	$F9 : 0.00%
$E2 : 0.00%	$EA : 0.00%	$F2 : 0.00%	$FA : 0.00%
$E3 : 0.00%	$EB : 0.00%	$F3 : 0.00%	$FB : 0.00%
$E4 : 0.00%	$EC : 0.00%	$F4 : 0.00%	$FC : 0.00%
$E5 : 0.00%	$ED : 0.00%	$F5 : 0.00%	$FD : 0.00%
$E6 : 0.00%	$EE : 0.00%	$F6 : 0.00%	$FE : 0.00%
$E7 : 0.00%	$EF : 0.00%	$F7 : 0.00%	

Die Tabelle zeigt, dass der erste Teil der Kodes öfter als die anderen verwendet wird. Der hintere Teil der Kodes, der mit den komplizierteren Kanji-Zeichen verbunden ist, wird seltener verwendet. Wenn wir in die Einzelheiten gehen, sehen wird, dass $82 öfter als 50% verwendet wird. Das ist der Kode für das japanische Hiragana. In der gesprochenen japanischen Sprache sind über 70% Hiragana. Daraus lässt sich ableiten, dass die kurzen Kodes vor allem dem Hiragana zugeordnet werden sollten, um die Übertragungsgeschwindigkeit zu erhöhen. In der Amateurfunkkonversation wird Katakana oft verwendet, so dass es wie Hiragana ebenfalls einen größeren Anteil an den kurzen Kodes bekommen sollte.

Innerhalb des Mehrbytekodes MBCS wird Hiragana im Bereich $829F-$82F2 definiert und korrespondiert mit den Werten $021F-$0272 im Index. Wird dieser Bereich direkt dem VARICODE zugeordnet, ergibt sich eine Zeichenlänge von 15 bis 16 Bits einschliesslich der Zeichenlücke (GAP). Wird der Bereich dem Kodebereich $0080-$00D3 zugeordnet, würde sich die Zeichenlänge auf 12 bis 13 Bits verkürzen. Katakana ist dem Bereich $0280-$02D6 zugeordnet mit den Indexwerten $8340-$8396 im MBCS, lasst uns also Katakana dem Bereich $00D4-$012A zuordnen.

Ein weiterer Punkt, der beachtet werden muss, ist die Kodezuordnung bei MBCS-Sprachen ausserhalb des Japanischen. Meine kurze Überprüfung zeigt, dass dem Bereich $0081-$00FE keine Zeichen (definiert mit dem ersten Byte) zugeordnet sind und aus diesem Grund keine Probleme bereiten. Zeichen im Bereich $0100-$012A sind als Hangul und Chinesisch definiert, aber ich kenne ihre Häufigkeit nicht. Zusätzlich können Nicht-MBCS-Sprachen ausserhalb des Englischen Sonderzeichen in diesem Bereich belegen. Hier sollte sich der Zeichensatz auf die Sprache am Empfangsort beziehen. Die japanischen Zeichen im Bereich $E040-$FEFF sollten daher in den Bereich $A040-$BEFF verschoben werden, in dem keine japanischen Zeichen definiert sind. Die Häufigkeit von Zeichen in diesem Bereich ist nicht so hoch, aber die Umsetzung in diesen Bereich ergibt eine bessere Kodiereffizienz. Der Index, $4840-$5F7F wird in den Bereich $1840-$2F7F umgesetzt.

Die Zusammenfassung der Zuordnung ergibt diese Tabelle :

$0000-$007F	Entspricht der PSK31-Kodierung
$0080-$00D3	Umgesetzt in den Bereich $021F-$0272, falls japanischer Schriftsatz
$00D4-$012A	Umgesetzt in den Bereich $0280-$02D6, falls japanischer Schriftsatz
$012B-$021E	Keine Umsetzung
$021F-$0272	Umgesetzt in den Bereich $0080-$00D3, falls japanischer Schriftsatz
$0273-$027F	Keine Umsetzung
$0280-$02D6	Umgesetzt in den Bereich $00D4-$012A. falls japanischer Schriftsatz
$02D7-$183F	Keine Umsetzung
$1840-$2F7F	Umgesetzt in den Bereich $4840-$5F7F, falls japanischer Schriftsatz
$2F80-$483F	Keine Umsetzung
$4840-$5F7F	Umgesetzt in den Bereich $1840-$2F7F, falls japanischer Schriftsatz

In diesem Bespiel ist der längste Kode bei Verwendung dieser Tabelle

$5F7F 11011111111101101101 (21Bits)

Das letzte japanische Schrifzeichen ist

$2F7F 10110101111111101111 (20Bits)

Modulation

In Japan werden in allen PSK31-QSOs, die die japanischen Zeichen verwenden zwei Varicode-Bytes verwendet, um ein MBCS-Zeichen zu übertragen. Dabei geht die Kompatibilität zu den existierenden PSK31-Programmen verloren.

Aus diesem Grund habe ich mich entschlossen, nicht PSK31 zu verwenden, sondern GMSK (BT=1.0) als experimentelle Modulation einzusetzen. GMSK ist eine Art von FSK, ist aber auch ein Verwandter von PSK. Der Klang von GMSK hört sich wie PSK31 an, es hat aber ein abweichendes Spektrum, an dem der Nutzer die Sendearten unterscheiden kann. Theoretisch hat GMSK keine Amplitudenkomponente und stellt dadurch keine Anforderungen an die Linearität des Transceivers (ist das nicht FB?).

Wie bei PSK31sendet MMVARI mit GMSK eine 0 durch Invertiertung des Symbols und eine 1 durch Beibehaltung der Lage. Dadurch ist auch die Seitenbandlage (USB/LSB) uninteressant. Eine Detaildarstellung von GMSK würde hier zu weit führen. Es sei auf die zugehörige Fachliteratur verwiesen.

Experimental-Programm MMVARI

Ich habe das PC-Soundkartenprogramm MMVARI geschrieben, um das vorgeschlagene Schema zu implementieren. Es handelt sich um ein neues VARICODE-Experiment. MMVARI enthält daher nur die Grundfunktionen für ein Standard-QSO. Es wird nur ein japanisches Menü bereitgestellt, dass aber in jeder MBCS-Sprachen verwendet werden kann. Nicht-MBCS-Sprachen wie Englisch werden bei diesem experimentellen Projekt vorerst nicht berücksichtigt. Ich werde erst einmal die Verwendbarkeit feststellen und mich dann an die Programmkosmetik machen.
Nachsatz DM3ML : Inzwischen ist das Menü auch in Englisch bereitgestellt worden und MMVARI ist für Nicht-MBCS –Sprachen verwendbar.

Anhang

Beispiel einer in C geschriebenen Indexübersetzung:

```
static int SwapHiragana(int index)
{
	if( index <= 0x00d3 ){ 	// $80-$D3
		index += 0x21f - 0x0080;	// $80-$D3 to Hiragana
	}
	else if( index <= 0x12a ){ 	// $D4-$12A
		index += 0x280 - 0x00d4;	// $D4-$12A to Katakana
	}
	else if( (index >= 0x021f) && (index <= 0x0272) ){	// Hiragana
		index -= 0x21f - 0x0080;	// Hiragana to $80-$D3
	}
	else if( (index >= 0x0280) && (index <= 0x02d6) ){	// Katakana
		index -= 0x280 - 0x00d4;	// Katakana to $D4-$12A
	}
	else if( (index >= 0x1840) && (index <= 0x2f7f) ){
		index += 0x4840 - 0x1840;
	}
	else if( (index >= 0x4840) && (index <= 0x5f7f) ){
		index -= 0x4840 - 0x1840;
	}
}

UINT Index2Mbc(int index, BOOL fJA)
{
const BYTE _tIndex2Ascii[]={
	0x20,0x65,0x74,0x6F,0x61,0x69,0x6E,0x72,	/*00*/
	0x73,0x6C,0x0A,0x0D,0x68,0x64,0x63,0x2D,	/*08*/
	0x75,0x6D,0x66,0x70,0x3D,0x2E,0x67,0x79,	/*10*/
	0x62,0x77,0x54,0x53,0x2C,0x45,0x76,0x41,	/*18*/
	0x49,0x4F,0x43,0x52,0x44,0x30,0x4D,0x31,	/*20*/
	0x6B,0x50,0x4C,0x46,0x4E,0x78,0x42,0x32,	/*28*/
	0x09,0x3A,0x29,0x28,0x47,0x33,0x48,0x55,	/*30*/
	0x35,0x57,0x22,0x36,0x5F,0x2A,0x58,0x34,	/*38*/
	0x59,0x4B,0x27,0x38,0x37,0x2F,0x56,0x39,	/*40*/
	0x7C,0x3B,0x71,0x7A,0x3E,0x24,0x51,0x2B,	/*48*/
	0x6A,0x3C,0x5C,0x23,0x5B,0x5D,0x4A,0x21,	/*50*/
```

```
        0x00,0x5A,0x3F,0x7D,0x7B,0x26,0x40,0x5E,    /*58*/
        0x25,0x7E,0x01,0x0C,0x60,0x04,0x02,0x06,    /*60*/
        0x11,0x10,0x1E,0x07,0x08,0x1B,0x17,0x14,    /*68*/
        0x1C,0x05,0x15,0x16,0x0B,0x0E,0x03,0x18,    /*70*/
        0x19,0x1F,0x0F,0x12,0x13,0x7F,0x1A,0x1D,    /*78*/
};

        if( index <= 0x007f ){
                index = _tIndex2Ascii[index];
        }
        else if( fJA ){
                index = SwapHiragana(index);
        }

        UINT mbc;
        int m, b;
        if( index >= 0x0100 ){
                index -= 0x0100;
                m = index % 192;
                b = index / 192;
                mbc = 0x8140 + m + (b * 256);
        }
        else {
                mbc = index;
        }
        return mbc;
}

int Mbc2Index(UINT mbc, BOOL fJA)
{
const BYTE _tAscii2Index[]={
        0x58,0x62,0x66,0x76,0x65,0x71,0x67,0x6B,    /*00*/
        0x6C,0x30,0x0A,0x74,0x63,0x0B,0x75,0x7A,    /*08*/
        0x69,0x68,0x7B,0x7C,0x6F,0x72,0x73,0x6E,    /*10*/
        0x77,0x78,0x7E,0x6D,0x70,0x7F,0x6A,0x79,    /*18*/
        0x00,0x57,0x3A,0x53,0x4D,0x60,0x5D,0x42,    /*20*/
        0x33,0x32,0x3D,0x4F,0x1C,0x0F,0x15,0x45,    /*28*/
        0x25,0x27,0x2F,0x35,0x3F,0x38,0x3B,0x44,    /*30*/
        0x43,0x47,0x31,0x49,0x51,0x14,0x4C,0x5A,    /*38*/
        0x5E,0x1F,0x2E,0x22,0x24,0x1D,0x2B,0x34,    /*40*/
        0x36,0x20,0x56,0x41,0x2A,0x26,0x2C,0x21,    /*48*/
        0x29,0x4E,0x23,0x1B,0x1A,0x37,0x46,0x39,    /*50*/
        0x3E,0x40,0x59,0x54,0x52,0x55,0x5F,0x3C,    /*58*/
        0x64,0x04,0x18,0x0E,0x0D,0x01,0x12,0x16,    /*60*/
        0x0C,0x05,0x50,0x28,0x09,0x11,0x06,0x03,    /*68*/
        0x13,0x4A,0x07,0x08,0x02,0x10,0x1E,0x19,    /*70*/
        0x2D,0x17,0x4B,0x5C,0x48,0x5B,0x61,0x7D,    /*78*/
};
        int index, m, b;
        if( mbc >= 0x8100 ){
                if( (mbc & 0x00ff) >= 0x0040 ){
                        mbc -= 0x8100;
                        m = mbc % 256;
                        b = mbc / 256;
                        index = 0x100 - 0x40 + m + (b * 192);
                }
                else {
                        return -1;          /* error */
                }
        }
        else {
                index = mbc;
```

```
	}
	if( index <= 0x007f ){
		index = _tAscii2Index[index];
	}
	else if( fJA ){
		index = SwapHiragana(index);
	}
	return index;
}
```

MMVARI-Entwicklungsgeschichte

BetaVer0.42 2007/JAN/15 : - MBCS-Warnung beim Start in englischer Umgebung entfernt

BetaVer0.41 2005/MAR/20
- Sendemöglichkeit für BPSK-VariJA (MBCS) freigegeben (Tnx to JE4IVN)
- Tippfehler im englischen Teil behoben
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.40 2005/JAN/08 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.39 2004/NOV/06 : - Übertragung zu Turbo Hamlog/Win Version 5 verbessert

BetaVer0.38 2004/NOV/03
- Verbindung (Gateway) zu Turbo Hamlog/Win Version 5 eingeführt (Tnx to JG1MOU)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.37 2004/OCT/08 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.36 2004/SEP/30 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.35 2004/SEP/22
- Makroanwendung erweitert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.34 2004/SEP/11
- Spezial-Macros eingebaut (z.B. OnPTT, OnQSO, OnStart)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.33 2004/SEP/08
- AFC für GMSK/PSK verbessert
- Zeitgeber (OnTimer) als Macro eingebaut
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.32 2004/SEP/04
- Abschwächgeschwindigkeit erhöht
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.31 2004/AUG/28 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.30 2004/AUG/15 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.29 2004/AUG/09 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.28 2004/AUG/07
- Fehler bei der TX/RX-Umschaltung in RTTY behoben
- Option Soundwiedergabe eingebaut
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.27 2004/AUG/03

- MFSK-Dekoder verbessert (Synchronisation, AFC, Arbeitsgeschwindigkeit usw.)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.26 2004/JUL/31
- Sendeart MFSK hinzugefügt
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.25 2004/JUL/27
- Taste Makroerweiterung eingebaut (View-Menü)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.24 2004/JUL/22
- Makroanwendungen erweitert
  Neues Kommando : <%Menu=...>
  CAT für Kenwood und JST245 mit dem Kommando <%RadioKHz=...>
  CAT-Kommando <%RadioMode=...>
* Siehe auch Datei EMMVARI.TXT und die Menüs zu Einzelheiten
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.23 2004/JUL/19
- Fehler in der AFC-Pegelberechnung behoben
- Makros zur Frequenzsteuerung eingebaut :
* Drei Makrotypen stehen nun zur Verfügung :
  <%RadioKHz=YAESU-HF,14073.000> FT1000MP, FT920 u.a...
  <%RadioKHz=YAESU-VU,14073.000> FT847, FT736 u.a
  <%RadioKHz=CI-V,14073.000> ICOM
  Beispiel: 500Hz up : <%RadioKHz=YAESU-VU,<%RadioKHz>+0.5>
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.22 2004/JUL/14
- AFC verbessert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.21 2004/JUN/29 : - Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.20 2004/JUN/28
- Mehrere Funktionen zum Rechtsklick in das Spektrum hinzugefügt
- Synchronisation für aus dem Takt geratene Signale verbessert
-- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.19 2004/JUN/26
- Kerb(Notch)-Filter hinzugefügt (mit Rechtsklick in das Spektrum)
- BPF verbessert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.18 2004/JUN/12
- Sendeart RTTY hinzugefügt
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.17 2004/JUN/04
- Makroanwendung erweitert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.16 2004/MAR/20
- MMTTY/MMSSTV – Nutzer-Toneinstellung hinzugefügt
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.15 2004/MAR/02
- Farbeinstellung des Wasserfalls durch Nutzer möglich gemacht
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.14 2004/FEB/28

- Automatische TX-Alphabet-Zeichenumsetzung vom 2-Byte- zum 1-Byte-Alphabet eingeführt
- Taktfrequenzen 22050, 44100, 48000Hz für Soundkarte eingeführt
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.13 2004/FEB/22
- Fehler in der Kanal-Umsetzung im VariSTD/JA-Modus behoben
- Der Nutzer kann die Hintergrundfarbe des Empfangsfensters einstellen
- Menü zu Rechtsklick in das Empfangsfenster eingebaut
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.12 2004/FEB/18
- Aussetzeffekte bei manchen PCs behoben
- Taktfrequenz 6000Hz für Soundkarten eingebaut
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.11 2004/FEB/15
- Aussetzeffekte bei manchen PCs behoben ?
- Makroanwendung erweitert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.10 2004/FEB/12 :- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.09 2004/FEB/09
- Sendearten BPSK und bpsk eingebaut
  BPSK – Nicht komaptibel mit HALPSK oder WINPSK/J (TX ist gesperrt in VariJA)
  bpsk – Konventioneller Standard-Varicode kompatibel mit HALPSK und WINPSK/J
  (BPSK/bpsk noch nicht getestet)
- Amplituden-Monitorfenster bei BPSK-Modulation zu Fehlerbehebung eingebaut.
  Klicken Sie rechts auf die Sync-Taste, um es zu aktivieren.
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.08 2004/FEB/04
- Editieren im Sendefenster verbessert
- Makroanwendungen erweitert
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.07 2004/JAN/27
- Fehler bei der PTT-Steuerung per CAT behoben
- Sendeart FSK modulation (fm=1.0) für V/UHF eingeführt um die Frequenzdrift zu verarbeiten
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.06 2004/JAN/23
- Werkzeug zur Kalibrierung der Soundkarte eingebaut
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.05 2004/JAN/21
- Englisches Meün eingebaut (Tnx Kim HL3IB und Sung HL1AQ)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.04 2004/JAN/19
- Textdatei EProject.txt in das MMVARI-Paket eingebaut (Tnx JA7UDE)
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer 0.03 2004/JAN/15
- Änderung des Programmnamens von MGMSK in MMVARI (wegen der Modulation)
- Änderung der Indexzuordnung für japanische Schift von $E040-$FEFF zu $A040-$BEFF
- Nutzer kann die Tasten TX und TXOFF festlegen
- Einige Programmfehler behoben und Verbesserungen eingeführt

BetaVer0.02
- Soundaufzeichnung und –wiedergabe eingebaut
- CAT-Kommandos eingebaut

- Logfunktion eingebaut
- Verbindung zu TurboHamlog/Win eingebaut
- Einige Programmfehler behoben

BetaVer0.01 2004/JAN/09
- AFC-Steuerung auf der RX-Karteikarte eingebaut
- Zeichenkode im Editmenü eingebaut
- VARICODE-Tabelle unter Options hinzugefügt

BetaVer0.00 2004/JAN/08 : - Erste Beta-Version veröffentlicht

MMVARI-Beschreibung

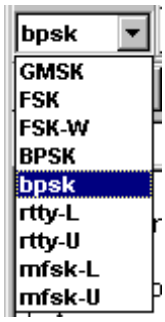
Version 0.42 (Jan 15, 2007) by JE3HHT Makoto Mori
Translated into English by JA7UDE Nobuyuki Oba
Übersetzt ins Deutsche von Eike, DM3ML (Okt 2009)

Übersicht

Lesen Sie bitte zuerst das Kapitel MMVARI-Projekt (Original . EPROJECT.TXT). Es beschreibt die Zielsetzung für MMVARI.

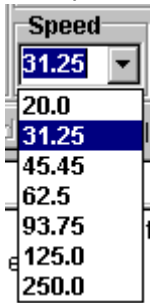
Dieses Papier beschreibt die Grundfunktionen des Programms MMVARI. MMVARI befindet sich noch in Entwicklung. Sie finden zu den einzelnen Menüpunkten interessante Tipps in der Statusleiste unterhalb des Hauptfensters, wenn Sie den Mauscursor darauf setzen.

Sendearten



MMVARI ab Version 0.25 unterstützt die folgenden Sendearten :
- GMSK : Voreingestellt für den Multibyte-Zeichensatz MBCS
- FSK : Für V/UHF-Geräte, die ungenügend frequenzstabil sind
- BPSK : Kompatibel mit HALPSK, WINPSK/J, Standard-VARICODE
- bpsk : Kompatibel mit HALPSK, WINPSK/J, Standard-VARICODE
- rtty : ist nur eine schnelle Erweiterung von MMVARI. Implementiert wurde nur der Standard-BAUDOT-Kode. Unterstützt werden nur Großbuchstaben, Zahlen und eine Untermenge an Sonderzeichen.
- mfsk : MFSK16 wurde implementiert. Der Kode ist MFSK-VARICODE.

Hinweis : Stellen Sie den NF-Pegel Ihres Senders sorgfältig ein. Verwenden Sie keinen Kompressor! Achten Sie darauf, dass die Transceiver-ALC gerade noch nicht anspricht.



Für die **PSK**-Sendung sind folgende Datenraten wählbar :

PSK31	31.25Bps
PSK63	62.5Bps
PSK125	125.0Bps
PSK250	250.0Bps

RTTY wurde wie folgt implementiert :

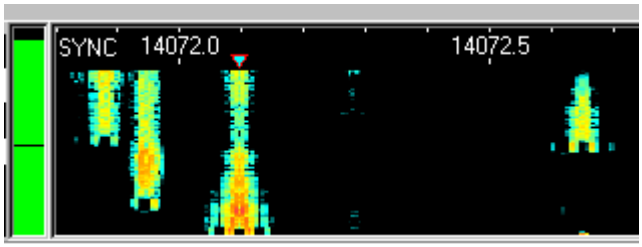
- Shift : 170Hz shift. Die Shift kann mit dem Macro <%TxShift...>, <%RxShift...> geändert werden.
- Trägerfrequenz : Mitte zwischen Mark und Space
- Seitenbandlage : RTTY-L mit LSB (voreingestellt), RTTY-U mit USB
- Kode : 5-bit BAUDOT S-BELL (64 Zeichen)
- Zurückschaltung in den Bu-Kode bei RX (UOS) ist voreingestellt eingeschaltet.
  Es kann mit dem Macro <%UOS=ON/OFF/ONOFF> umgeschaltet werden.
- Die Funktion UOS beim Senden ist immer eingeschaltet.
- Das Pausenzeichen Bu-Umschaltung (Diddle) ist immer eingeschaltet. Der Diddle kann
  mit dem Macro <%DIDDLE=BLK/LTR> geändert werden.
- Getastet wird nur inAFSK. FSK wird nicht unterstützt.

MFSK16 wurde wie folgt implementiert :

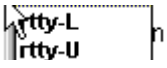
Baudrate	15.625 bps
Anzahl der Töne	16
Abstand der Töne in Hz	15.625
Max. Breite des Signals in Hz	234.375
Viterbi-Kode	NASA K=7, R=1/2
Verschachtelung	Diagonal interleaver (AFKP...)
VARICODE	MFSK-Standard

Signal beim Empfang abstimmen

Wenn Sie ein Signal empfangen möchten, klicken Sie auf die Mitte des Signals im FFT-Spektrum oder im Wasserfall. Sie können die Breite der Skala in den Schritten 500Hz, 1kHz, 2kHz und 3kHz wählen. Günstige Werte sind 1K oder 2K. Zusätzlich können Sie die Transceiverabstimmung optimieren. Die grüne Leiste links im Wasserfall zeigt die relative Signalstärke an. Die schwarze Linie ist mit der Maus verschiebbar und steuert den Squelch (Schwelle, über der das Signal dekodiert wird).

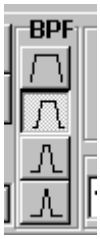


Für GMSK, FSK, BPSK und bpsk können Sie am Transceiver LSB oder USB wählen. Die Seitenbandlage ist für die Dekodierung uninteressant.
In RTTY müssen Sie unter LSB die Sendeart rtty-L und unter USB die Sendeart rtty-U wählen. Sie erreichen damit in beiden Fällen die Solllage für Mark und Space.



Wenn Sie auf ein Signal in der Sendeart MFSK abstimmen wollen, schalten Sie je nach Seitenbandlage wie bei RTTY auf mfsk-L (LSB) oder mfsk-U (USB). Klicken Sie bei mfsk-L im Wasserfall auf die rechte Begrenzung des Signalbandes und bei mfsk-U auf die linke Begrenzung.

Bandpassfilter BPF



Sie können bei MMVARI zur Verbesserung der Selektion symmetrisch zum Signal liegende Bandpassfilter zusätzlich einschalten, um das QRM zu unterdrücken. Je schmaler das Filter gewählt wurde, desto mehr Rechnerleistung benötigt MMVARI.

Automatische Zeitsteuerung ATC

Die automatische Zeitsteuerung ATC synchronisiert die Abtastung des empfangenen Signals. Lassen Sie die ATC im Normalfall eingeschaltet. Informieren Sie sich Kapitel zum Taktabgleich (clock calibration) über Einzelheiten.

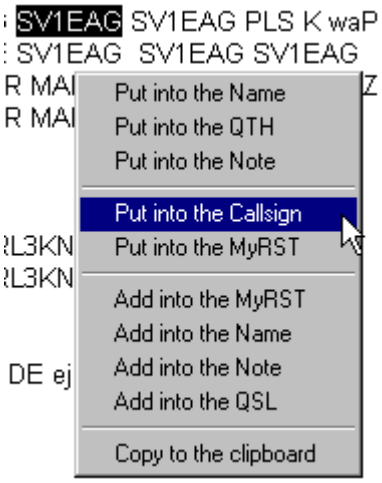
Hinweis 1 : Die ATC hat nicht mit der automatischen Schwellensteuerung (Automatic Threshold Control) von MMTTY zu tun.

Hinweis 2 : MMVARI verwendet ab Version 0.20 eine neue Synchronisiermethode zum Signalempfang. Diese Methode ergibt bessere Dekodierergebnisse, auch wenn die ATC dem Signal nur mit Mühe folgen kann. Sie sollten aber in jedem Fall den Takt Ihrer Soundkarte kalibrieren, um möglichst gute Ergebnisse bei der Dekodierung zu erreichen. Informieren Sie sich Kapitel zum Taktabgleich (clock calibration) über Einzelheiten.

Empfangs-Tipps

Mit einem Mausklick können Sie ein Wort aus dem Empfangsfenster in das Log transportieren.

MMVARI sucht automatisch das Rufzeichen der Gegenstation und das empfangene RST und überträgt sie in das zugehörige Feld im Log.
Wenn Sie auf ein anderes Wort klicken, öffnet sich ein Menüfenster, in dem Sie wählen können, welchem Feld des Logs das Wort zugeordnet werden soll. Bei einem Rechtsklick auf ein Wort wird in jedem Fall das Auswahlmenü geöffnet.
Mit dem Scrollbalken auf der rechten Seite des Empfangsfensters, können Sie den Empfangspuffer um bis zu 1024 Zeilen zurück verfolgen.



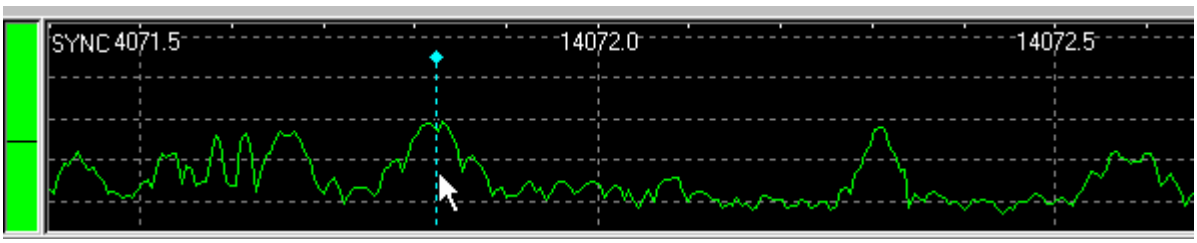
Squelch
(Zusatz von DM3ML)

Wenn Sie bei gedrückter linker Maustaste den Cursor auf den schwarzen Strich setzen, können Sie die Schwelle, ab der ein Nutzsignal dekodiert wird, nach oben oder unten verschieben. Sie können dadurch vom Rauschen erzeugte Fehlausschriften im RX-Fenster unterdrücken.

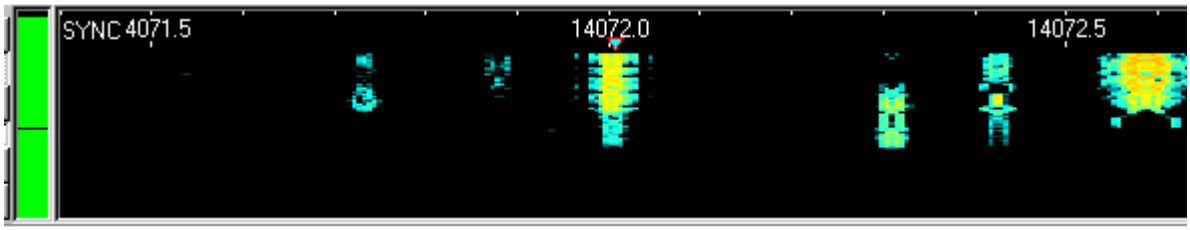
Wahl der Anzeige
(Zusatz von DM3ML)

Mit diesen Tasten können Sie die Signalanzeige im rechten oberen Bereich auswählen :

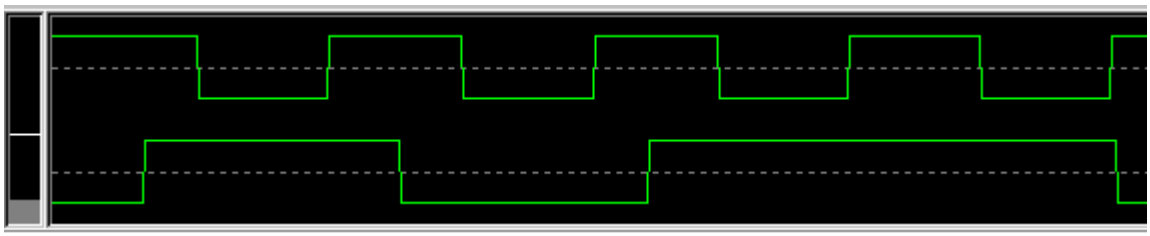
FFT = Fast Fourier Transformation (schnelle Fourier-Transformation) = Spektrumsanzeige :



W.F. = Wasserfall



Sync : Synchronisation



Sende-Tipps

Sendefenster

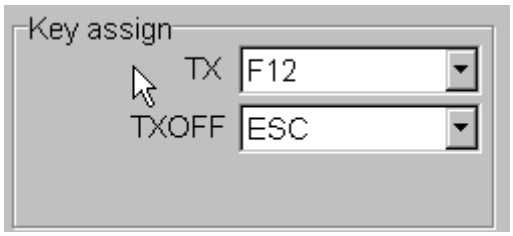
Sie können zu sendenden Text in das Sendefenster im Voraus eingeben. Beim Umschalten auf Senden werden die Zeichen aus dem Sendepuffer bis zur Cursorposition gesendet.

Sende-Empfangs-Schalter

Wollen Sie auf Senden schalten, klicken Sie auf die Sendetaste oben links im Hauptfenster.

Ein weiterer Klick auf diese Taste schaltet auf Empfang zurück, nach dem alle Zeichen aus dem Sendefenster gesendet worden sind.

Mit einem Klick auf die Taste TXOFF wird MMVARI unmittelbar auf Empfang zurück geschaltet.



Hinweis: Normalerweise entspricht die Taste ESC der Tastatur der Taste TXOFF von MMVARI. Wollen Sie die Taste ESC für einen anderen Zweck verwenden, können Sie der MMVARI-Taste TXOFF auf der Karteikarte TX der MMVARI-Einstellung eine andere Tastatur-Taste zuordnen.

Tasten NET und AFC
(Zusatz von DM3ML)

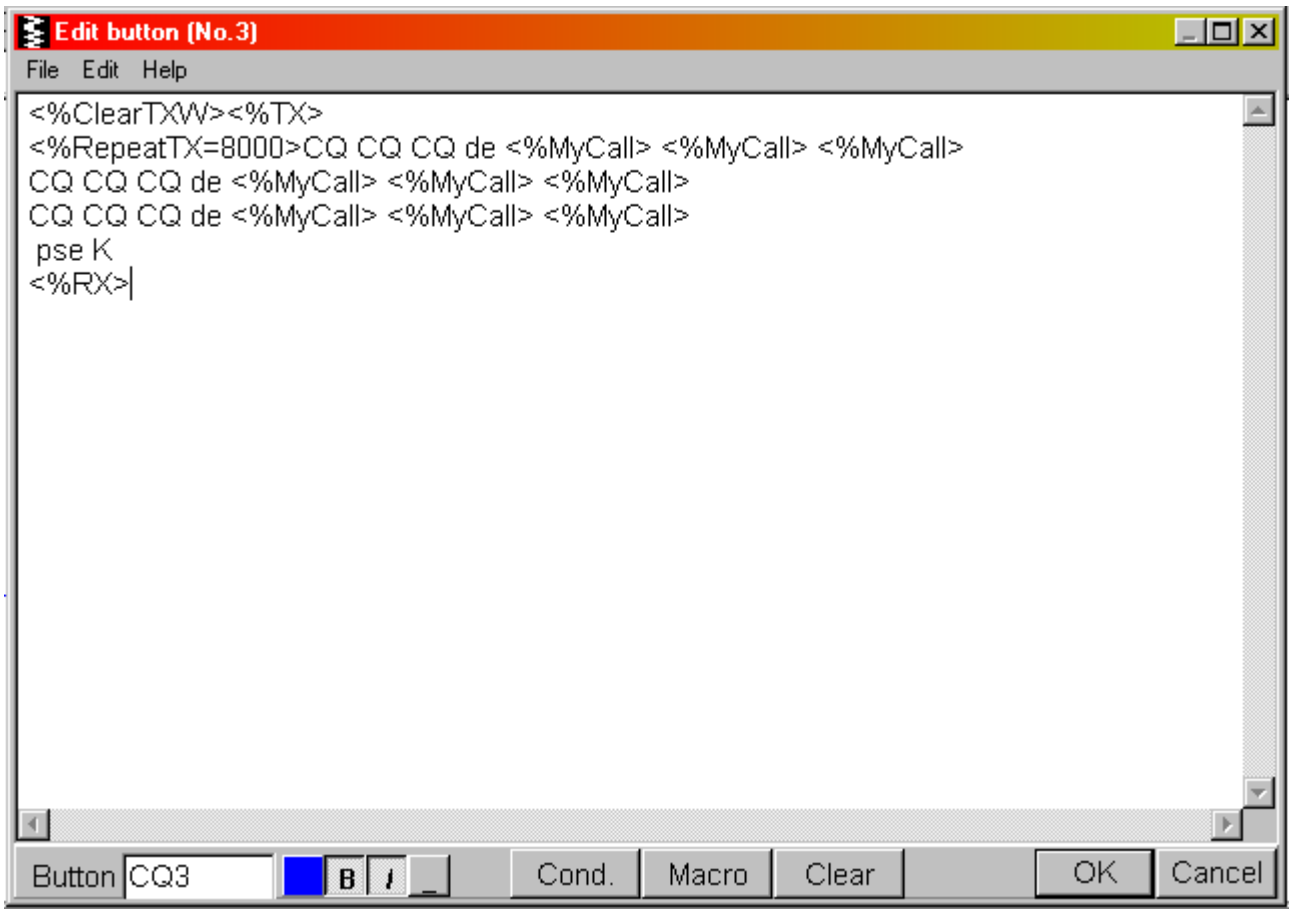
Ist die Taste ***NET*** gedrückt, senden Sie auf der gleichen NF-Frequenz, auf der Sie auch empfangen. Diese Einstellung sollten Sie beim Anruf einer Gegenstation wählen. Wenn Sie ***NET*** ausschalten, ist Ihre Sendefrequenz von der Gegenstation unabhängig. Wählen Sie diese Einstellung, wenn Sie selbst CQ rufen und sich nicht von der Gegenstation über das Band ziehen lassen wollen.

Die Taste ***AFC*** schaltet die automatische Scharfabstimmung ein (Automatic Frequency Control). Lassen Sie die AFC nach Möglichkeit immer eingeschaltet und schalten Sie sie nur dann aus, wenn eine starke Station im Nachbarkanal die Abstimmung an sich zieht, Sie aber eine leise Station daneben mitschreiben wollen.

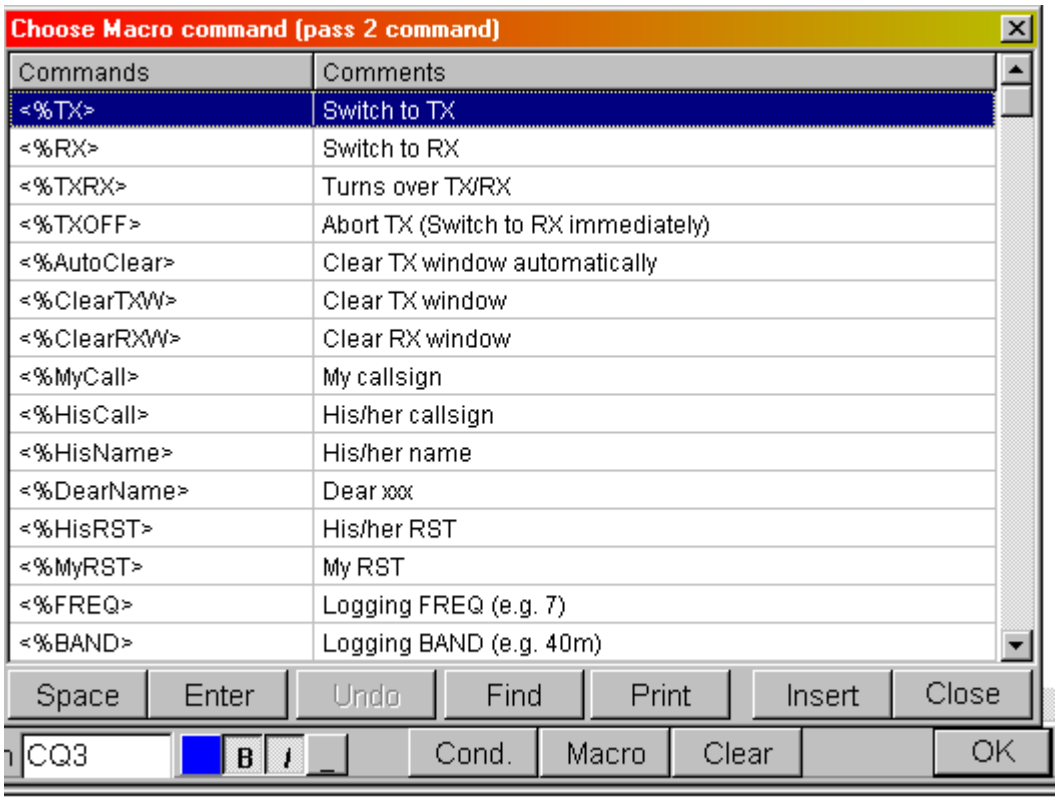
Macros

MMVARI hat 144 programmierbare Macro-Tasten. Unterhalb des Hauptfensters werden jeweils 36 Tasten angezeigt. Sie können vier Gruppen zu je 36 Tasten mit den Scrolltasten rechts vom Macro-Feld durchschalten.

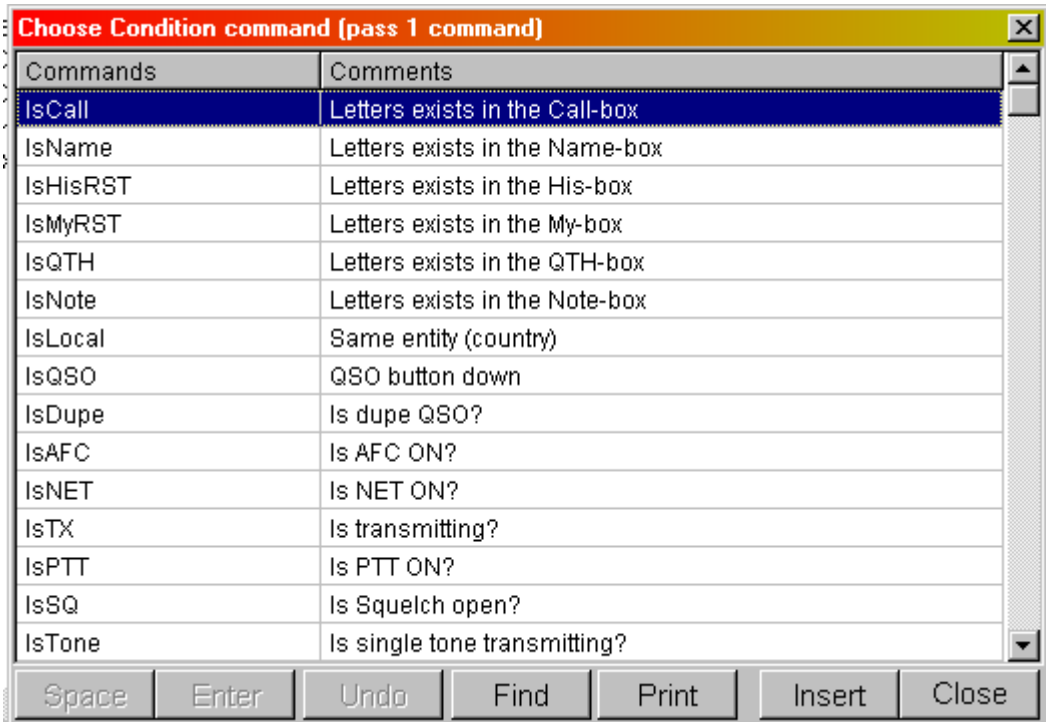
MMVARI verfügt über umfangreiche Möglichkeiten zur Macroprogrammierung, Sie können Texte ersetzen, das Programm steuern und Bedingungen vorgeben Mit einem Linksklick auf ein Macro, wird dessen Inhalt in das TX-Vorschreibfenster geholt. Mit einem Rechtsklick auf eine Macrotaste können Sie das zugehörige Macro aus einer Kombination von Text und einzelnen Macros programmieren.



Klicken Sie auf die Taste ***MACRO*** im Editierfenster, um eine Liste der Macros anzuzeigen.



Wählen Sie aus dieser Liste ein gewünschtes Macro aus. Mit einem Klick auf die Taste ***COND*** können Sie eine Bedingung zum Aktivieren eines bestimmten Macros auswählen.



Sie können ein Macro auch mit einer F-Taste starten.

Vorbemerkungen

Ein Macro ist eine Kombination von Kommandos, Text und Platzhaltern. Kommandos werden z.B. für die Sende-Empfangsumschaltung (<%TX> oder <%RX>) benötigt Die Platzhalter stehen z.B. für den Namen

der Gegenstation <%HisName> oder den Standort der Gegenstation <%HisQTH>. Sie sind erst dann verfügbar, wenn sie in das Logfenster eingetragen worden sind. Stehen sie nicht zur Verfügung, wird ein Leerzeichen ausgegeben oder bei nicht erfüllter Bedingung z.B. ‚lsName' durch ‚dr OM' ersetzt.

Position und Auswirkung eines Macros

Die Macros <%RX> und <%TX> werden unabhängig von ihrer Position innerhalb eines Macros wirksam. Die drei Beispiele werden unabhängig von der Anordnung von <%TX> und <%RX> auf gleiche Weise ausgeführt: MMVARI schaltet auf Senden, sendet den Text ‚VVV123' und schaltet zurück auf Empfang,

<%TX>VVV<%RX>123
<%TX><%RX>VVV123
VVV<%TX><%RX>123

Die Kommandos <%MoveTop> und <%MoveEnd> setzen ein Macro unabhängig vom schon in das TX-Vorschreibfenster eingegebenen Text an den Anfang (Top) oder das Ende (End) des zu sendenden Textes.
.

<%MoveTop><%HisCall> de <%MyCall> I had QRM in your last transmission.

<%MoveEnd>

Ausgabe eines Macros

Ein Macro wird bei seinem Aufruf an der aktuellen Cursorposition im TX-Vorschreibfenster ausgegeben. Enthält das Macro das Kommando <%RX> wird es unabhängig vom Kommando <%MoveEnd> nach dem Text mit dem <%TX>-Kommando eingefügt.

CWID

Das Kommando CWID erzeugt eine programmierbare CW-Zeichenfolge, die als A2-Signal auf der aktuellen Trägerfrequenz gesendet wird. Falls das Macro nur eine CWID enthält, sendet MMVARI den CW-Text. Wenn das Macro sowohl einen Text als auch eine CWID enthält, sendet MMVARI in der vorgegebenen Reihenfolge. Wollen Sie die CWID senden und unmittelbar danach auf Empfang gehen, setzten Sie KEINEN Wagenrücklauf (CR) an das Ende des Macros, z.B.

<%TX>TNX AGN 73, SK...<%CWID><%RX><%EOF>

Automatisches Löschen des Sendefensters

Mit dem Kommando <%AutoClear> wird das Sendevorschreibfenster jeweils automatisch gelöscht, wenn Sie auf Empfang zurückschalten:

BTU <%HisCall> de <%MyCall> KN
<%RX><%AutoClear>

Hinweis: Haben Sie die Sendung von Hand mit TXOFF gestoppt, funktioniert <%AutoClear> nicht.

Sendewiederholung

Das Kommando <%RepeatTX=xxxx> wiederholt ein Macro nach xxxx Millisekunden. Sie benötigen innerhalb eines Macros mit diesem Kommando keine S/E-Umschaltung mit <%TX> und <%RX>. Wenn Sie z.B. das Kommando <%RepeatTX=5000> in Ihr Macro eingebaut haben, wird es aller 5s gesendet.
Das Macro bleibt gedrückt und wirkt so lange, bis sie das gleiche Macro oder ein anderes anklicken.

Hinweis: Ein Klick in das Spektrum, den Wasserfall, das RX- oder das TX-Fenster beendet das Macro ebenfalls.

Das folgende Beispiel sendet solange aller 5s ein CQ bis ein empfangenes Signal den Squelchpegel überschreitet

```
#define ReceiveTime     5000
#if !IsSQ
<%RepeatTX=ReceiveTime><%ClearTXW>
CQ CQ CQ de <%MyCall> <%MyCall> <%MyCall> pse (<%VARITYPE>) K
#endif
```

Hinweis: Wollen Sie den TX ohne ein TX-Kommando einschalten, verwenden Sie das Kommando < %Repeat=...>. command.

Das nächste Beispiel zeigt die Reaktion von MMVARI auf eine empfangene CQ-Zeichenkette:

```
#define _CaptureString CQ
#if !IsRepeat
#macro <%SetCaptureLimit>
#endif
#if IsCaptureText(<%String=_CaptureString>)
#macro <%Message="<%String=_CaptureString>" was detected>
#else
#macro <%Repeat=1000>
#endif
```

Ausführung eines Programms

Mit dem Kommando <%Execute=xyz> können Sie ein Programm Ihrer Wahl starten.'xyz' steht für den Programmnamen und die zugehörigen Parameter. Soll das zu startende Programm die von MMVARI bisher benutzte Soundkarte verwenden, geben Sie die Soundkarte vorher mit dem Kommando <%Suspend> frei. Mit dem folgenden Beispiel wird die Soundkarte freigegeben und dann das Programm MMSSTV gestartet:

```
<%Suspend><%Execute=C:\MMSSTV\MMSSTV.exe>
```

Wollen Sie MMVARI vor dem Start von MMSSTV beenden, verwenden Sie das Kommando <%Exit>:

```
<%Suspend><%Execute=C:\MMSSTV\MMSSTV.exe><%Exit>
```

Hinweis: Sie können den Windows-Pfad durch Spezifikation des vollen Pfads überschreiben.

Bedingungen innerhalb eines Macros

Bei den MMVARI-Macros können Sie auf unterschiedliche Bedingungen zum Zeitpunkt der Aktivierung des Macros reagieren und eine von mehreren Möglichkeiten durch das Macro als Text senden.

Die Syntax für einen Bedingungsblock sieht so aus:

```
#if test1          fall die Bedingung 'test1' erfüllt ist
```

```
  |                    wird der hier stehende Text gesendet
#elseif test2    falls die Bedingung 'test1' nicht erfüllt ist, aber 'test2' zutrifft
  |                    wird der hier stehende Text gesendet
#else            falls weder Bedingung 'test1' noch 'test2' erfüllt ist
  |                    wird der hier stehende Text gesendet
#endif           Ende des Bedingungsblocks
```

Die Syntax ist wie folgt einzuhalten:

- Der Bedingungsblock beginnt mit ***#if*** und endet mit ***#endif***.
- Das Bedingungskommando #if oder #elseif und das zugehörige Argument (test1, test2) müssen in der gleichen Zeile stehen. Weitere Macrokommandos müssen nicht in der gleichen Zeile stehen
- #elseif und #else können weggelassen werden
- Die Bedingungen können in bis zu 64 Ebenen angeordnet sein
- Ein '!' negiert die Bedingung. "#if !IsCall" bedeutet z.B., dass die Bedingung erfüllt ist, wenn KEIN Rufzeichen im Rufzeichenfeld des Logs steht.
- Es gibt drei Typen von Argumenten
    - Is: Wahr (True) oder Falsch (False) (Boolesche Algebra)
    - Str: Zeichenkette
    - Val: Wert
- Für den Vergleich der Argumente ‚Zeichenkette' und ‚Wert' gelten diese Operanden :
    - = Wahr, wenn gleich
    - != Wahr, wenn ungleich
    - > Wahr, wenn größer als
    - < Wahr, wenn kleiner als
    - >= Wahr, wenn größer oder gleich
    - <= Wahr, wenn kleiner oder gleich
    - >> Wahr, wenn die Zeichenkette das angegebene Wort enthält

  Beispiele :

  ```
  #if ValFreq >= 144                Wahr, wenn die Frequenz größer oder gleich 144 ist
  #if StrMacro(<%HisQTH>) >> Osaka  Wahr, wenn das QTH der Gegenstation die
  Zeichenkette 'Osaka' enthält
  ```

- Die Bedingungen können durch die Operatoren && (UND) und || (ODER) verknüpft werden

  Beispiele:

  ```
  #if IsNET && IsAFC    Wahr, wenn NET UND AFC eingeschaltet sind
  #if IsNET || IsAFC    Wahr, wenn NET ODER AFC eingeschaltet sind
  ```

Das folgende Beispiel ruft die Gegenstation nur an, wenn im Rufzeichenfeld ein Rufzeichen eingetragen worden ist:

```
#if IsCall
<%TX><%RX>
<%HisCall> <%HisCall> de <%MyCall> <%MyCall> pse k
#endif
```

In diesem Beispiel werden unterschiedliche Texte gesendet, abhängig davon, ob die Gegenstation aus dem eigenen Land (Local) kommt oder nicht:

```
<%TX>
#if IsLocal
Hallo, freut mich einen Landsmann zu treffen. Danke fuer den Anruf.
#else
Hello, thanks for your call.
#endif
```

Dieses Beispiel sendet den CQ-Ruf in einer Sendeart abhängig von der eingestellten Frequenz (hier ab 144 MHz aufwärts in FSK, darunter in GMSK):

```
#if ValFreq >= 144
<%MODE=FSK><%SkipCR>
#else
<%MODE=GMSK><%SkipCR>
#endif
<%TX>
CQ CQ CQ de <%MyCall> <%MyCall> <%MyCall> pse (<%VARITYPE>) k
<%RX>
```

In diesem Beispiel wird das Texteingabefenster geöffnet und der eingegebene Text in CW gesendet:

```
#if StrMacro(<%Input=Input CW text>)
<%TX><%RX><%CWID=<%Input$>><%EOF>
#endif
```

Wenn Sie rechts in den Wasserfall oder das FFT-Spektrum klicken, öffnet sich ein Menüfenster. Sie können ein ‚AS' (Warten!) in CW damit senden. Dieses AS ist innerhalb von MMVARI mit diesem Macro programmiert. Mit den verschiedenen TX-Seiten wird verhindert, dass die Seite, die Sie gerade editieren, nicht verloren geht.

```
#if !IsTX
<%AutoNET><%AutoReturn><%SkipCR>
#if ValPage!=4
<%Page=4><%SkipCR>
#else
<%Page=3><%SkipCR>
#endif
<%ClearTXW><%SkipCR>
#endif
<%TX><%CWID=@><%RX><%EOF>
```

Zeichenketten-Variablen

Sie können innerhalb von MMVARI-Macros Zeichenketten-Variablen definieren. Eine Zeichenketten-Variable wird mit ***#define*** und erzeugt ein Zeichenkette <%String=name>. Die definierte Variable kann als Argument in einem Bedingungs-Macro oder als Parameter in einem Kommando verwendet werden.

Die Syntax von #define ist : #define Name String

Name steht für eine beliebige Zeichenkette aus alphanumerischen Zeichen, die NICHT mit einer Zahl beginnt. Die Zeichenkette kann aus beliebigen Zeichen bestehen. Sie kann auch ein Kommando enthalten.

```
Beispiele   #define Greetings    MAIDODESU...
            #define NowSpeed    <%BAUD>
            #define ImaSpeed    NowSpeed
```

Hinweis : Die Zahl der mit ***#define*** definierten Variablen ist nicht begrenzt. Falls eine bestehende Variable neu definiert wird, wird der alte Wert durch den neuen überschrieben. Alle Variablen haben globalen Charakter. Eine Variable innerhalb eines Macros kann durch ein anderes Macro definiert werden. Die Werte aller Variablen werden mit dem Programmende von MMVARI gelöscht.

Beispiel für eine #define-Instruktion :

In diesem Macro wird eine aktuelle Wettermeldung erzeugt, falls noch keine erzeugt wurde. Wenn die Wetter-Zeichenkette (TodayWX) mit der Eingabeanforderung <%Input> definiert ist, kann sie in den Wettermacros verwendet werden.

```
#if !IsDefined(TodayWX) || !StrMacro(<%String=TodayWX>)
#define TodayWX        <%Input=Input_WX>
#endif
#if StrMacro(<%String=TodayWX>)
Today's weather is <%String=TodayWX>.
#endif
```

Im nachstehenden Beispiel werden die aktuellen Werte der Stationsbeschreibung definiert :

```
#define Boundary       14
#define MyRIG          FT1000MP(50W)
#define MyLANT         Vertical(7m)
#define MyHANT         Magnetic loop(90cm)
#if ValFreq < Boundary
#define MyANT          MyLANT
#else
#define MyANT          MyHANT
#endif
My rig is <%String=MyRIG>, and antenna is <%String=MyANT>.
```

In diesem Beispiel wird eine QTH-Angabe in Abhängigkeit von der JCC-Nummer erzeugt :

```
#define QTH_HOME    Takatsuki-city
#define QTH_25006   Mishima-gun
#define QTH_2701    Kobe-city
#if StrNote >> 25006
#define MyQTH       QTH_25006
#elseif StrNote >> 2701
#define MyQTH       QTH_2701
#else
#define MyQTH       QTH_HOME
#endif
My QTH is <%String=MyQTH>.
```

Ausgabeformat und eingebauter Rechner

<%Format=...> : Dieses Kommando ist ein Macro, das das Ausgabe-Druckformat mit einer kleinen Rechnung festlegt :

```
<%Format=format,equation>
```

Die Notation entspricht der Syntax des Befehls printf(...) in der Programmiersprache C. Sie können das Symbol in den [] weglassen.

```
%[-][#][0][w][.p]TYP
```

-	Links bündig
0	Führende Nullen zulassen
w	Mindestzahl an Zeichen zur Ausgabe kopieren
.p	Kopiert das Minimum eines Bruchs zur Ausgabe
TYP	Formattype der Ausgabe

TYP	Type	Ausgabe
c	Character	Ein Zeichen
s	String	Zeichenkette
d	Integer	Dezimalzahl mit Vorzeichen

i	Integer	Dezimalzahl mit Vorzeichen
o	Integer	Oktalzahl ohne Vorzeichen
u	Integer	Dezimalzahl ohne Vorzeichen
x	Integer	Hexadezimalzahl ohne Vorzeichen (0..9, a..f)
X	Integer	Hexadezimalziffer ohne Vorzeichen (0..9, A..F)
f	Fließkommazahl	[-]dddd.dddd mit Vorzeichen
e	Gleitkommzahl mit Exponent	[-]d.dddd or e[+/-]ddd mit Vorzeichen
g	Gleitkommazahl im Format e oder f mit Vorzeichen	
E	Gleitkommazahl wie e aber E als Exponent	
G	Gleitkommazahl wie g aber E als Exponent	

Gleichung

Sie können die Argumente mit Operatoren verknüpfen. Die Berechnungen werden unmittelbar ausgeführt und in das Macro eingebaut.

Diese arithmetischen Operatoren werden unterstützt :

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Restbetrag

Die Priorität entspricht den Regeln der Arithmetik. Soll die Ordnung geändert werden, müssen Sie Klammern verwenden :

10 + 20 * 30 = 610
(10 + 20) * 30 = 900

Beispiele
Baudrate verdoppeln:

<%BAUD=<%Format=%f,<%BAUD>*2>>

Zweite Ziffer des gegebenen RST, hier 7 von 579 :

<%Format=%c,<%Skip$=1,<%HisRST>>>

RX-Frequenz 100 Hz höher als die TX-Frequenz einstellen :

<%TxCarrier=<%Format=%d,<%RxCarrier>+100>>

Nutzer-bezogenes Menüfenster erzeugen

Mit den Macros kann sich der Nutzer sein eigenes Macro-Auswahlfenster schaffen, um die Macros für den täglichen Bedarf zusammenzufassen.

<%Menu=...> das Kommando erzeugt ein Menüfenster. Die ausgewählten Zeichenketten in dem Menü werden durch das Kommano ***<%Input$>*** ausgewählt. Wahlweise kann die ausgewählte Zeichenkette durch das Kommando ***ValMenu*** bestimmt werden.

Die Menüpunkte werden durch ein Komma voneinander getrennt. Falls die Zeichenkette ein Komma enthält, müssen Sie sie in „ setzen. In der Zeichenkette kann ein Macro enthalten sein. Eine Begrenzung für die Zahl der Macros besteht nicht.

<%Menu=Menu1, Menu2, Menu3, Menu4, ...>
<%MenuB=Index, Menu1, Menu2, Menu3, Menu4, ...>

Hinweis : <%MenuB=...> setzt einen schwarzen Gliederungspunkt vor jeden Eintrag

Mit einem Minuszeichen als Trennzeichen können Sie eine Leerzeile zwischen die Menüeinträge setzen. Der Zugriffsschlüssel wird durch ein & vor der Menünummer definiert :

<%Menu=Menu&1, Menu&2, -, Menu&3, Menu&4, ...>

<%MenuB=Index1, Menu&1, Menu&2, -, Index2, Menu&3, Menu&4, ...>

Das folgende Beispiel zeigt ein Menü, das Ende eines QSO, das in das TX-Fenster transportiert wird.

<%Menu=<%DearName> Thank you..., Sayonara..., "TNX AGN <%DearName>, 73..."><%Input$>

Um die Operation durch das Index-Menü festzulegen verwenden Sie die Eigenschaft von ***ValMenu***. Im speziellen Fall wird ***if*** im ersten Schritt interpretiert. Setzen Sie ein ***#macro*** davor, um den Ausdruck **< *%Menu=...>*** ebenfalls im ersten Schritt zu interpretieren.

```
#macro <%Menu=&73 CU SK, &TU SK EE, &SU, &EE>
#if ValMenu==1
<%TX><%RX><%CWID=73CU:><%EOF>
#elseif ValMenu==2
<%TX><%RX><%CWID=TU:EE><%EOF>
#elseif ValMenu==3
<%TX><%RX><%CWID=SU><%EOF>
#elseif ValMenu==4
<%TX><%RX><%CWID=EE><%EOF>
#endif
```

Hinweis : Falls die Menüauswahl abgebrochen wird, gibt ***ValMenu*** eine 0 zurück und die Zeichenkette **< *%Input$>*** ist leer.
Hinweis : Das Trennzeichen kann nicht gewählt werden.***ValMenu*** enthält kein Trennzeichen.
12. Rig control using Macro

Transceiversteuerung mit CAT-Kommandos

Mit diesen Kommandos kann MMVARI Ihren Transceiver steuern ;

<%RadioKHz>	Gibt die aktuelle VFO-Frequenz zurück
<%RadioKHz=...>	Setzt die VFO-Frequenz
<%RadioMode>	Gibt die am Transceiver eingestellte Sendeart zurück
<%RadioMode=...>	Setzt die Sendeart am Transseiver
<%RadioOut=...>	Gibt ein Kommando an den Transceiver
#if IsRadioLSB	Fragt die Seitenbandlage ab, hier LSB

Die Kommandos können auch über ein Netzwerk gegeben werden, ein direkter Anschluss des Transceivers ist aber zu bevorzugen.

Hinweis : Die CAT-Steuerung funktioniert nur, wenn sie richtig eingestellt ist. Falls das nicht der Fall ist, werden die Kommandos ignoriert. Gehen Sie zu Options > Setup Radio Command…(hier für einen Elecraft K3):

Hinweis : Um die Kommandos <%RadioKHz> und <%RadioMode> effektiv wirksam zu machen, muss die Abfragerate (Polling interval) richtig eingestellt werden. Sie geben die Werte zurück, die mit <%RadioKHz=...> und <%RadioMode=...> eingestellt wurden.

<%RadioKHz=RigType,RigFreq(KHz)>

Dieses Kommando setzt die VFO-Frequenz eines Geräts ***RigType***. Im Moment unterstützt MMVARI diese Gerätetypen :

YAESU-VU	FT847, FT736, etc.
YAESU-HF	FT1000MP, FT920, etc.
CI-V	ICOM (5-Byte-Kommandos, alle ICOM-TCVR ausser IC-735)
CI-V4	ICOM (nur IC-735, 4 Byte-Kommando)
KENWOOD	KENWOOD
JST245	JRC JST245

Dieses Beispiel erzeugt ein Menüfenster zum Einstellen der Frequenz für einen YAESU-Kurzwellen-Transceiver :

```
#define _Rig     YAESU-HF
#macro <%IME=OFF>
#if StrMacro(<%Input=Input VFO FREQ(KHz)>)
#if IsRadioLSB
#macro <%RadioKHz=_Rig,<%Input$>+<%RxCarrier>*0.001>
#else
#macro <%RadioKHz=_Rig,<%Input$>-<%RxCarrier>*0.001>
#endif
#endif
```

Mit diesem Menü kann ein YAESU-Transceiver auf voreingestellte Frequenzen gesetzt werden :

```
#define _Rig     YAESU-VU
#macro <%Menu=7028.5,10141.5,14072.5,18102.5,21072.5,28072.5>
#if ValMenu
#if IsRadioLSB
#macro <%RadioKHz=_Rig,<%Input$>+<%RxCarrier>*0.001>
#else
#macro <%RadioKHz=_Rig,<%Input$>-<%RxCarrier>*0.001>
#endif
#endif
```

Hier wird die Frequenz eines JST-245 um 500 Hz erhöht :

```
#define _Rig     JST245
#macro <%RadioKHz=_Rig,<%RadioKHz>+0.5>
```

Dieses Beispiel setzt die Empfangs-NF (RxCarrier) auf 1750 Hz, um zu verhindern, dass mit einer zu niedrigen NF-Frequenz auf Senden geschaltet wird und korrigiert die VFO-Frequenz abhängig von der Einstellung auf LSB oder USB entsprechend :

```
#define _Tone   1750
#define _Rig     CI-V
#define _OffKHz          <%Format=%f,(<%RxCarrier>-_Tone)*0.001>
#if IsRadioLSB
#macro <%RadioKHz=_Rig,<%RadioKHz>-_OffKHz>    #else
#macro <%RadioKHz=_Rig,<%RadioKHz>+_OffKHz>
#endif
#macro <%RxCarrier=_Tone>
```

<%RadioMode=RigType,RigMode>

Mit diesem Kommando wird die Sendeart am Transceiver eingestellt. Die Sendeart kann wahlweise auf LSB, USB, CW, AM, FM, RTTY, PACKET gesetzt werden :
Das Beispiel schaltet einen KENWOOD-Transceiver auf LSB : <%RadioMode=KENWOOD,LSB>

Hinweis : Nicht alle Transceiver lassen sich auf die oben genannten Modi schalten. Sie können dann das Kommando <%RadioOut=...> nutzen.

Mit diesem Macro erzeugen Sie ein Auswahlmenü für einen YAESU-Transceiver :

```
#define _Rig     YAESU-VU
#define _t_Mode          LSB,USB,CW,AM,FM,RTTY,PACKET
#macro <%MenuB="<%Table=<%RadioMode>,_t_Mode>",_t_Mode>
#if ValMenu
#macro <%RadioMode=_Rig,<%Input$>>
#endif
```

<%RadioOut=character_string>

Mit diesem Kommando erzeugen Sie eine ASCII- oder Hexa-Zeichenkette, die entsprechend des Handbuchs Ihres Transceivers an das Gerät geschickt wird und dort den gewünschten Schaltvorgang auslöst :

```
\$##...  ##=00-FF        Erzeugt ein Byte im Hexadezimal-Format
                         z.B: \$FE55AA  -> $FE,$55,$AA)
ICOM CI-V-Adressen sind anstelle von xx einzutragen :
\x##             ##=00-FF: Erzeugt ein Byte im Hexadezimal-Format
                 z.B.: \xFE\x55\xAA  -> $FE, $55, $AA)
\w##             ##=00-99, Erzeugt eine Verzögerungszeit
                 z.B.:  \w05  ->warte 50ms)
\r               Sendet einen Wagenrücklauf (CR)
\n               Sendet eine Zeilenschaltung (LF)
\c....           Kommentar
```

\\	'\' sendet eine Zeichenkette character
Others	Sendet die Zeichen wie eingegeben

Beispiele :

Setzt das VFO-A-Filter eines Yaesu FT-1000MP auf 500Hz
<%RadioOut=\$020000008C>

Vertauscht beim ICOM IC-706 den VFO A mit VFO B
<%RadioOut=\$FEFExxE007BOFD>

Setzt den VFO-B eines Kenwoodtransceivers auf 14.073MHz
<%RadioOut=FB00014073000;>

Schaltet einen Yaesu FT-847 auf CW(W)
(<%RadioMode=YAESU-VU,CW> würde den FT-847 auf CW(N) setzen)
<%RadioOut=\$0200000007>

#if IsRadioLSB

Diese Bedingung fragt den Transceiver ab, ob er auf LSB gestellt ist. Abhängig davon kann eine Frequenzkorrektur berechnet werden. MMVARI gibt entweder TRUE (wahr) oder FALSE (falsch) zurück :

Rückgabewert	Rig mode
TRUE	LSB, RTTY, PACKET
FALSE	andere Einstellungen

Bevor Sie diese Bedignung verwenden, müssen Sie die aktuelle Frequenzeinstellung des VFOs abfragen. MMVARI gibt in jedem Fall TRUE oder FALSE zurück, wenn die Sendeart mit <%RadioMode=...> eingestellt wurde..

Hinweis : Wenn die Einstellung nicht LSB oder USB ist, ist MMVARI nicht in der Lage die Seitenbandlage richtig einzustellen. Verwenden Sie <%RadioMode> um die Einstellung des Transceivers zu bestimmen.

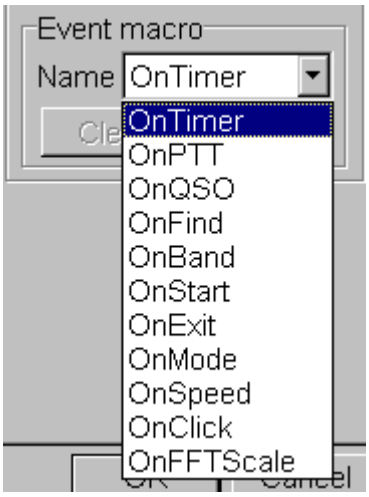
Ereignis-Macros

Das Ereignis-Macro (Event macro) wird aufgerufen, wenn eine bestimmte Bedingung erfüllt ist. Diese Ereignis-Macros stehen unter MMVARI zur Verfügung :

OnTimer:	wird jede Sekunde aufgerufen
OnPTT:Called	wird gerufen, wenn sich der PTT-Status ändert
OnQSO:	wird gerufen, wenn die QSO-Taste gedrückt wird
OnFind:	wird gerufen, wenn die Funktion HisCall gesetzt ist
OnStart:	wird gerufen, wenn MMVARI gestartet wird
OnBand:	wird gerufen, wenn im Log das Band geändert wird
OnExit:	wird gerufen, wenn MMVARI beendet wird
OnMode:	wird gerufen, wenn die Sendeart geändert wird
OnSpeed:	wird gerufen, wenn die Sendegeschwindigkeit geändert wird
OnClick:	wird gerufen, wenn in den Text im RX-Fenster geklickt wird
OnFFTScale:	Called when the FFT scale is changed

Definition eines Ereignis-Macros

Ereignis-Macros können auf der TX-Karteikarte des MMVARI-Einstellmenüs (***Options > Setup MMVARI***) definiert werden.



Wählen Sie das gewünschte Macros aus dem sich öffnenden Menü aus und drücken Sie die Taste EDIT. Wenn Sie das Macro nicht mehr ausführen wollen, löschen Sie den Macrotext.

Dieses Macro ist ein ***OnTimer event macro*** und fragt die AFC-Frequenz aller Sekunden im Wasserfall ab :

```
<%WaterMsg=4,<%AFCFrequency>Hz>
```

Das nächste Macro ist ein ***OnQSO event macro***: Es zeigt den Status im RX-Fenster an, wenn die QSO-Taste gedrückt wurde. Dieser Status wird auch im RX-Log abgespeichert.

```
#if IsQSO
#macro <%RxStatus=LogON <%HisCall> on <%BAND>/<%MODE>>
#else
#macro <%RxStatus=LogOFF <%HisCall>>
#endif
```

Mit diesem ***OnMode macro*** wird abhängig von der Sendeart (hier RTTY) das Macro-Menü umgeschaltet :

```
#if StrMode>>rtty
#macro <%SeekTop><%SeekNext>
#else
#macro <%SeekTop>
#endif
```

Definition von Ereignis-Macros über Macros

Jedes der Ereignis-Macros kann mit einer Macroanweisung definiert werden:

```
<%OnTimer=...>
<%OnPTT=...>
<%OnQSO=...>
<%OnFind=...>
<%OnBand=...>
<%OnStart=...>
<%OnExit=...>
<%OnMode=...>
<%OnSpeed=...>
<%OnClick=...>
<%OnFFTScale=...>
```

Das nächste Macro kann z.B. einer Macro-Taste zugeordnet werden und legt dann die Funktion des OnTimer-Ereignisses fest :

```
<%OnTimer=<%WaterMsg=4,<%AFCFrequency>Hz>>
```

Wenn Sie zwei und mehr Ereignis-Macros in einem Macro festlegen wollen, benutzen Sie \r\n als Begrenzung:.

```
<%OnTimer=#if ƒ@IsSQ\r\n#macro <%WaterMsg=4,<%AFCFrequency>Hz>\r\n#endif>
```

Mit der folgenden Sequenz wird ein Menü-Fenster erzeugt, mit dem der Nutzer verschiedene Funktionen des OnTimer-Macros auswählen kann:

```
#macro <%Menu=AFC, Metric(MFSK), RadioMode, WaterNoise, UTC, Local, -, Stop>
#if ValMenu == 1
#macro <%OnTimer=<%WaterMsg=4,<%AFCFrequency>Hz>>
#elseif ValMenu == 2
#macro <%OnTimer=<%WaterMsg=4,<%MetricMFSK>>>
#elseif ValMenu == 3
#macro <%OnTimer=<%WaterMsg=4,<%RadioMode>>>
#elseif ValMenu == 4

#macro <%OnTimer=<%WaterMsg=4,<%WaterNoise>dB>>
#elseif ValMenu == 5
#macro <%OnTimer=<%WaterMsg=4,<%UTIME>z>>
#elseif ValMenu == 6
#macro <%OnTimer=<%WaterMsg=4,<%LTIME>>>
#elseif ValMenu == 7
#macro <%OnTimer=>
#endif
```

Die folgende Sequenz erzeugt ein Edit-Menü für Macros. Es kann auf eine Macro-Taste gelegt werden:

```
#macro <%Menu=<%Events>>
#if ValMenu
#macro <%EditMacro=<%Input$>>
#endif
```

Prozeduren

MMVARI unterstützt einen Prozedur-Aufruf, mit dem ein wiederholter Aufruf der gleichen Operation vorgenommen werden kann. Ein Macro-Kommando kann die Prozedur so oft wie gewünscht aufrufen. Es gibt keine Begrenzung für die Anzahl der Wiederholungen. Die Prozedur kann auch in einem erweiterten Menü eingebaut werden.

Definition einer Prozedur

Die Prozedur wird zwishen #proc und #endp definiert. Sie hat dieses Format:

```
#proc  Name  Dummy1, Dummy2...
          |
#endp
```

Name ist der Name der Prozedur. Er muss mit einem Buchstaben beginnen. Dummy1, Dummy2... sind Symbole für Pseudoparameter (Platzhalter), die nur innerhalb der Prozedur wirksam werden. Sie werden ersetzt durch Argumente, die beim Aufruf der Prozedur übergeben werden. Eine Prozedur kann bis zu 64 Platzhalter haben.

Beispiel für eine Prozedur

```
#proc Slider @Title, @Command, @Min, @Max, @Step, @NumScales
<%DisableCR>
#macro <%Slider=@Title, <%@Command>, @Min, @Max, @Step, @NumScales>
#if StrMacro(<%Input$>)
```

```
<%@Command=<%Input$>>
#endif
#endp
```

Hinweis : Die Definition der Prozedur weist den Parameter keinen Wert zu. Die Parameter werden erst interpretiert, wenn die Prozedur aufgerufen wird,

Hinweis : Die Funktion der Platzhalter ist es einen Platz im Text freizuhalten. Um Irrtümer zu vermeiden, sollten die Platzhalter einen langen Namen bekommen oder ein @ vor ihrem Namen.

Hinweis : Die Definition einer Prozedur bleibt so lange effektiv, wie MMVARI läuft. Eine Prozedur, die als OnStart-Ereignis definiert wurde, kann aus Macros heraus gerufen werden, so lange MMVARI läuft. Wird eine Prozedur neu definiert, ist jeweils die neue Variante aktiv.

Aufruf einer Prozedur

Ein Macro kann eine Prozedur mit dem Kommando <%CallProc=Name, Arg1, Arg2...> aufrufen. Der Name der Prozedur wurde mit #proc definiert.
Arg1, Arg2, ... sind die Argumente, die an die Prozedur zu übergeben sind. Ist die Zahl dieser Argumente kleiner als die Zahl der Platzhalter in der Prozedur, wird dem Rest der Platzhalter der Wert NULL zugewiesen. Ist die Zahl der Argumente größer als die Zahl der Platzhalter in der Prozedur, werden die überzähligen Argumente ignoriert.

Beispiel für eine Prozedur für einen softwaredefinierten Schieberegler :

```
<%CallProc=Slider, CW speed, CWSpeed, 10, 40>
<%CallProc=Slider, Digital output level, DigitalLevel, 1024, 32768, 1024>
<%CallProc=Slider, Play back speed, PlayBackSpeed, 1, 20>
```

Hinweis : Prozeduren werden zuerst fediniert. Eine Prozedur <%CallProc=...> wird im zweiten Schritt ausgeführt. Daher kann die Definition einer Prozedur auch nach dem Macro angeordnet sein, das die Prozedur aufruft. Allerdings wird <%CallProc=...> in einem #macro zuerst ausgeführt, so dass die Prozedur hier VOR dem #macro definiert sein muss.

Hinweis : Obwohl der rekursive Aufruf von Prozeduren verboten ist, achten Sie auf den Pufferüberlauf (stack overflow), wenn Sie eine große Anzahl verschachtelter Aufrufe machen. Hier ist ein Beispiel für eine rekursiven Aufruf:.

```
<%DisableCR>
<%CallProc=Repeat, 3, CQ CQ CQ de <%MyCall> <%MyCall> <%MyCall><%CR>>
<%BS> pse k...<%CR>
#proc  Repeat  @N, @Text
<%DisableCR>
#define  _RepCount  <%Format=%d,@N-1>
#if _RepCount >= 0
@Text
<%CallProc=Repeat, _RepCount, @Text>
#endif
#endp
```

Hinweis : Verwenden Sie <%DebugProc=...> anstelle von <%CallProc=...>, um den Fehler im TX-Fenster zu beheben. <%DebugProc=...> zeigt an, wie die Prozedur ausgeführt wird.

Steuerprogramm für ein erweitertes Menü

Eine Prozedur kann als Steuerprogramm (handler) für ein erweitertes Menü eingesetzt werden. Mit dem erweiterten Menü kann der Nutzer ein MMVARI-Menü seinen Wünschen anpassen. <%AddMenu=...> und <%InsertMenu=...>erzeugen ein erweitertes Menü:

```
<%AddMenu=Name, Caption, Procedure, Arg1, Arg2...>
```

```
<%InsertMenu=Name, InsPos, Caption, Procedure, Arg1, Arg2...>
```

Name:	Name des Menüs mit Zugriffstaste (&x) oder Index (1...)
InsPos:	Überschrift für den Zugriffspunkt (&x) oder Index (1...)
Caption:	Überschrift des Menüs, Zugriffstaste (&x) oder Index (1...)
Procedure:	Name des Steuerprogramms (handler procedure)
Arg:	Argumente für das Steuerprogramm (können entfallen)

Ein einfaches Beispiel für ein erweitertes Menü wird nachfolgenden angezeigt. Die Macro-Kommandos werden üblicherweise durch OnStart-Ereignis-Macro definiert.

```
<%DisableCR>
#define _Name E&xtension
<%AddMenu=_Name, &CW speed..., Slider, CW speed, CWSpeed, 10, 40>
<%AddMenu=_Name, &Digital output level..., Slider, Digital output level,
                DigitalLevel, 1024, 32768, 1024>
<%AddMenu=_Name, ->
<%AddMenu=_Name, CQ DX(&1), OnCQDXClick, 1, 3, 3, 4000>
<%AddMenu=_Name, CQ DX(&3), OnCQDXClick, 3, 3, 3, 5000>

#proc   OnCQDXClick  @Nline, @Ncq, @Ncall, @Interval
<%DisableCR><%ClearTXW><%AutoClear><%TX><%RX>
<%RepeatText=@Nline,<%RepeatText=@Ncq,
                CQ DX<%SP>>de<%RepeatText=@Ncall,<%SP><%MyCall>><%CR>>
<%BS><%SP>pse DX k<%CR><%RepeatTX=@Interval>
#endp

#proc Slider @Title, @Command, @Min, @Max, @Step, @NumScales
<%DisableCR>
#macro <%Slider=@Title, <%@Command>, @Min, @Max, @Step, @NumScales>
#if StrMacro(<%Input$>)
<%@Command=<%Input$>>
#endif
#endp
```

Hinweis : Wenn Sie auf den Titel des erweiterten Menüs klicken, wird das Steuerprogramm automatisch aufgerufen:.

```
On$xxxClick     xxx = Name des Menüs  (z.B.: On$E&xtensionClick)
```

Hinweis : Wenn Sie eine Zugrifftaste definiert haben, können Sie alle Menükommandos einschliesslich <%DoMenu=...>, <%DeleteMenu=...>, <%AddMenu=...> damit rufen. Achten Sie darauf, dass Sie nicht eine Zugriffstaste für zwei und mehr Kommandos definiert haben.

<%AddMenu=...> und <%InsertMenu=...> können dazu benutzt werden, zu einem vordefinierten Menü weitere Punkte hinzuzufügen oder vorhandene Punkte durch andere zu ersetzen. Hier ist ein Beispiel:

```
<%DisableCR>
#if !IsDefined(_fShellHelp)
#define  _fShellHelp  0
#endif
<%AddMenu=&E, ->
<%AddMenu=&E, Edit &AS(CW) macro..., OnCommand, <%EditMacro=AS(CW)>>
<%AddMenu=&E, Edit &OnStart event..., OnCommand, <%OnStart>>
<%InsertMenu=&O, &B, &Digital output level..., Slider, Digital output level,
                DigitalLevel, 1024, 32768, 1024>
<%InsertMenu=&O, &B, ->
<%AddMenu=&H, &P, OnShellEdit, project.txt, e, 1>

<%AddMenu=&H, &O, OnShellEdit, mmvari.txt, e, 1>
<%AddMenu=&H, &S, OnShellEdit, Samples.txt, , 3>
<%AddMenu=&H, &H, OnShellEdit, history.txt, e, 1>
```

```
<%InsertMenu=&H, &D, &Use Shell's standard editor, InvRegVal, _fShellHelp>
<%InsertMenu=&H, &D, ->

#proc On$&HelpClick
<%DisableCR><%CheckMenu=&H, &U, _fShellHelp>
#endp

#proc OnCommand  @Command
<%DisableCR>@Command
#endp

#proc  OnInvVal  @Value
<%DisableCR>
#DEFINE         @Value          <%Inv=@Value>
#endp

#proc  OnShellEdit  @File, @Prifix, @Flag
<%DisableCR>
#if IsEnglish
#define  _FileName  <%Folder>@Prifix@File
#else
#define  _FileName  <%Folder>@File
#endif
#if _fShellHelp
<%Shell=_FileName>
#else
<%EditFile=_FileName, @Flag>
#endif
#endp

#proc Slider @Title, @Command, @Min, @Max, @Step, @NumScales
<%DisableCR>
#macro <%Slider=@Title, <%@Command>, @Min, @Max, @Step, @NumScales>
#if StrMacro(<%Input$>)
<%@Command=<%Input$>>
#endif
#endp
```

Hinweis : Ein Menüpunkt, der weitere Unterpunkte hat, kann nicht überschrieben werden.
Hinweis : Wenn Sie einen neuen Menüpunkt einführen, ändert sich die Indexnummer. Nehmen Sie lieber eine Zugriffstaste, um Irrtümer zu vermeiden.
Hinweis: Der Klick auf einen Titel ein definiertes Menüs ruft das folgende Steuerprogramm. Dieses Steuerprogramm ist optional.

```
On$&FileClick, On$&EditClick, On$&ViewClick, On$&OptionsClick, On$&HelpClick
```

Wiederholungsblock

Der Wiederholblock (Repeat block) wird zwischen den Kommandos #repeat und #endp definiert. Der Block wird N-mal wiederholt. N wird unmittelbar nach #repeat definiert. Ist N=0 wird der Block nicht ausgeführt.ted.

```
#repeat statement
          |
#endp
```

In diesem Wiederholblock sind die Wiederholziffer ***$repeat*** und der Zähler ***$counter*** intern definiert:

```
#repeat  3
CQ CQ CQ de <%MyCall> <%MyCall> <%MyCall> (<%Format=%d, 1 + $repeat - $counter>)
#endp
```

Hinweis : Wird der Wiederholblock verschachtelt, hat jede Ebene ihren eigenen unabhängigen $repeat und $counter.

```
#repeat  3
<%DisableCR>Outside=($counter/$repeat)[
#repeat  2
<%DisableCR>
#if $counter > 1
,
#endif
Inside=($counter/$repeat)
#endp
]<%CR>
#endp
```

Speicherwiedergabe (Sound playback)

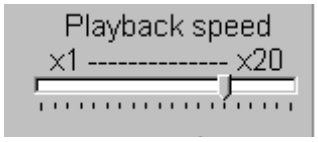
Mit der Speicherwiedergabe können Sie die letzten 15/30/60-Sekunden, der durch den Wasserfall von MMVARI gelaufenen Signale erneut abrufen. Klicken Sie auf eine der Tasten 60, 30 oder 15.

60	Wiederholt die letzten 60 Sekunden
30	Wiederholt die letzten 30 Sekunden
15	Wiederholt die letzten 15 Sekunden

Diese Funktion ist nützlich in diesen Gelegenheiten :

- Jemand hat mich gerufen, ich habe aber den Cursor nicht genau auf das Signal gesetzt
- Jemand hat mich gerufen, ich hatte aber die falsche Sendeart eingeschaltet

Sie können auch Signale im Wasserfall wie z.B. MFSK-Signale, die länger zum Einrasten brauchen, öfter wiederholen und die Einstellung optimieren.

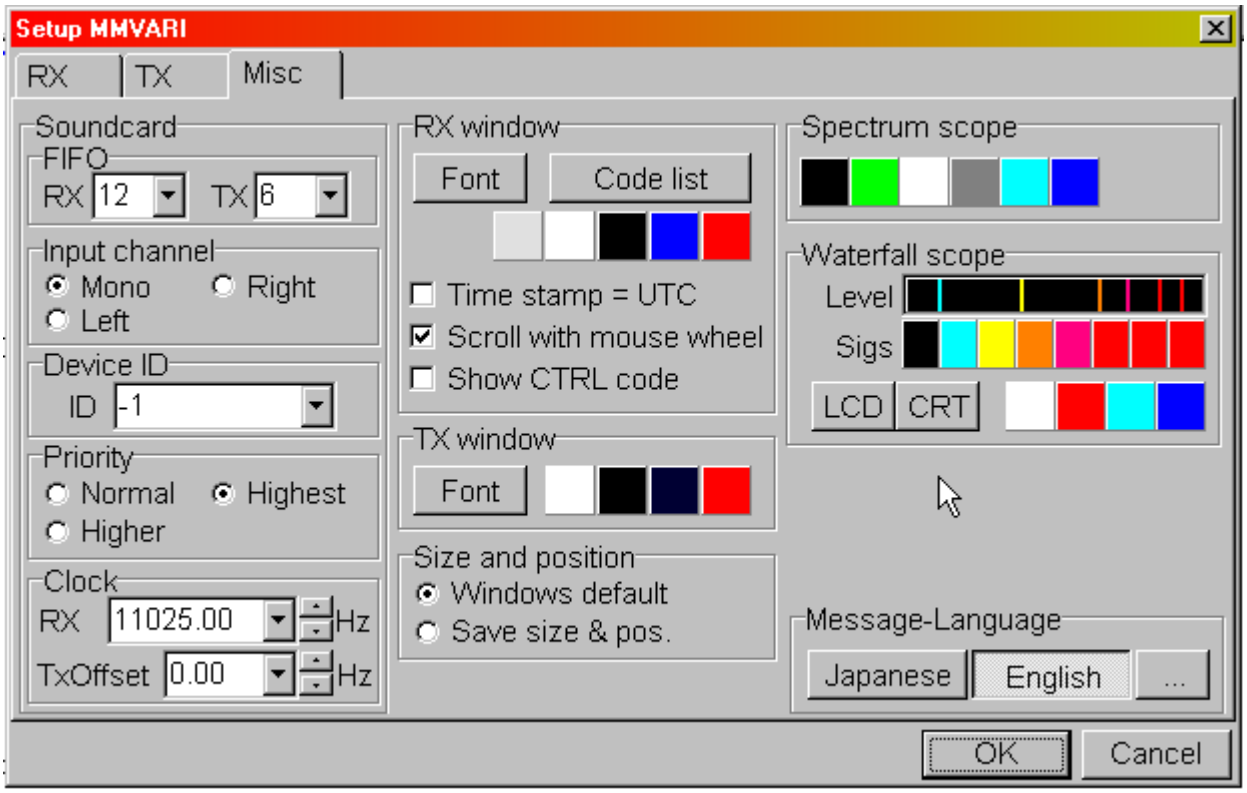


Mit dem Schieberegler rechts neben den Wiederholtasten, können Sie die Geschwindigkeit der Wiedergabe zwischen x1 und x20 einstellen. Voreingestellt ist x5, d.h. eine 15-Sekunden-Aufzeichnung wird in 3 Sekunden abgespielt und dekodiert. Die Geschwindigkeit kann durch Ihren CPU-Takt begrenzt werden.
(*Hinweis DM3ML* : Dieser Schieberegler ist in Version 0.42 nicht mehr im Hauptfenster, sondern auf der Karteikarte RX (Options > Setup MMVARI) zu finden:

MMVARI speichert konstant die letzten 60 Sekunden des Wasserfalls im PCM-Format. Die Abtastfrequenz liegt bei 11025Hz und es werden etwa 1.3MB an RAM benötigt.

Soundkarteneinstellungen

Die Soundkarteneigenschaften werden auf der Karteikarte MISC im MMVARI-Einstellmenü gewählt :



- Fifo – RX (Empfangspufferspeicher)
  Erhöhen Sie die Tiefe des Fifo-Speichers, wenn der Empfangston stottert.

- Fifo – TX (Sendepufferspeicher)
  Erhöhen Sie die Tiefe des Fifo-Speichers, wenn der Sendeton zerhackt wird. Je tiefer der Fifo-Speicher eingestellt wird, um so länger ist die Zeit zwischen der Eingabe von Text über die Tastatur und die Textausgabe über den Sender.

- DeviceID (Soundkarten-Nummer)
  Falls Ihr PC über mehr als eine Soundkarte verfügt, stellen Sie hier die Nummer der Soundkarte ein. Die Zählung beginnt mit 0. Die Karte mit der Nummer -1 ist die unter Windows voreingestellte Soundkarte.

MMVARI unterstützt die Soundkartenliste, die ursprünglich für MMTTY/MMSSTV entwickelt wurde. Stellen Sie sicher, das die mmw-Datei unter MMVARI installiert ist. Mit der mmw-Datei können Sie sich die passende Soundkarte heraussuchen.
Weitere Informationen finden Sie in dem Paket MMW.zip in den Texten EReadme.txt und EMMW.txt.

Hinweis DM3ML : Bekommen Sie bei mehreren Soundkarten im PC die Einstellungen für den RX- und den TX-Kanal nicht wie gewünscht hin, definieren Sie Ihre MMVARI-Soundkarte (z.B. das USB-Codec) als Windows-Standardkarte für Aufzeichnung und Wiedergabe und nehmen Sie die Einstellung ‚-1' bei MMVARI.

Programmpriorität (priority)

Falls Ihr TX-Sound stottert, erhöhen Sie die Priorität von MMVARI gegenüber anderen Programmen.

Eingabekanal (Input channel)

Wählen Sie für den RX den Soundkarteneingang zwischen links, rechts oder Stereo. Auf der Sendeseite wird jeweils der linke UND der rechte Kanal mit dem Sendesignal belegt.

Empfangstakt (Clock – RX)

Hier legen Sie die Abtastfrequenz fest. Voreingestellt ist 11025 Hz. MMVARI unterstützt diese Abtastraten :

1 - 11025Hz: Typischer Soundkartentakt, von allen Soundkarten am Markt unterstützt
2 - 12000Hz: Unüblicher Soundkartentakt mit TX-Offset. Von den meisten Karten unterstützt.
3 - 6000Hz: Verwendet bei langsamen CPUs. Wird von den einigen Karten unterstützt.

8000Hz	mitunter verwendbar
18000Hz	mitunter verwendbar.
22050Hz	Standardfrequenz, mitunter verwendbar
24000Hz	mitunter verwendbar
44100Hz	Standardfrequenz, mitunter verwendbar
48000Hz	Für eine optische Übertragung (S/PDIF).

Hinweis : Enthalten in der Liste ist auch die Frequenz 11100Hz, das ist eine Variante der 11025 Hz-Einstellunng.

Hinweis : Mit 8000Hz laufen das Spektrum und der Wasserfall etwas langsamer als mit den anderen Takten.

Taktunterschied TX (Clock – TxOffset)
Legt die Abweichung des TX-Taktes gegenüber dem RX-Takt fest, soweit die Soundkarte diese Einstellung gestattet. Die Berechnung erfolgt nach der Formel :

TX sampling frequency = RX sampling frequency + TxOffset [Hz]

Taktabgleich der Soundkarte

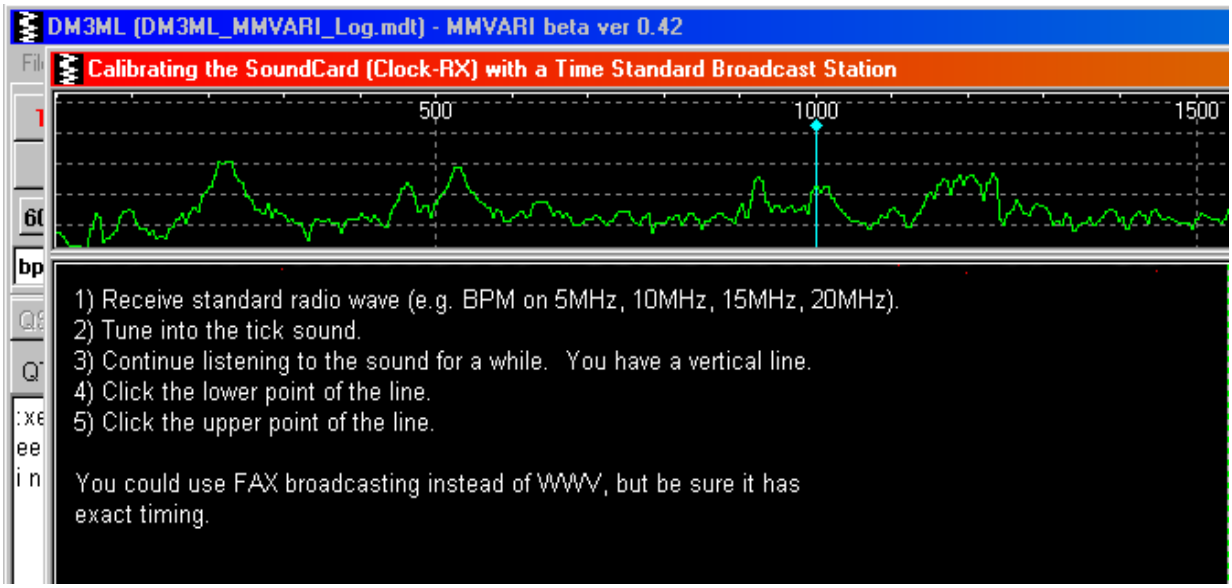
Die Taktfrequenz der Soundkarte ist bei MMVARI nicht so kritisch wie bei SSTV. Falls aber die RX- und die TX-Frequenz deutlich voneinander abweichen, kann es zu Problemen bei der Dekodierung und Synchronisation kommen.

- Abgleich mit MMSSTV

Sie können für MMVARI die unter MMSSTV bestimmten Werte verwenden. Der Abgleich mit MMSSTV ist der einfachste Weg. Stellen Sie sicher, das die automatische Schräglaufkorrektur bei MMSSTV abgeschaltet ist.

- RX-Frequenz zuerst abgleichen

Gegen Sie ins Options-Menu und wählen Sie ***Calibrating the SoundCard***. Es öffnet sich ein weiteres Fenster. Folgenden Sie den hier gegebenen Instruktionen.



Der Abgleich verwendet den Sender BPM, der in Ostasien gut zu empfangen ist. (Hinweis : In DL eignet sich der Sender RPM auf 9996 kHz). Sie können mit dieser Prozedur nur den Empfangstakt abgleichen. Den TX-Takt gleichen Sie im folgenden Schritt ab.

- TXOffset abgleichen

Gehen Sie zur Karteikarte TX, wählen Sie den Modus ***External (full-duplex) loop-back*** und verbinden Sie den Eingang und den Ausgang der Soundkarte miteinander. Sie müssen nicht unbedingt einen realen Draht einbauen, oft können Sie diese Verbindung auch über den Windows-Mixer herstellen, in dem Sie den Mixer als ***Line input*** angeben.

Stellen Sie im MMVARI-Hauptfenster diese Optionen ein :

- NET off, AFC on und ATC on.
- Löschen Sie das TX-Vorschreibfenster und erzeugen Sie ein Leerlaufsignal
- Klicken Sie auf die Sendetaste TX. Stellen Sie sicher, dass der Transceiver ausgeschaltet ist.
- Klicken Sie im Spektrum aif das RX-Signal. Stellen Sie die Differenz zwischen TX- und RX-Signal fest.

Nach einer Weile, wird die ATC-Anzeige stabil. Setzen Sie den Cursor in das Fenster der ATC-Zeitanzeige und beachten Sie den Wert für den RXOffset in der Statusleiste. Ziehen Sie den RXOffset vom Wert des TXOffset ab.

NeuerTxOffset = Aktueller TxOffset - RxOffset [Hz]

Ist z.B. der aktuelle TXOffset=0 und für den RXOffset wird -74.40 angezeigt, ist der neue TxOffset=74.40.

Das wars! Stellen Sie den ***loop-back*** zurücl von ***Externa***l auf ***Interna***l.

- **Eine anderer einfacher TxOffset-Abgleich**

Tragen Sie 0 als TxOffset ein und senden Sie das Leerlaufsignal für eine Weile. Lassen Sie sich den ATC-Wert von einer empfangenden Station geben. Tragen Sie diesen Wert mit invertiertem Vorzeichen als TxOffset ein. Voraussetzung ist, dass die empfangende Station korrekt abgeglichen worden ist.

Namenkonventionen bei MMVARI

MMVARI definiert die Namen der Varianten von VARICODE wie folgt : as follows. The automatically refers to the definition.

VariSTD	Standard PSK31 VARICODE (256 codes)
VariJA	Japanese VARICODE (12160 codes)
VariHL	Hangul VARICODE (24448 codes)
VariBV	Chinese (BV) VARICODE (24448 codes)
VariBY	Chinese (BY) VARICODE (24448 codes)

Das Macro <%VARITYPE> bezieht sich auf diese Definitionen. Die konventionelle Methodfe, die den Standard VARICODE verwendet, sendet zwei unübersetzte Kodes für ein MBCS-Zeichen ist definiert als "VariSTD/JA" oder "VariSTD/HL" abhängig von der am PC eingestellten Spracheinstellung.

Diese Tabelle ergibt Modulation und <%VARITYPE>-Kombination :

GMSK	VariJA, VariHL, VariBV, VariBY
FSK	VariJA, VariHL, VariBV, VariBY
BPSK	VariJA, VariHL, VariBV, VariBY

bpsk VariSTD/JA, VariSTD/HL, VariSTD/BV, VariSTD/BY

Hinweis : Wenn eine SBCS-Sprache ausgewählt wurde (Sendeart beginnt mit einem Kleinbuchstaben), wird in jedem Fall VariSTD verwendet.

Hinweis : In bpsk ist "VariSTD/JA" unüblich, so dass "Japanese" besser ist. Das folgende Macro-Beispiel kann für diesen Zweck verwendet werden :

```
RRR <%HisCall> de <%MyCall>
#if IsLocal
#if StrVARITYPE == STD/JA
--- Japanese ---
#else
--- <%VARITYPE> ---
#endif
#endif
```

73, Mako JE3HHT