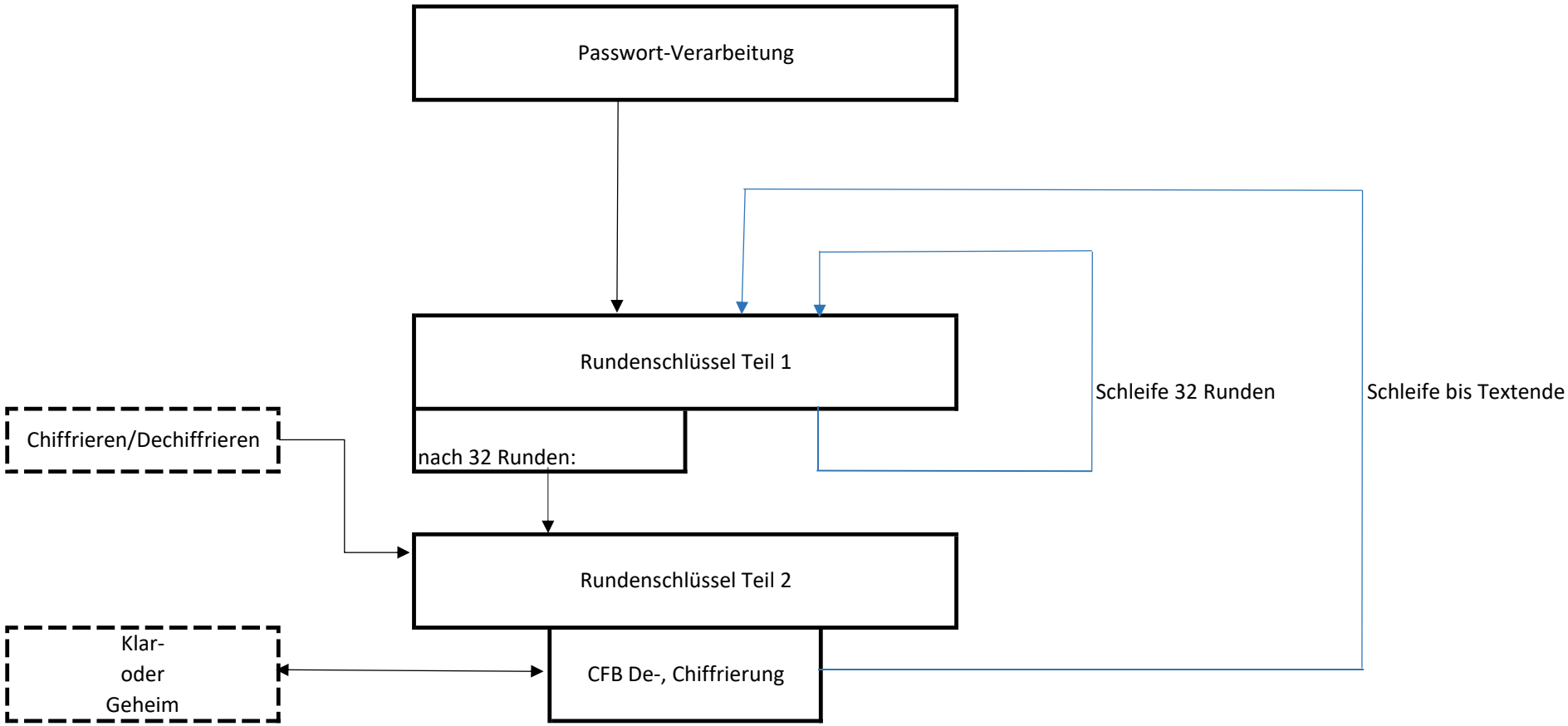
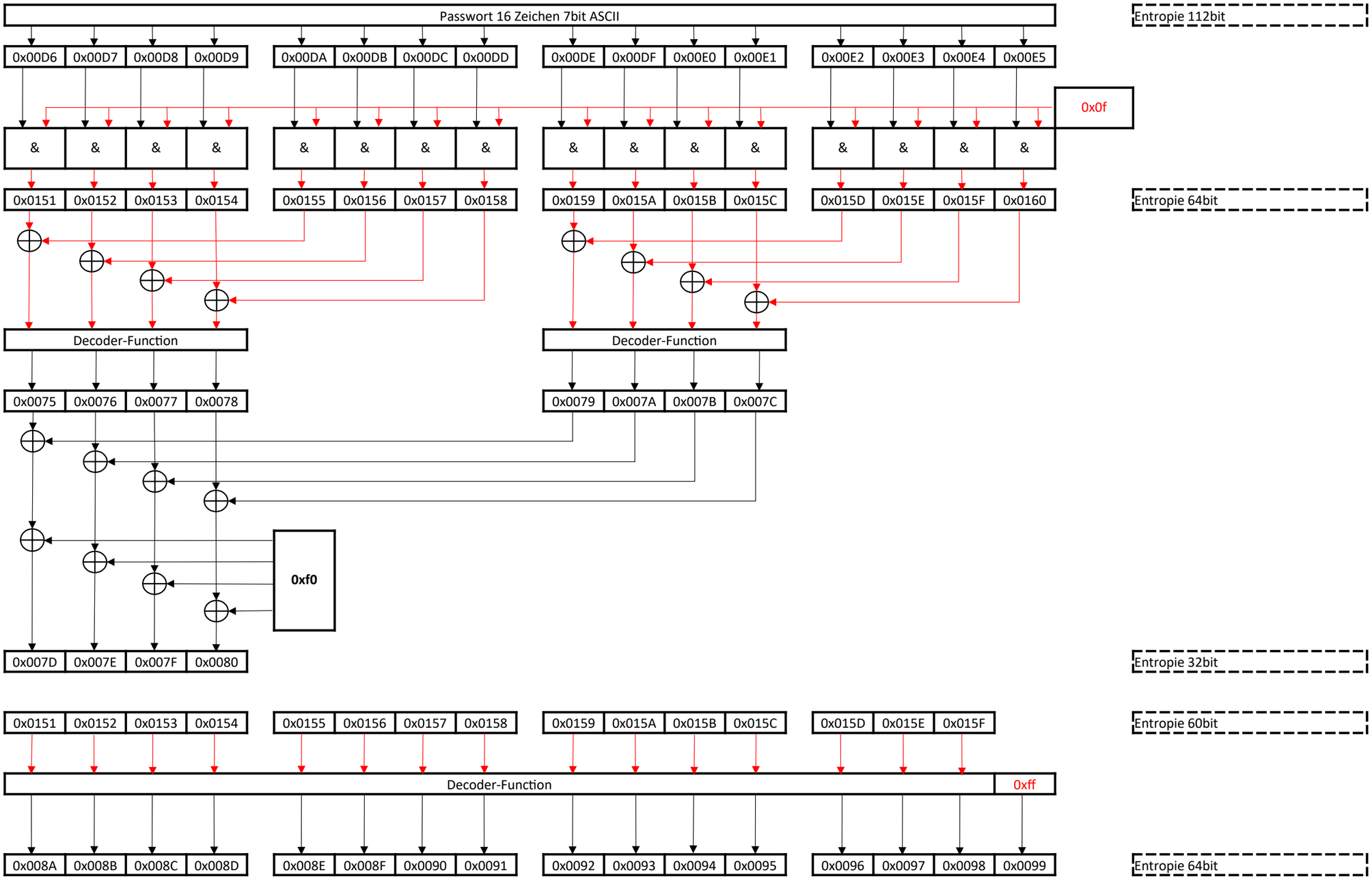
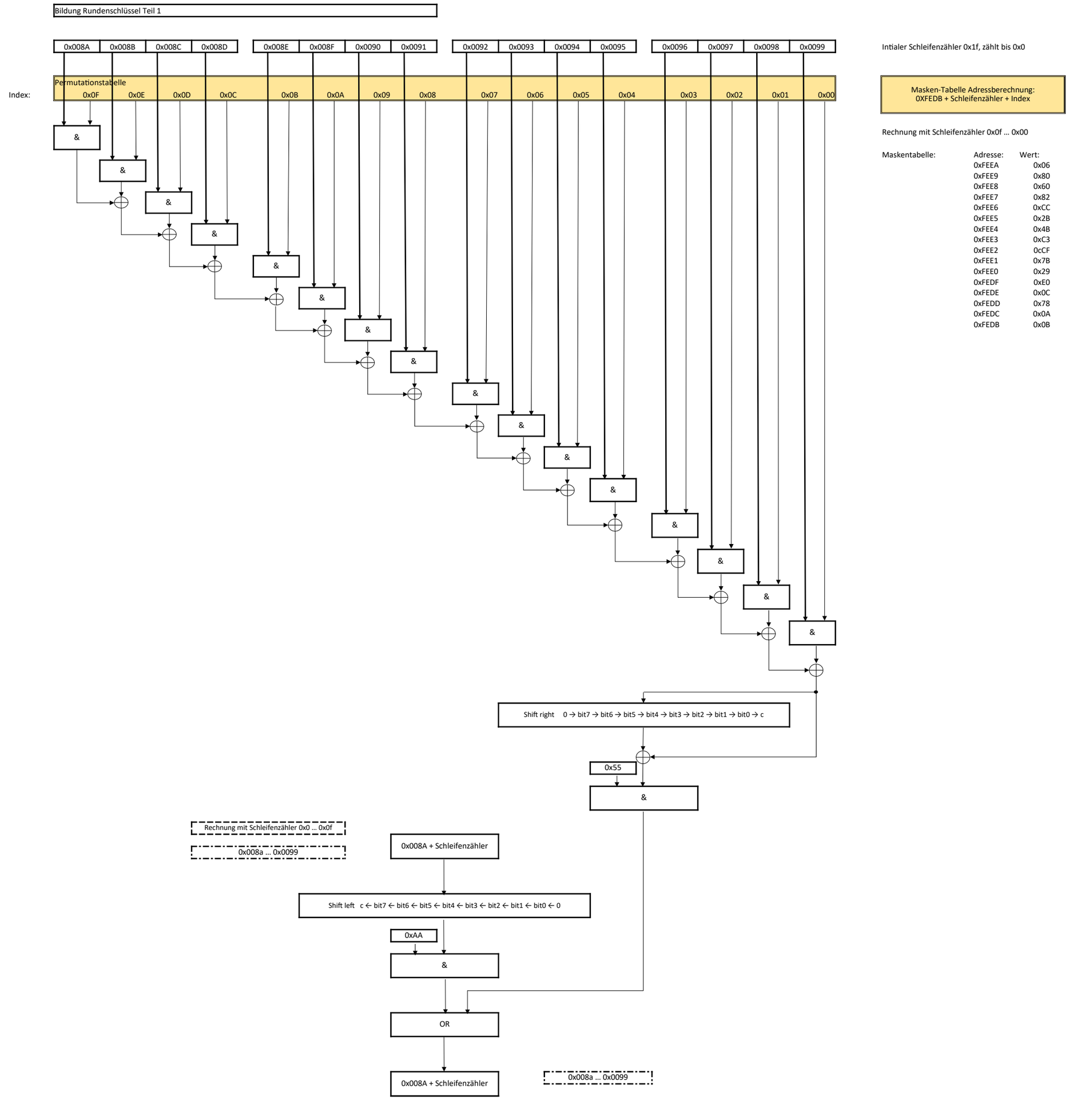

Passwort-Verarbeitung
Rundenschlüssel Teil 1
Schleife 32 Runden
Schleife bis Textende
Chiffrieren/Dechiffrieren
nach 32 Runden:
Rundenschlüssel Teil 2
Klar-
oder
Geheim
CFB De-, Chiffrierung


Passwort 16 Zeichen 7bit ASCII
Entropie 112bit
0x00D6 0x00D7 0x00D8 0x00D9
0x00DA 0x00DB 0x00DC 0x00DD
0x00DE 0x00DF 0x00E0 0x00E1
0x00E2 0x00E3 0x00E4 0x00E5
0x0f
&
0x0151 0x0152 0x0153 0x0154
0x0155 0x0156 0x0157 0x0158
0x0159 0x015A 0x015B 0x015C
0x015D 0x015E 0x015F 0x0160
Entropie 64bit
Decoder-Function
Decoder-Function
0x0075 0x0076 0x0077 0x0078
0x0079 0x007A 0x007B 0x007C
0xf0
0x007D 0x007E 0x007F 0x0080
Entropie 32bit
0x0151 0x0152 0x0153 0x0154
0x0155 0x0156 0x0157 0x0158
0x0159 0x015A 0x015B 0x015C
0x015D 0x015E 0x015F
Entropie 60bit
Decoder-Function
0xff
0x008A 0x008B 0x008C 0x008D
0x008E 0x008F 0x0090 0x0091
0x0092 0x0093 0x0094 0x0095
0x0096 0x0097 0x0098 0x0099
Entropie 64bit
Decoder-Function
Hinweis:
Adressregister sind vierstellig,
Variablen und Konstanten sind
zweistellig dargestellt
Zeichenerklärung:
4bit
8bit
Programmschritt

	Input																Output
Relevanter Arbeitsbereich	0	10	20	30	40	50	60	70	80	90	a0	b0	c0	d0	e0	f0	f0
	1	21	41	61	81	a1	c1	e1									e1
	2	12	42	52	82	92	c2	d2									d2
	3	43	83	c3													c3
	4	14	24	34	84	94	a4	b4									b4
	5	25	85	a5													a5
	6	16	86	96													96
	7	87															87
	8	18	28	38	48	58	68	78									78
	9	29	49	69													69
	0a	1a	4a	5a													5a
	0b	4b															4b
	0c	1c	2c	3c													3c
	0d	2d															2d
	0e	1e															1e
	0f																0f
Nichtrelevanter Arbeitsbereich	11	31	51	71	91	b1	d1	f1									f1
	13	53	93	d3													d3
	15	35	95	b5													b5
	17	97															97
	19	39	59	79													79
	1b	5b															5b
	1d	3d															3d
	1f																1f
	22	32	62	72	a2	b2	e2	f2									f2
	23	63	a3	e3													e3
	26	36	a6	b6													b6
	27	a7															a7
	2a	3a	6a	7a													7a
	2b	6b															6b
	2e	3e															3e
	2f																2f
	33	73	b3	f3													f3
	37	b7															b7
	3b	7b															7b
	3f																3f
	44	54	64	74	c4	d4	e4	f4									f4
	45	65	c5	e5													e5
	46	56	c6	d6													d6
	47	c7															c7
	4c	5c	6c	7c													7c
	4d	6d															6d
	4e	5e															5e
	4f																4f
	55	75	d5	f5													f5
	57	d7															d7
	5d	7d															7d
	5f																5f
	66	76	e6	f6													f6
	67	e7															e7
	6e	7e															7e
	6f																6f
	77	f7															f7
	7f																7f
	88	98	a8	b8	c8	d8	e8	f8									f8
	89	a9	c9	e9													e9
	8a	9a	ca	da													da
	8b	cb															cb
	8c	9c	ac	bc													bc
	8d	ad															ad
	8e	9e															9e
	8f																8f
	99	b9	d9	f9													f9
	9b	db															db
	9d	bd															bd
	9f																9f
	aa	ba	ea	fa													fa
	ab	eb	cd														eb
	ae	be															be
	af																af
	bb	fb															fb
	bf																bf
	cc	dc	ec	fc													fc
	ce	de															de
	cf																cf
	dd	fd															fd
	df																df
	ed																ed
	ee	fe															fe
	ef																ef
	ff																ff


Bildung Rundenschlüssel Teil 1
0x008A
0x008B
0x008C
0x008D
0x008E
0x008F
0x0090
0x0091
0x0092
0x0093
0x0094
0x0095
0x0096
0x0097
0x0098
0x0099
Intialer Schleifenzähler 0x1f, zählt bis 0x0
Permutationstabelle
Index:
0x0F
0x0E
0x0D
0x0C
0x0B
0x0A
0x09
0x08
0x07
0x06
0x05
0x04
0x03
0x02
0x01
0x00
Masken-Tabelle Adressberechnung:
0XFEDB + Schleifenzähler + Index
Rechnung mit Schleifenzähler 0x0f ... 0x00
Maskentabelle:
Adresse:
Wert:
0xFEEA 0x06
0xFEE9 0x80
0xFEE8 0x60
0xFEE7 0x82
0xFEE6 0xCC
0xFEE5 0x2B
0xFEE4 0x4B
0xFEE3 0xC3
0xFEE2 0cCF
0xFEE1 0x7B
0xFEE0 0x29
0xFEDF 0xE0
0xFEDE 0x0C
0xFEDD 0x78
0xFEDC 0x0A
0xFEDB 0x0B
&
Shift right 0 → bit7 → bit6 → bit5 → bit4 → bit3 → bit2 → bit1 → bit0 → c
0x55
Rechnung mit Schleifenzähler 0x0 ... 0x0f
0x008a ... 0x0099
0x008A + Schleifenzähler
Shift left c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0
0xAA
OR
0x008A + Schleifenzähler
0x008a ... 0x0099

Bildung Rundenschlüssel Teil 2 und De-, Chiffrierung CFB

0x0094 → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → ⊕ 0x009D → 0x0085

0x0093 → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → ⊕ 0x009C → 0x0084

0x0092 → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → ⊕ 0x009B → 0x0083

0x0091 → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← c → ⊕ 0x009A → 0x0082

0x0078 → ⊕ 0x0080 → & 0x0f

0x0077 → ⊕ 0x007F → & 0x0f

0x0076 → ⊕ 0x007E → & 0x0f

0x0075 → ⊕ 0x007D → & 0x0f

Shift right 0 → bit7 → bit6 → bit5 → bit4 → bit3 → bit2 → bit1 → bit0 → c (×4)

0xFE8A + Byte aus Schieberegister → Permutationstabelle S-Box 0XFE8A ... 0xFE99 → Permutationstabelle S-Box 0XFE8A ... 0xFE99

0xFE7A + Byte aus Schieberegister → Permutationstabelle S-Box 0XFE7A ... 0xFE89 → Permutationstabelle S-Box 0XFE7A ... 0xFE89

0xFE6A + Byte aus Schieberegister → Permutationstabelle S-Box 0XFE6A ... 0xFE79 → Permutationstabelle S-Box 0XFE6A ... 0xFE79

0xFE5A + Byte aus Schieberegister → Permutationstabelle S-Box 0XFE5A ... 0xFE69 → Permutationstabelle S-Box 0XFE5A ... 0xFE69

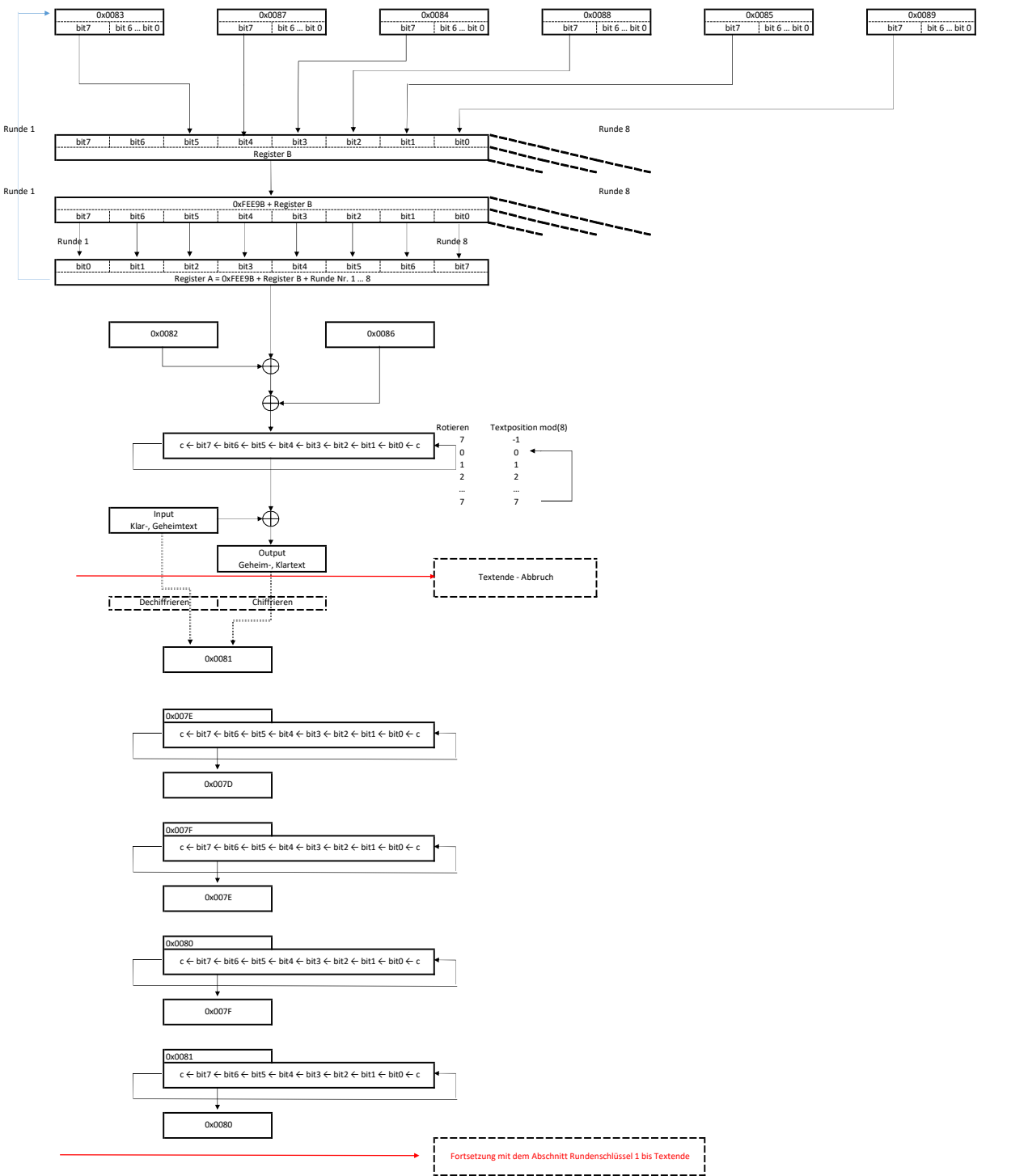
OR 0x007C → 0x0089

OR 0x007B → 0x0088

OR 0x007A → 0x0087

OR 0x0079 → 0x0086

Nächste Abfolge



c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0083
c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0084
c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0085
c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0087
c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0088
c ← bit7 ← bit6 ← bit5 ← bit4 ← bit3 ← bit2 ← bit1 ← bit0 ← 0	0x0089

```
*******************************************************************************
* Programm-Analyse für Schulungs- und wissenschaftliche Zwecke  !              *
* Keine Prüfmöglichkeiten auf evtl. noch existierende Copyright !              *
*------------------------------------------------------------------------------*
* f9dasm: M6800/1/2/3/8/9 / H6309 Binary/OS9/FLEX9 Disassembler V1.78          *
* Loaded binary file PX1000_EPROM_bin                                          *
* Bearbeitet mit ULTRAEDIT                                                     *
* Assembler geprüft mit THRSim11 68HC11 Simulator (C) Harry Broeders           *
* Kommentare in Z80 oder C oder mit Beispielwerten                             *
*******************************************************************************

************************************************
** Used Labels                                 *
************************************************

M0000   EQU     $0000 /* ioPort1dataDirection DDR
M0001   EQU     $0001 /* ioPort2dataDirection DDR
M0002   EQU     $0002 /* ioPort1data          DATA
M0003   EQU     $0003 /* ioPort2data          DATA - Port Empfange Daten
M0008   EQU     $0008 /* timerControl and Status Register; bit7 - 5 Status ICF, OCF1, TOF; Bit 4 - 0 R/W EICI, EOCI1, ETOI, IEDG, OLVL1
M0009   EQU     $0009 /* Counter High-byte
M000A   EQU     $000A /* Counter Low-byte
M000B   EQU     $000B /* OutputCompare High-byte
M000C   EQU     $000C /* OutputCompare Low-byte
M000D   EQU     $000D /* InputCapture High-byte
M000E   EQU     $000E /* InputCapture Low-byte
M0010   EQU     $0010 /* SerialRateAndModeControlRegister
M0011   EQU     $0011 /* SerialControlAndStatusRegister
M0012   EQU     $0012 /* 0x0012 SerialReceiverDataRegister
M0013   EQU     $0013 /* 0x0013 SerialTransmitDataRegister
M0014   EQU     $0014 /* RWMemoryControlRegister
M0020   EQU     $0020 /* Variable
M0023   EQU     $0023 /* Initalwert 0x0287 Text-Start-Ende-adresse
M0025   EQU     $0025 /* Initalwert 0x0287 Zeiger Text-Ende
M0027   EQU     $0027 /* Initalwert 0x0287 Zeiger Text-aktual Position
M0029   EQU     $0029 /* Initalwert 0x0287
M002B   EQU     $002B /* Initalwert 0x0287 Zeiger Text-aktual Position
M002D   EQU     $002D /* Initalwert 0x0287 Zeiger Text-Start
M002F   EQU     $002F /* Textadresse 0x03nn ... 0x63nn
M0030   EQU     $0030 /* Variable
M0031   EQU     $0031 /* Variable
M0032   EQU     $0032 /* Variable
M0033   EQU     $0033 /* Variable
M0034   EQU     $0034 /* Variable
M0035   EQU     $0035 /* Tastatur z/s
M0036   EQU     $0036 /* Tastatur z/s
M0037   EQU     $0037 /* Variable
M0038   EQU     $0038 /* Variable
M0039   EQU     $0039 /* Variable
M003A   EQU     $003A /* Variable
M003B   EQU     $003B /* Variable
M003C   EQU     $003C /* Variable
M003D   EQU     $003D /* Variable
        EQU     $0040 /* Schleifenzähler
M0044   EQU     $0044 /* Variable
M0045   EQU     $0045 /* Variable
        EQU     $0046 /* Schleifenzähler
M004D   EQU     $004D /* Variable
M004E   EQU     $004E /* Variable  ... 0x0056, 8 ASCII Zeichen
M004F   EQU     $004F /* Variable
M0055   EQU     $0055 /* Variable
M0056   EQU     $0056 /* Variable
M0057   EQU     $0057 /* Variable
M0058   EQU     $0058 -> Zeiger auf 0x02fb
Temporäre Variable Wort-Länge
M005A   EQU     $005A -> Zeitschleifen, Baudraten und sonstiges: 0xabad, 0x4b75, 0x9b9b; 0x9c9d, 0x0341,
M005B   EQU     $905B ->
M005C   EQU     $005C /* Variable
M005D   EQU     $005D /* Schleifen-Variable
M005E   EQU     $005E /* Variable
M005F   EQU     $005F /* Schleifen-Variable
M0060   EQU     $0060 /* Variable 0x0126 oder 0x0087
M0062   EQU     $0062 /* Variable 0x008 oder 0x0001
        EQU     $0064  Konstante 100
M0066   EQU     $0066 /* Variable
M006C   EQU     $006C /* Variable Ablageort SP
M006E   EQU     $006E /* Variable
M0070   EQU     $0070 /* Zeiger-Variable
M0071   EQU     $0071 /* Variable, Lowteil Zeiger 0x0070
M0072   EQU     $0072 /* Variable
M0073   EQU     $0073 /* Zeiger-Variable  -> 0x00d8, 0x0288
        EQU     $0074 /* Schleifen-Variable Startadresse
M0075   EQU     $0075 /* Schlüsselspeicher aus der Zusammenfassung K0-3 xor K4-7
M0076   EQU     $0076 /* Schlüsselspeicher aus der Zusammenfassung K0-3 xor K4-7
M0077   EQU     $0077 /* Schlüsselspeicher aus der Zusammenfassung K0-3 xor K4-7
M0078   EQU     $0078 /* Schlüsselspeicher aus der Zusammenfassung K0-3 xor K4-7
M0079   EQU     $0079 /* Schlüsselspeicher aus der Zusammenfassung K8-11 xor K12-15
M007A   EQU     $007A /* Schlüsselspeicher aus der Zusammenfassung K8-11 xor K12-15
M007B   EQU     $007B /* Schlüsselspeicher aus der Zusammenfassung K8-11 xor K12-15
M007C   EQU     $007C /* Schlüsselspeicher aus der Zusammenfassung K8-11 xor K12-15
M007D   EQU     $007D /* PRNG - 0x80              Teilschlüssel aus K0-K15
M0081   EQU     $0081 /* PRNG - 0x7e              Teilschlüssel aus K0-K15
M0082   EQU     $0082 /* Ergo - XOR               Teilschlüssel aus K0-K15
M0083   EQU     $0083 /* Ergo - XOR               Teilschlüssel aus K0-K15
M0085   EQU     $0085 /* Ergo - XOR               Teilschlüssel aus K0-K15
```

```
M0086   EQU     $0086 /* KeySpeicher              Schlüssel zum Chiffrieren    Beim Rotieren um 8 Position wird er nicht verändert!
M0087   EQU     $0087 /* KeySpeicher              Schlüssel zum Chiffrieren
M0088   EQU     $0088 /* KeySpeicher              Schlüssel zum Chiffrieren
M0089   EQU     $0089 /* KeySpeicher              Schlüssel zum Chiffrieren
M008A   EQU     $008A /* KeySpeicherInit 1 und 2
M008B   EQU     $008B /* KeySpeicherInit 1 und 2
M008C   EQU     $008C /* KeySpeicherInit 1 und 2
M008D   EQU     $008D /* KeySpeicherInit 1 und 2
M008E   EQU     $008E /* KeySpeicherInit 1 und 2
M008F   EQU     $008F /* KeySpeicherInit 1 und 2
M0090   EQU     $0090 /* KeySpeicherInit 1 und 2 mit 0x008d - 0x008a XOR nach 0x0085 - 0x0082
M0091   EQU     $0091 /* KeySpeicherInit 1 und 2 mit 0x008d - 0x008a XOR nach 0x0085 - 0x0082
M0092   EQU     $0092 /* KeySpeicherInit 1 und 2 mit 0x008d - 0x008a XOR nach 0x0085 - 0x0082
M0093   EQU     $0093 /* KeySpeicherInit 1 und 2 mit 0x008d - 0x008a XOR nach 0x0085 - 0x0082
M0094   EQU     $0094 /* KeySpeicherInit 1 und 2
M0095   EQU     $0095 /* KeySpeicherInit 1 und 2
M0096   EQU     $0096 /* KeySpeicherInit 1 und 2
M0097   EQU     $0097 /* KeySpeicherInit 1 und 2
M0098   EQU     $0098 /* KeySpeicherInit 1 und 2
M0099   EQU     $0099 /* KeySpeicherInit
M009B   EQU     $009B /* Schleifen Variable
M009C   EQU     $009C /* Variable Tastaturwerte
M009D   EQU     $009D /* Zeiger-Variable
M00A0   EQU     $00A0 /* Variable Tastaturwerte
M00A6   EQU     $00A6 /* Variable
M00A7   EQU     $00A7 /* Zeiger auf Tastatur-Port 0x4001
M00A8   EQU     $00A8 /* Variable??
M00A9   EQU     $00A9 /* Zeiger-Variable  -> 0x004d
M00AB   EQU     $00AB /* Zeiger-Variable  -> 0x0055
M00AD   EQU     $00AD /* Zeiger-Variable  -> 0x0008
M00B1   EQU     $00B1 /* Zeiger-Variable
M00B3   EQU     $00B3 /* Variable
M00BE   EQU     $00BE /* Variable Chiffrieren/Dechiffrieren
M00C4   EQU     $00C4 /* Zeiger-Variable Tastatur ASCII
M00C5   EQU     $00C5 /* Variable
M00CA   EQU     $00CA /* Zeiger-Variable
M00CB   EQU     $00CB /* Variable
M00CC   EQU     $00CC /* Variable
M00CD   EQU     $00CD /* Variable
M00CF   EQU     $00CF /* Zeiger auf ??
M00D5   EQU     $00D5 /* 0x00d6 ... 0x00e5 Klartext-Schlüssel
M00D8   EQU     $00D8 /* Variable  ... 0x00df Kopie aus 0x004e ... 0x0056
M00DC   EQU     $00DC /* Variable
M00E2   EQU     $00E2 /* Variable
M00E4   EQU     $00E4 /* Variable
M00E6   EQU     $00E6 /* Hunderter, Anzahl der Text-Zeichen
M00E7   EQU     $00E7 /* Zehner; Anzahl der Text-Zeichen
M00E8   EQU     $00E8 /* Variable
M00EF   EQU     $00EF /* Variable
M00F0   EQU     $00F0 /* Variable
M00F1   EQU     $00F1 /* Variable
M00F2   EQU     $00F2 /* Variable
M00F3   EQU     $00F3 /* Variable
M00F4   EQU     $00F4 /* Variable
M00F5   EQU     $00F5 /* COM-Port-Puffer
M00FF   EQU     $00FF /* CPU-RAM Prüfung
M0100   EQU     $0100 /* RAM Variablen 0x0100 ... 0x0150
M0108   EQU     $0108 /* -"-           Schrittweise Initialisierung
        EQU     $012C /* Signalwert Modem
M0150   EQU     $0150 /* RAM Variablen 0x0100 ... 0x0150
M0151   EQU     $0151 /* Teil-Schlüssel
M0158   EQU     $0158 /* Adresse Teil der Schlüssel-Verwurstellung
M0161   EQU     $0161 /* Startadresse Zeichen "01234567" bis 0x0168
M016A   EQU     $016A /* Startadresse Sonderzeichen-1 RAM
M0170   EQU     $0170 /* Startadresse Suchbereich Sonderzeichen-1
M0171   EQU     $0171 /* Endadresse Sonderzeichen-1 RAM
M017D   EQU     $017D /* Startadresse Sonderzeichen-2 RAM
M0184   EQU     $0184 /* Endadresse Sonderzeichen-2 RAM
        EQU     $018C /* Signalwert Modem
M018E   EQU     $018E /* Adressberechnung 0x018e + 2*(0x002f)
M0190   EQU     $0190 /* Startadresse Init Variablenbereich (0x0190) + (0x0288) Text 1 ... n
        EQU     $01B5 /* Signalwert Modem
M0256   EQU     $0256 /* Endadresse RAM Test
M0287   EQU     $0287 /* Initaladresse, wird auch für SP genutzt
M0288   EQU     $0288 /* Endadresse Variablenadressen Text 1 ... Text n
M02FB   EQU     $02FB /* Speicherwerte; 0x058 zeigt auf 0x02fb
        EQU     $0318 /* Signalwert Modem
        EQU     $0341 /* Signalwert TimerCSR
        EQU     $036B /* Signalwert Modem
        EQU     $03E8 /* Konstante 1000
        EQU     $04E2 /* Schleifenwert Warteschleife
        EQU     $062F /* Signalwert Modem
        EQU     $06D6 /* Signalwert Modem
        EQU     $09BE /* Schleifenwert Warteschleife
M09E5   EQU     $09E5 /* RAM Segment 1
M1143   EQU     $1143 /* RAM Segment 2
        EQU     $1388 /* Schleifenwert Warteschleife bei Kaltstart
M18A1   EQU     $18A1 /* RAM Segment 3
M1FFF   EQU     $1FFF /* RAM Ende
M4001   EQU     $4001 /* Tastatur-Port-Data
M401F   EQU     $401F ????
M4040   EQU     $4040 /* unbekannter RAM zugriff
M4B75   EQU     $4B75 /* Zeiger von 0x005a
M8000   EQU     $8000 /* LCD-Port-COMMAND
M8001   EQU     $8001 /* LCD-Port-Data
```

```
M9B9B   EQU     $9B9B /* Zeiger von 0x005a
M9C9D   EQU     $9C9D /* Zeiger von 0x005a
MABAD   EQU     $ABAD /* Zeiger von 0x005a

; Funktionen JSR/BSR; No.: -> Sprungverteiler Nr.:
; ----------------------------------------
E1EB Call 0xe1eb; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger0x0073;auf D = n * (0x0190) + (0x0288)
E20D Call 0xe20d; Inhalt aus IX = 0x018e + 2*(0x002f) Rückgabe Flag
E212 Call 0xe212; Adressberechnung IX = 0x018e + 2*(0x002f) Flag
E21A Call 0xe21a; Init Textlänge Zeiger
E24E Call 0xe24e; A++ Bereich: 0x01 bis 0x063
E25A Call 0xe25a; Dekrement (0x002f), == 0 Return A = 0x63 (0x002f)
E264 Call 0xe264; Rückgabe Z-Flag
E26F Call 0xE26F; No.: 5;  Übergabe A; Rückgabe A Taste (TEXT)
E27B Call 0xe27b; Sprungverteiler 6; Abfrage Taste "TEXT"
E283 Call 0xE283; No.: 10; Abfrage "DELETE TEXT?"
E2C0 Call 0xe2c0
E2E8 Call 0xE2E8; No.: 7; Clear Text; Ausgabe Freier Speicherplatz
E2F3 Call 0xe2f3; Prüfung Textspeicher verfügbar
E30D Call 0xE30D; No.: 11 Clear all Text, Prüfe Tastendruck "Clear All"
E349 Call 0xe349; Zeitschleifen Tastatur
E35C Call 0xe35c; "Clear All" Lösche und Fülle RAM mit Initalwerten
E3AA Call 0xE3AA; No.: 12; SET MARGIN AT ...
E3E7 Call 0xe3e7
E3FB Call 0xE3FB; No.: 13; Verlgeich (0x0027) mit(0x002d)->0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
E40A Call 0xE40A; No.: 32; ... weiter mit ->0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
E416 Call 0xE416; No.: 8
E46F Call 0xE46F; No.: 9
E4E6 Call 0xE4E6; No.: 14; "INSERT TEXT 00 ? "
E542 Call 0xe542
E5BC Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B(0x0034), IX (0x002b)
E5DE Call 0xe5de
E665 Call 0xe665
E6B1 Call 0xe6b1; unverändert A
E6BF Call 0xE6BF; No.: 1
E6F1 Call 0xE6F1; No.: 2
E72F Call 0xE72F; No.: 3
E74D Call 0xE74D; No.: 4
E75E Call 0xe75e
E76C Call 0xe76c; (0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
E773 Call 0xE773; No.: 24
E777 Call 0xE777; No.: 21
E792 Call 0xE792; No.: 22
E7B0 Call 0xE7B0; No.: 23
E7D1 Call 0xE7D1; No.: 25
E807 Call 0xe807
E81E Call 0xe81e; No.: 27; Fülle Speicherbereich 0x0100... 0x014fmit 0x0 bzw. 0xff
E845 Call 0xE845; No.: 26
E853 Call 0xE853; No.: 19
E926 Call 0xe926
E963 Call 0xe963
E99A Call 0xe99a; Übergabe A, Rückgabe A &= 0xdf; if A > 0x60 and < 0x7b then a ... z -> A ... Z
E9A5 Call 0xe9a5
E9A8 Call 0xe9a8; Fülle 0x00d8 ... 0x00df mit (0x004e ...) oder 0x20 wenn (0x004e ...) == 0
E9C3 Call 0xe9c3
EA01 Call 0xea01
EA0F Call 0xea0f
EA1E Call 0xea1e; Übergabe D = 0xnn02
EA26 Call 0xea26; Übergabe D, Rückgabe
EA63 Call 0xea63; Übergabe B via D
EA73 Call 0xea73; Übergabe D
EA94 Call 0xea94; Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
EA99 Call 0xea99
EAAA Call 0xeaaa
EAB0 Call 0xeab0
EAB4 Call 0xeab4
EACD Call 0xeacd; Zeitschleife 0x64 * Wiederholungen;
EACF Call 0xeacf; Zeitschleife B * Wiederholungen; Übergabe B
EAD6 Call 0xead6; Zeitschleife x ms
EADF Call 0xeadf; IF (IX)(0x018e+2*(0x002f)) == 0; A = 0x00 else A= 0xff
EAEB Call 0xeaeb; Call A Division 10;  Übergabe A, Rückgabe A, B; A= (A/10) + 58; B Anzahl der Division
EAF5 Call 0xeaf5; Übergabe B, Rückgabe A
EB4B Call 0xeb4b
EBBF Call 0xebbf; Init 0x0046 .. 0x004d; Zeiger
EBDA Call 0xebda; Lösche 0x004e ... 0x0055
EBE6 Call 0xebe6; Initalisierung LCD mit Sonderzeichen
EC37 Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
EC3B Call 0xec3b; LCD COMMAND; Übergabe A
EC52 Call 0xec52; LCD Data nach Busy; Übergabe A
EC57 Call 0xec57; LCD Data; Übergabe A
EC66 Call 0xec66; Warte auf Busy LCD-Controller
EC74 Call 0xec74; Rückgabe  A = (0x0030) bzw. (0x0031)
ECA2 Call 0xeca2
ECC2 Call 0xecc2; Ausgabe Fehler Text auf LCD; Übergabe IX
ECC5 Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
ECCA Call 0xecca
ECCF Call 0xeccf
ECEA Call 0xecea; Textausgabe LCD, Übergabe IX
ECF6 Call 0xecf6; -> Call 0xeeae; Längenberechnung Übergabe D, IX
ECF9 Call 0xecf9; Rückgabe A
ED7F Call 0xed7f; ---> 0xeec5; Übergabe A
ED82 Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
ED92 Call 0xed92
EDCD Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
EDEC Call 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe
```

```
EDF6 Call 0xedf6; Textnr. Chiffriert
EEAB Call 0xeeab; Längenberechnung D, IX 0x00e4 Zeiger auf ...
EEAE Call 0xeeae; Längenberechnung Übergabe D, IX
EEC5 Call 0xeec5; 8/4-Bitausgabe LCD Steuerwort;
EECD Call 0xEECD; No.: 31
EED3 Call 0xEED3; No.: 20
EEDD Call 0xEEDD; No.: 34 -> Daten inkonsistent 0xEE9D
EF37 Call 0xed37; Sende Akustik-Koppler
EFB8 Call 0xefb8; Senden, Übergabe A, B
F001 Call 0xF001; No.: 33 Empfangen und Dechiffrieren via Sprungverteiler 35
F006 Call 0xF006; No.: 35 Empfangen und Dechiffrieren
F06A Call 0xf06a; Empfangen (LOAD/Receiv) Übergabe A (0x003c); Rückgabe
F13C Call 0xf13c
F234 Call 0xf234
F252 Call 0xf252; Kopiere (0x0025) > (0x002d), A = 0xff -> (0x003a), (0x8d)= 0x0
F25E Call 0xf25e; Bereitschaft Ready to Receive
F27E Call 0xF27E; Bearbeitung RAM bis 0x03b6, Rückgabe B (1 oder 0); Modem
F2CC Call 0xf2cc; Übergabe B (1 oder 0); Rückgabe IX = 0x0008 oder 0x0001
F2E1 Call 0xf2e1
F2FD Call 0xf2fd; Empfangsroutine; Rückgabe A
F377 Call 0xF377; No.: 36;  Senden - Tonbandausgabe
F3BE Call 0xf3be; Lese Port-1 Data; Rückgabe B mit Bit 4 = 1
F3C8 Call 0xf3c8; Lese Port-1 Data; Rückgabe B mit Bit 2 = 1
F3D5 Call 0xf3d5
F3F9 Call 0xe3f9; Tastendruck; Rückgabe A
F437 Call 0xf437; Tastaturabfrage Rückgabe A+B, IX = 0x009d, z-Flag
F4EB Call 0xf4eb; Kopiere (0x00a9)->byte->(0x00ab); Adresse--; Schleifenzähler: (0x00ad)--
F501 Call 0xf501; Lösche 0x016a ... 0x018f mit 0x0, Fülle 0x016a und 0x17d + n mit Werten aus 0xffda ... 0xffe8; n = 0 ... 7"
F52C Call 0xf52c; Suche Byte größer 0 ab Adr. 0x0170 Rückgbabe B (>0x06Anzahl?)
F539 Call 0xf539
F58C Call 0xf58c; Wandelt Hex in ASCII-Hex-Code 0x0 -> 00 0xff ->FF; Übergabe A; Rückgabe B, A
F597 Call 0xf597; DAA Übergabe A; Rückgabe A
F59E Call 0xF59E; No.: 15
F5D9 Call 0xf5d9
F611 Call 0xf611
F61C Call 0xf61c
F646 Call 0xf646; Serielle Daten lesen
F660 Call 0xf660; Serielle Daten lesen Übergabe A
F6A5 Call 0xf6a5; Init Serielle und Timer
F6B2 Call 0xF6B2; No.: 16
F759 Call 0xf759; IX Adresse 0x0058 Zeiger auf 0x02fb
FA16 Call 0xfa16; Übergabe A Rückgabe AF  /* A-high = (!A-low | A-high);A-low unverändert
FA20 Call 0xFA20; No.: 28; EN/DECRYPTION
FA5B JP   0xfa5b; Schlüsselwort-verwurstelung
FABB JP   0xfabb; PRGN
FC93 Call 0xFC93; No.: 29 Eingabe Schlüssel
FD08 Call 0xFD08; No.: 30
FD68 Call 0xfd68
FE21 Call 0xfe21; Abfrage IX (0x0025) <= (0x0027); while (0x0025 <=0x0027)ix++ 0x0025 erstes ASCII im Speicher TestIfStringIsHex

; Internal Registers 0x0000-0x001F
; ------------------------------------------
symbol 0x0000 ioPort1dataDirection DDR
symbol 0x0001 ioPort2dataDirection DDR -> Serieller Port; Senden Port2Pin4, Empfangen Port2Pin3, Takt Port2Pin2
symbol 0x0002 ioPort1data          DATA
symbol 0x0003 ioPort2data          DATA -> Serieller Port; Empfangen
symbol 0x0008 timerControl and Status Register; 8 bit lesen, untere 5bit Schreiben
symbol 0x0009 Counter High-byte
symbol 0x000A Counter Low-byte
symbol 0x000B OutputCompare High-byte Status TCSR1
symbol 0x000C OutputCompare Low-byte Status TCSR1
symbol 0x000D InputCapture High-byte
symbol 0x000E InputCapture Low-byte
symbol 0x0010 SerialRateAndModeControlRegister; bit0,1 Speed; bit2,3 Format-ClockSource-Port2Pin2.. 4defined
symbol 0x0011 SerialControlAndStatusRegister, Transmit/Receive Control and Satus Register
symbol 0x0012 SerialReceiverDataRegister
symbol 0x0013 SerialTransmitDataRegister
symbol 0x0014 RWMemoryControlRegister

**************************************************
** Pin/Port-Beschaltung                          *
**************************************************
PortNr.  IC-Pin Funktiom
 P20      8     RX
 P15     18     CDT
 P14     17     TX
 P12     15     Betriebsgeschwindigkeit Modem
 P11     14     Modem Ein/Aus
 P24     12     Druckerausgabe; getrickst 1 Start 9 Data 1 Stopp; Bit 9 immer auf 1 setzen für 2 Stopp-bit
 P17     20     Power/Modem-LED
 P23     11     Drucker Ready
 P21      9     Masse
 P22     10     ON/OFF - Standby
 P10     13     Spannungsunterschreitung

Port-2 bit DDR Bit == 0 INPUT; Bit == 1 OUTPUT
-          7   6   5   4   3   2   1   0
          PC2 PC1 PC0 P24 P23 P22 P21 P20
OUTPUT                PRT     ON/OFF Standby
INPUT                     Ready   Gnd  RX

Port-1-
          7   6   5   4   3     2        1          0
         P17 P16 P15 P14 P13   P12      P11        P10
Output   LED Keyb     TX Keyb Speed  Modem On/Off
```

```
Input            CDT                                U.min

; 4700 Klartext/GTX Zeichen == 0x125c RAM für Klar-und Geheim- Text
; 0x025c für Schlüssel und SP

; External Memory Space 0x0020-0x007F
; ------------------------------------------
; Internal Ram 0x0080-0x00FF
; ------------------------------------------
symbol 0x0080 RAM
; Main External Ram 0x0100-0x01FF
// RAM Prüfung von 0x0020 bis 0x0288 !!!!!!!!
; ------------------------------------------
; I/O 0x4000-0x4001 & 0x8000-0x8001
; ------------------------------------------
symbol 0x4000 KeyboardCommands
symbol 0x4001 KeyboardData
symbol 0x8000 DisplayCommands
symbol 0x8001 DisplayData

; EPROM / ROM 0xE000-0xFFFF
; ------------------------------------------
vector 0xFFF0 sci vector sci entry
vector 0xFFF2 tof vector tof entry
vector 0xFFF4 ocf_vector ocf_entry
vector 0xFFF6 icf_vector icf_entry
vector 0xFFF8 irq vector int entry
vector 0xFFFA swi_vector swi_entry
vector 0xFFFC nmi_vector nmi_entry
vector 0xFFFE res_vector reset

************************************************
** Tip                                         *
** #M oder #$ direkter Wert                    *
** $ immer hex-werte                           *
** ohne # bzw. ohne #M aus Speicherzelle xy    *
** BNI == BMI Intel                            *
** Übertragungsoformat                         *
** Audio Start 0 ... 8 Parity(even) St.St.     *
** Daten Start 0 ... 7 Parity(even) St.St.     *
************************************************

/*****************************************************************************************************************\
/* Tasten:          ----> Zeichencodierung siehe 0xff02 <----                                                    *\
/* ON/STOP + B           Lautsprecher aus                                                                        *\
/*         + K           Conversion Character                                                                    *\
/* ON/STOP               Ein/Aus, Lautsprecher zurücksetzen                                                      *\
/*                       Restart wenn im Sleep - nach 50 Sekunden warten                                         *\
/* ON/STOP + PRINT       Löscht gesammten RAM                                                                    *\
/* Shift L/R             Großschreibung                                                                          *\
/* CAPS LOCK             immer -"-                                                                               *\
/* Shift + CLEAR ALL     = CLR LINE; Löscht Textblock                                                            *\
/* Shift + DUMP          = LOAD Text; einlesen via Tape-Anschluß-> "READ TEXT OO FROM TAPE"                       *\
/* Shift + DELETE        = DEL TXT; lösche alle Texte                                                            *\
/* Shift + INSERT        = INS TXT; Insert Text                                                                  *\
/* Shift + PRINT         List Text                                                                               *\
/* Shift left + CODE     = KEY; Neuen Schlüssel eingeben, 16 Zeichen-> "NEW KEY...+ PRESS AGAIN"                 *\
/* Shift left + MARGIN   = Low; Verringerung/Erhöhung Übertragungsgeschwindigkeit                                *\
/* Shift left + TAB      = SET Tab ab Position des Cursor                                                        *\
/* Shift left + TEXT     = v Pfeil;Cursor an den start des nächsten höher nummerierten Textblock                 *\
/* Shift right + CODE    = TXT; Schlüssel aus einem Textblock verwenden-> "NEW KEY TEXT OO + PRESS AGAIN"        *\
/* Shift right + MARGIN  = High; Verringerung/Erhöhung Übertragungsgeschwindigkeit                               *\
/* Shift right + TAB     = CLR; Lösche TAB                                                                       *\
/* Shift right + TEXT    = ^ Pfeil;Cursor an den Start des nächsten höher nummerierten Textblock                 *\
/* SPACE                 Leerzeichen wird mit Unterstrich im LCD angezeigt                                       *\
/* Shift <-              Cursor ein Zeichen rückwärts                                                            *\
/* Shift <<-             Cursor an den Anfang der Zeile                                                          *\
/* Shift v               Cursor ein Zeichen abwärts                                                              *\
/* Shift vv              Cursor an das Ende des Texblockes                                                       *\
/* Shift ^               Cursor zur ersten Zeile                                                                 *\
/* Shift ^^              Cursor an den Anfang des Texblockes                                                     *\
/* Shift ->              Cursor ein Zeichen vorwärts                                                             *\
/* Shift ->>             Cursor an das Ende der Zeile                                                            *\
/* TEXT                  Beginne neuen Textblock                                                                 *\
/* TAB                   Tabulator                                                                               *\
/* DELETE                Zeichen löschen                                                                         *\
/* CLEAR ALL             Löscht alle Texte                                                                       *\
/* MARGIN                Anzahl der Zeichen pro Zeile (10 ... 80)                                                *\
/* SERACH                Suche 1 ... 8 Zeichen, ON/STOP Abbruch                                                  *\
/* DUMP                  Textausgabe via Tape-Anschluß                                                           *\
/* SEND                  Sende Text, Signal-LED wird eingeschalten! ?Pin/Port LED?                               *\
/* RCVE                  Empfange Text, LED blinkt                                                               *\
/* CODE                  De,- Chiffrieren, nur wenn Schlüssel eingegeben ist                                     *\
/* PRINT                 bei ansgeschlossenen Drucker, Ausgabe                                                   *\
/*                       getrickst 1 Start 9 Data 1 Stopp; Bit 9 immer auf 1 setzen für 2 Stopp-bit              *\
/* Shift left  1         !   0x21                                                                                *\
/* Shift right 1         @   0xC0                                                                                *\
/* Shift left  2         ~                                                                                       *\
/* Shift right 2         &   0xA0                                                                                *\
/* Shift left  3         #   0xA3                                                                                *\
/* Shift right 3         £   0x60                                                                                *\
/* Shift left  4         $   0x24                                                                                *\
/* Shift right 4         *   0xAA                                                                                *\
/* Shift left  5         Ā   0x82                                                                                *\
/* Shift right 5         ā   0x03                                                                                *\
```

```
/* Shift left  6          Ä   0x84                                                                                    *\
/* Shift right 6          ä   0x7B                                                                                    *\
/* Shift left  7          Ö   0x05                                                                                    *\
/* Shift right 7          ö   0xFC                                                                                    *\
/* Shift left  8          Ü   0xDE                                                                                    *\
/* Shift right 8          ü   0x7E                                                                                    *\
/* Shift left  9          (   0x28                                                                                    *\
/* Shift right 9          )   0xA9                                                                                    *\
/* Shift left  0                                                                                                      *\
/* Shift right 0          Ø                                                                                           *\
/* Shift left  -          +   0x2B                                                                                    *\
/* Shift right -          %                                                                                           *\
/* Shift left  :          <   0x3C                                                                                    *\
/* Shift right :          >   0xBE                                                                                    *\
/* Shift left  ;          ?   0x3F                                                                                    *\
/* Shift right ;          '   0x27                                                                                    *\
/* Shift left  ,          [   0xDB                                                                                    *\
/* Shift right ,          ]   0xDD                                                                                    *\
/* Shift left  .          /   0xAF                                                                                    *\
/* Shift right .          \   0x5C                                                                                    *\
                                                                                                                      *\
/******************************************************************************************************************\

/*****************************************************************************************\
/* Codierung des ASCII mit Parity                                                        *\
/*                                                                                       *\
/* RAW: E1 E2 63 E4 65 66 E7 E8 69 6A EB 6C ED EE 6F F0 71 72 F3 74 F5 F6 77 78 F9 FA    *\
/* Txt:  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z    *\
/*                                                                                       *\
/* RAW: 41 42 C3 44 C5 C6 47 48 C9 CA 4B CC 4D 4E CF 50 D1 D2 53 D4 55 56 D7 D8 59 5A    *\
/* Txt:  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z    *\
/*                                                                                       *\
/* RAW: B1 B2 33 B4 35 36 B7 B8 39 30 2D 3A BB 2E AC 8D                                  *\
/* Txt:  1  2  3  4  5  6  7  8  9  0  -  :  ;  .  , wr/zv                               *\
/*                                                                                       *\
/* RAW: 21 22 A3 24 82 84 05 DE 28 BD 2B 3C 3F AF DB                   // shift left    *\
/* Txt:  !  "  #  $  Ä  Ä  Ö  Ü  (  =  +  <  ?  / [                                     *\
/*                                                                                       *\
/* RAW: C0 A6 60 AA 03 7B FC 7E A9 FF A5 BE 27 5C DD                   // shift right   *\
/* Txt:  @  &  £  *  ä  ä  ö  ü  )  Ø  %  >  '  \  ]                                    *\
/*****************************************************************************************\

/*****************************************************************************************\
/* 14 LCD Symbole und Befehls-byte                                                       *\
/* - - - - - - - - - --1-1-1-1-1-1-1-1-2-2-2-2-2-2-2-2-2-3-3-3-3-3-3-3-3-3-3-3           *\
/* 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 9 0 1 2 3 4 5 6 7 8 9       *\
/*                                                                                       *\
/* Telefon:XXXX  FORMAT:8040L    SHIFT LOCK  BATT      INSERT  M:XXXX    LINE TEXT       *\
/*    0xc0-0xc4  0xc5 0xc7 0xcb  0xcc  0xcf  0xd5      0xd6    0xdd-0xe0                  *\
/* Pos.:                 0xca                                  0xd2 0xd9                 *\
/* 64     68       69    71   74    75    76   79      82  93   96    85               *\
/*****************************************************************************************\

************************************************
** Program Code / Data Areas                   *
************************************************

        ORG     $E000

*E000: 7E E3 39       '~.9'           ZE000   JMP     ZE339       <---\                 // Kaltstart
// Einsprung sowie Interrupt-Sprungverteiler                           |
*E003: CE 13 88       '...'           ZE003   LDX     #M1388          |                 // IX = 0x1388; Schleifenzähler
*E006: 09             '.'             ZE006   DEX           <---\     |                 // IX--; IX =0x1387
*E007: 26 FD          '&.'                    BNE     ZE006 >---/     |                 // Schleife xx ms Wartezeit if (z) == 0
*E009: 7F 00 08       '...'                   CLR     >M0008          |                 // (0x0008) = 0; Port21 Output, TCSR1 Timer Control Status 1
*E00C: 8E 00 FF       '...'                   LDS     #M00FF          |                 // SP = 0x00ff;
*E00F: 86 CA          '..'                    LDAA    #$CA            |                 // A = 0xca; 1100 1010; 0 = Eingang 1 = Ausgang
*E011: 97 02          '..'                    STAA    M0002           |                 // (Port-1-Data) = A;
*E013: 86 DE          '..'                    LDAA    #$DE            |                 // A = 0xde; 1101 1110
*E015: 97 00          '..'                    STAA    M0000           |                 // (Port-1-DDR) = A; Port-1 Init I/O
*E017: 86 14          '..'                    LDAA    #$14            |                 // A = 0x14; 0001 0100
*E019: 97 03          '..'                    STAA    M0003           |                 // (Port-2 Data) = A;
*E01B: 97 01          '..'                    STAA    M0001           |                 // (Port-2 DDR) = A;
*E01D: CE 00 20       '.. '                   LDX     #M0020          |                 // IX = 0x0020
// Schleife                                                            |
*E020: A6 00          '..'            ZE020   LDAA    ,X      <---\   |                 // A = (IX); ab 0x0020
*E022: A1 00          '..'                    CMPA    ,X          |   |                 // cmp A, (IX); muß gleich sein
*E024: 26 DA          '&.'                    BNE     ZE000       | >-+                 // if (z) == 0; Kaltstart RAM-Fehler
*E026: 63 00          'c.'                    COM     ,X          |   |                 // !(IX); complement (IX)
*E028: 43             'C'                     COMA                |   |                 // !A; Complement A
*E029: A1 00          '..'                    CMPA    ,X          |   |                 // cmp A, (IX); Muß gleich sein!
*E02B: 26 D3          '&.'                    BNE     ZE000       | >-+                 // if (z) == 0; RAM Fehler
*E02D: 63 00          'c.'                    COM     ,X          |   |                 // !(IX)
*E02F: 43             'C'                     COMA                |   |                 // A != A
*E030: A1 00          '..'                    CMPA    ,X          |   |                 // cmp A, (IX); muß gleich sein
*E032: 26 CC          '&.'                    BNE     ZE000       | >-/                 // if (z) == 0; RAM Fehler
*E034: 08             '.'                     INX                 |                     // IX++; 0x0021 ...
*E035: 8C 02 88       '...'                   CPX     #M0288      |                     // cmp IX, 0x0288
*E038: 25 E6          '%.'                    BCS     ZE020 >-----/                     // if (c) == 1; von 0x0020 bis 0x0288 RAM prüfen
// RAM OK
*E03A: 8E 02 87       '...'                   LDS     #M0287                            // SP = 0x0287
*E03D: 72 04 08       'r..'                   OIM     #$04,M0008                        // (0x0008) |= 0x04; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*E040: 7F 00 57       '..W'                   CLR     >M0057                            // (0x0057) = 0
*E043: 86 05          '..'                    LDAA    #$05                              // A = 0x05
*E045: 97 56          '.V'                    STAA    M0056                             // (0x0056) = A; D = 0x0500
```

```
*E047: BD F7 59       '..Y'                   JSR     ZF759                                   // Call 0xf759; Adresse 0x0058 Zeiger auf 0x02fb
*E04A: 0E             '.'                     CLI                                             // Clear Interrupt Mask = 0
*E04B: BD EB E6       '...'                   JSR     ZEBE6                                   // Call 0xebe6   Initalisierung LCD mit Sonderzeichen
*E04E: 7F 00 38       '..8'                   CLR     >M0038                                  // (0x0038) = 0
*E051: 7F 00 37       '..7'                   CLR     >M0037                                  // (0x0037) = 0
*E054: CE 01 61       '..a'                   LDX     #M0161                                  // IX = 0x0161
*E057: 86 30          '.0'                    LDAA    #$30                                    // A = 0x30 "0"
*E059: C6 08          '..'                    LDAB    #$08                                    // B = 0x08; Rundenzähler
// Schleife Suche A in (IX) Warmstart/Kaltstart?
*E05B: A1 00          '..'            ZE05B   CMPA    ,X    <----\                            // cmp A, (IX); 0x30 + n == (0x0161 + n) ? n =  1 ... 8
*E05D: 26 07          '&.'                    BNE     ZE066      | >---\                      // if (z) == 0 Ausgabe "Copyright ..."
*E05F: 4C             'L'                     INCA               |     |                      // A++
*E060: 08             '.'                     INX                |     |                      // IX++
*E061: 5A             'Z'                     DECB               |     |                      // B--
*E062: 26 F7          '&.'                    BNE     ZE05B  >---/     |                      // if (z) == 0 Schleife 8 Runden
*E064: 20 19          ' .'                    BRA     ZE07F            | >---\                // JR 0xe07f
// JP Ausgabe Copyright                                                |     |
*E066: CE F8 34       '..4'           ZE066   LDX     #MF834       <---/     |                // IX = 0xf834; "COPYRIGHT 1984 WEST-TEC PX V2"
*E069: BD EC C5       '...'                   JSR     ZECC5                  |                // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*E06C: BD F5 01       '...'                   JSR     ZF501                  |                // Call 0xf501;Lösche 0x016a ... 0x018f mit 0x0, Fülle 0x016a und 0x17d + n mit Werten aus 0xffda ... 0xffe8; n = 0 ... 7
*E06F: CE 01 51       '..Q'                   LDX     #M0151                 |                // IX = 0x0151
*E072: C6 10          '..'                    LDAB    #$10                   |                // B = 0x10 Schleifenzähler
*E074: 86 0E          '..'                    LDAA    #$0E                   |                // A = 0x0e
// Schleife 16 Runden Fülle 0x0151 bis 0x0160 mit 0x0e; Schlüsselspeicher?    |
*E076: A7 00          '..'            ZE076   STAA    ,X    <---\            |                // (IX) = A
*E078: 08             '.'                     INX               |            |                // IX++
*E079: 5A             'Z'                     DECB              |            |                // B--
*E07A: 26 FA          '&.'                    BNE     ZE076 >---/            |                // if (z) == 0; 16 Runden
*E07C: 7E E3 42       '~.B'                   JMP     ZE342                  |                // JP 0xe342; Sprungverteiler - Tastatur
// JP Kaltstart                                                              |
*E07F: 96 3A          '.:'            ZE07F   LDAA    M003A              <---/                // A = (0x003a)
*E081: 81 FF          '..'                    CMPA    #$FF                                    // CMP A, 0xff
*E083: 26 06          '&.'                    BNE     ZE08B >---\                             // if (z) == 0
*E085: 7F 00 CC       '...'                   CLR     >M00CC    |                             // (0x00cc) = 0
*E088: BD F1 3C       '..<'                   JSR     ZF13C     |                             // Call 0xf13c
// 0xff in (0x003a)                                             |
*E08B: 86 FF          '..'            ZE08B   LDAA    #$FF  <---/                           // A = 0xff
*E08D: 97 3B          '.;'                    STAA    M003B                                   // (0x003b) = A
*E08F: 96 36          '.6'                    LDAA    M0036                                   // A = (0x0036); Tastatur z/s
*E091: 97 73          '.s'                    STAA    M0073                                   // (0x0073) = A
*E093: 7F 00 36       '..6'                   CLR     >M0036                                  // (0x0036) = 0; Tastatur z/s
*E096: BD F4 37       '..7'                   JSR     ZF437                                   // Call 0xf437;  Tastaturabfrage Rückgabe A+B, IX = 0x009d, z-Flag
*E099: D7 C5          '..'                    STAB    M00C5                                   // (0x00c5) = B
*E09B: 81 62          '.b'                    CMPA    #$62                                    // CMP A, 0x62
*E09D: 26 06          '&.'                    BNE     ZE0A5 >---\                             // if (z) == 0
*E09F: 73 00 3B       's.;'                   COM     >M003B    |                             // Complement (0x003b); 0xff -> 0x00
*E0A2: 7E E1 43       '~.C'                   JMP     ZE143     |                             // JP 0xe143
                                                                |
*E0A5: 81 7A          '.z'            ZE0A5   CMPA    #$7A  >---/                           // CMP A, 0x7a
*E0A7: 26 03          '&.'                    BNE     ZE0AC >----\                          // if (z) == 0
*E0A9: 7E E0 66       '~.f'                   JMP     ZE066      |                          // JP 0xe066; Ausgabe "Copyright ..."
                                                                 |
// JP Verleich A == 0x6d                                         |
*E0AC: 81 6D          '.m'            ZE0AC   CMPA    #$6D  <----/                          // CMP A, 0x6d
*E0AE: 26 2B          '&+'                    BNE     ZE0DB                    >---\        // if (z) == 0 Auswertung Funktionstasten -> Sprungverteiler
*E0B0: CE 00 20       '.. '           ZE0B0   LDX     #M0020                <---\ |        // IX = 0x0020
// Schleife bis IX = 0x1fff                                                   | |
*E0B3: A6 00          '..'            ZE0B3   LDAA    ,X     <---\          | |        // A = (IX)
*E0B5: 63 00          'c.'                    COM     ,X         |          | |        // Complement (IX)
*E0B7: 43             'C'                     COMA               |          | |        // A != A
*E0B8: A1 00          '..'                    CMPA    ,X         |          | |        // cmp A, (IX)
*E0BA: 26 0F          '&.'                    BNE     ZE0CB      | >---\    | |        // if (z) == 0; Ausgabe: Speicherfehler
*E0BC: 63 00          'c.'                    COM     ,X         |     |    | |        // Complement (IX)
*E0BE: 43             'C'                     COMA               |     |    | |        // A != A
*E0BF: A1 00          '..'                    CMPA    ,X         |     |    | |        // cmp A, (IX)
*E0C1: 26 08          '&.'                    BNE     ZE0CB      | >---|    | |        // if (z) == 0; Ausgabe: Speicherfehler
*E0C3: 08             '.'                     INX                |     |    | |        // IX++
*E0C4: 8C 1F FF       '...'                   CPX     #M1FFF     |     |    | |        // cmp IX, 0x1fff
*E0C7: 23 EA          '#.'                    BLS     ZE0B3  >---/     |    | |        // if (c) | (z) == 1; Schleife bis IX = 0x1fff
*E0C9: 20 08          ' .'                    BRA     ZE0D3            | >--\ | |        // JR 0xe0d3
                                                                       |    | | |
*E0CB: CE F8 12       '...'           ZE0CB   LDX     #MF812       <---/    | | |        // IX = 0xF812; "MEMORY ERROR"
*E0CE: BD EC C2       '...'                   JSR     ZECC2                 | | |        // Call 0xecc2; Ausgabe Fehler Text auf LCD; Übergabe IX
*E0D1: 20 DD          ' .'                    BRA     ZE0B0                 | >-/ |        // JR 0xe0b0
                                                                            |   |
*E0D3: CE F8 09       '...'           ZE0D3   LDX     #MF809            <---/   |        // IX = 0xf809; "MEMORY OK";
*E0D6: BD EC C5       '...'                   JSR     ZECC5                     |        // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*E0D9: 20 68          ' h'                    BRA     ZE143                     |  >---\ // JR 0xe143
                                                                                |      |
// JP Vergleich auf A == 0x6b; .... 0x79: 6 Funtkionstasten                     |      |
*E0DB: 81 6B          '.k'            ZE0DB   CMPA    #$6B                   <---/      | // CMP A, 0x6b
*E0DD: 26 05          '&.'                    BNE     ZE0E4  >----\                     | // if (z) == 0
*E0DF: BD F5 39       '..9'                   JSR     ZF539       |                     | // Call 0xf539; Zeichenkonvertierung Übergabe A,
*E0E2: 20 5F          ' _'                    BRA     ZE143       |                  >--+ // JR 0x0e143
                                                                  |                  |
// JP Vergleich A == 0x6c Rückgabe B                              |                  |
*E0E4: 81 6C          '.l'            ZE0E4   CMPA    #$6C   <----/                  | // CMP A, 0x6c
*E0E6: 26 05          '&.'                    BNE     ZE0ED  >----\                  | // if (z) == 0
*E0E8: BD F5 2C       '..,'                   JSR     ZF52C       |                  | // Call 0xf52c; Suche Byte größer 0 ab Adr. 0x0170 Rückgbabe B (> 0x06 Anzahl?)
*E0EB: 20 56          ' V'                    BRA     ZE143       |                  >--+ // JR 0x0e143
                                                                  |                  |
// JP Vergleich auf A == 0x64 Rückgabe B = 0x01, IX = 0x00cb      |                  |
*E0ED: 81 64          '.d'            ZE0ED   CMPA    #$64   <----/                  | // CMP A, 0x64
*E0EF: 26 07          '&.'                    BNE     ZE0F8  >----\                  | // if (z) == 0
*E0F1: CE 00 CB       '...'                   LDX     #M00CB      |                  | // IX = 0x00cb
```

```
*E0F4: C6 01          '..'                    LDAB    #$01        |                           |      // B = 0x01
*E0F6: 20 09          ' .'                    BRA     ZE101       |     >---\                 |      // JR 0xe101
                                                                  |         |                 |
// JP Vergleich auf A == 0x65                                     |         |                 |
*E0F8: 81 65          '.e'              ZE0F8 CMPA    #$65   <----/         |                 |      // CMP A, 0x65
*E0FA: 26 20          '& '                    BNE     ZE11C    >------\     |                 |      // if (z) == 0; A != 0x65;
*E0FC: CE 4B 75       '.Ku'                   LDX     #M4B75          |     |                 |      // IX = 0x4b75; Zeitschleife
*E0FF: C6 27          '.''                    LDAB    #$27            |     |                 |      // B = 0x27
*E101: DF 5A          '.Z'              ZE101 STX     M005A           | <---/                 |      // (0x005a) = IX
*E103: D7 5C          '.\'                    STAB    M005C           |                       |      // (0x005c) = B; RAM 0x005a: 4b 75 27; Folge?
*E105: BD F3 BE       '...'                   JSR     ZF3BE           |                       |      // Call 0xf3be; Lese Port-1 Data; Rückgabe B mit Bit 4 = 1
*E108: 7F 00 08       '...'                   CLR     >M0008          |                       |      // (0x0008) = 0; Port21 Output, TCSR1 Timer Control Status 1
*E10B: 0E             '.'                     CLI                     |                       |      // Clear Interrupt Mask = 0
*E10C: 75 10 02       'u..'             ZE10C EIM     #$10,M0002 <--\ |                       |      // (Port-1 Data) XOR= 0x10; toggle Port-14 TX
// Zeitschleife B * (IX) in Sekunden                                | |                       |
*E10F: DE 5A          '.Z'                    LDX     M005A           | |                       |      // IX = (0x005a) [0x4b75] Zeitschleife
*E111: D6 5C          '.\'                    LDAB    M005C           | |                       |      // B = (0x005c)
*E113: 09             '.'               ZE113 DEX           <---\   | |                       |      // IX--
*E114: 26 FD          '&.'                    BNE     ZE113 >---+   | |                       |      // if (z) == 0; Zeitschleife 1
*E116: 5A             'Z'                     DECB              |   | |                       |      // B--
*E117: 26 FA          '&.'                    BNE     ZE113 >---/   | |                       |      // if (z) == 0; Zeitschleife 2
*E119: 01             '.'                     NOP                   | |                       |      // NOP; ein Takt warten bis nächste Abfrage
*E11A: 20 F0          ' .'                    BRA     ZE10C >-------/ |                       |      // JR 0xe10c
                                                                      |                       |
// JP Übergabe A, Vergleich auf A == 0x79                             |                       |
*E11C: 81 79          '.y'              ZE11C CMPA    #$79    <-------/                       |      // CMP A, 0x79
*E11E: 26 1C          '&.'                    BNE     ZE13C >----\                            |      // if (z) == 0
*E120: 7F 00 CC       '...'                   CLR     >M00CC     |                            |      // (0x00cc) = 0
*E123: BD F2 34       '..4'                   JSR     ZF234      |                            |      // Call 0xf234; Textspeicher verfügbar
*E126: BD F2 5E       '..^'                   JSR     ZF25E      |                            |      // Call 0xf25e; Ready to Receive
*E129: C6 08          '..'                    LDAB    #$08       |                            |      // B = 0x08
*E12B: D7 8A          '..'                    STAB    M008A      |                            |      // (0x008a) = B
*E12D: C6 01          '..'                    LDAB    #$01       |                            |      // B = 0x01
*E12F: BD F2 CC       '...'                   JSR     ZF2CC      |                            |      // Call 0xf2cc; Übergabe B = 1; Rückgabe IX = 0x0008 oder 0x0001
*E132: BD F2 E1       '...'                   JSR     ZF2E1      |                            |      // Call 0xf2e1; LCD-Kommandos 0x01, 0x80 Zeitschleife; Abfrage Port-1-Data
*E135: 96 3C          '.<'                    LDAA    M003C      |                            |      // A = (0x003c)
*E137: BD F0 6A       '..j'                   JSR     ZF06A      |                            |      // Call 0xf06a; Empfangen (LOAD/Receiv) Übergabe A (0x003c); Rückgabe
*E13A: 20 07          ' .'                    BRA     ZE143      |                        >---+      // JR 0xe143
                                                                 |                            |
// JP Vergleich auf A == 0x68                                    |                            |
*E13C: 81 68          '.h'              ZE13C CMPA    #$68  <----/                            |      // CMP A, 0x68
*E13E: 26 03          '&.'                    BNE     ZE143                                 >---+      // if (z) == 0
*E140: BD FD 68       '..h'                   JSR     ZFD68                                     |      // Call 0xfd68
// JP Weiter kein passender Vergleich                                                           |
*E143: 96 73          '.s'              ZE143 LDAA    M0073       <---------------------------/      // A = (0x0073)
*E145: 97 36          '.6'                    STAA    M0036                                          // (0x0036) = A; Tastatur z/s
*E147: 96 30          '.0'                    LDAA    M0030                                          // A = (0x0030)
*E149: 36             '6'                     PSHA                                                   // Push A
*E14A: BD E5 BC       '...'                   JSR     ZE5BC                                          // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E14D: BD E7 6C       '..l'                   JSR     ZE76C                                          // Call 0xe76c; (0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
*E150: 32             '2'                     PULA                                                   // Pop A
*E151: D1 34          '.4'                    CMPB    M0034                                          // cmp B, (0x0034)
*E153: 25 07          '%.'                    BCS     ZE15C >----\                                   // if (c) == 1
*E155: 81 50          '.P'                    CMPA    #$50       |                                   // CMP A, 0x50; "P"
*E157: 22 03          '".'                    BHI     ZE15C >----+                                   // if (c) & (z) == 0
*E159: 11             '.'                     CBA                |                                   // cmp A  B
*E15A: 24 02          '$.'                    BCC     ZE15E      |              >---\                 // if (c) == 0
*E15C: D7 30          '.0'              ZE15C STAB    M0030 <----/                 |                 // (0x0030) = B
//JP                                                                               |
*E15E: 8E 02 87       '...'             ZE15E LDS     #M0287                    <---+                 // SP = 0x0287
*E161: 86 01          '..'                    LDAA    #$01                         |                 // A = 0x01
*E163: 97 56          '.V'                    STAA    M0056                        |                 // (0x0056) = A
*E165: 86 FF          '..'                    LDAA    #$FF                         |                 // A = 0xff
*E167: 97 57          '.W'                    STAA    M0057                        |                 // (0x0057) = A; (0x0056) = 0x01ff CPU-RAM Ende
*E169: BD EC A2       '...'                   JSR     ZECA2                        |                 // Call 0xeca2; Rückgabe A für Berechnung Sprungverteiler
// JP                                                                               |
*E16C: BD F3 F9       '...'             ZE16C JSR     ZF3F9 <----\                 |                 // Call 0xe3f9; Tastendruck; Rückgabe A
*E16F: 7F 00 3A       '..:'                   CLR     >M003A     |                 |                 // (0x003a) = 0
// JP Übergabe A (Taste)                                         |                 |
*E172: 81 90          '..'              ZE172 CMPA    #$90       |                 |      <-\        // CMP A, 0x90; Taste "<-"
*E174: 24 1F          '$.'                    BCC     ZE195      |     >---\       |        |        // if (c) == 0 -> Berechnung Sprünge via IX; Übergabe A (A >= 0x90); Rückgabe A -=0x90
*E176: 36             '6'                     PSHA               |         |       |        |        // Push A
*E177: BD EA DF       '...'                   JSR     ZEADF      |         |       |        |        // Call 0xeadf;  IF (IX)(0x018e+2*(0x002f)) == 0; A = 0x00 else A = 0xff
*E17A: 26 06          '&.'                    BNE     ZE182      |  >---\  |       |        |        // if (z) == 0
*E17C: 32             '2'                     PULA               |      |  |       |        |        // Pop A
*E17D: BD EA 94       '...'                   JSR     ZEA94      |      |  |       |        |        // Call 0xea94;Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E180: 20 EA          ' .'                    BRA     ZE16C >----/      |  |       |        |        // JR 0xe16c
                                                                        |  |       |        |
*E182: BD E2 0D       '...'             ZE182 JSR     ZE20D         <---/  |       |        |        // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E185: 26 03          '&.'                    BNE     ZE18A >----\         |       |        |        // if (z) == 0
*E187: BD E2 C0       '...'                   JSR     ZE2C0      |         |       |        |        // Call 0xe2c0; Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*E18A: 32             '2'               ZE18A PULA         <----/         |       |        |        // Pop A
*E18B: BD E5 DE       '...'                   JSR     ZE5DE              |       |        |        // Call 0xe5de; Übergabe A
*E18E: 20 CE          ' .'                    BRA     ZE15E              |   >---+        |        // JR 0xe15e
                                                                         |   |            |
// JP Adressierung Stackpointer                                          |   |            |
*E190: 8E 02 87       '...'             ZE190 LDS     #M0287             |   |            |        // SP = 0x0287
*E193: 20 DD          ' .'                    BRA     ZE172              |   |    >---/       |        // JR 0xe172
                                                                         |   |
// JP Berechnung Sprünge via IX; Übergabe A (A von 0x90 bis 0xb3); Rückgabe A -= 0x90                  // Entsprechend Taste-Code?
*E195: 80 90          '..'              ZE195 SUBA    #$90          <---/   |                |        // A -= 0x90
*E197: 16             '.'                     TAB                           |                |        // B = A
*E198: CE E1 A3       '...'                   LDX     #ME1A3                |                |        // IX = 0xe1a3; Sprungverteiler
*E19B: 3A             ':'                     ABX                           |                |        // IX += B
*E19C: 3A             ':'                     ABX                           |                |        // IX += B; IX += 2*B
```

```
*E19D: EE 00          '..'                    LDX     ,X                          |             // IX = (IX); Lade aus Adresse 0xe1a3+2B die Ansprungadrsse
*E19F: AD 00          '..'                    JSR     ,X                          |             // Call IX + 2*B -> entsprechend Sprungverteiler
*E1A1: 20 BB          ' .'                    BRA     ZE15E                    >---/             // JR 0xe15e


// Sprungtabelle maximal 45 Adressen
*E1A3: E6 BF          '..'            ME1A3                                                     // No.: 1
*E1A5: E6 F1          '..'                                                                      // No.: 2
*E1A7: E7 2F          './'                                                                      // No.: 3
*E1A9: E7 4D          '.M'                                                                      // No.: 4
*E1AB: E2 6F          '.o'                                                                      // No.: 5
*E1AD: E2 7B          '.{'                                                                      // No.: 6; Abfrage Taste "TEXT"
*E1AF: E2 E8          '..'                                                                      // No.: 7
*E1B1: E4 16          '..'                                                                      // No.: 8
*E1B3: E4 6F          '.o'                                                                      // No.: 9
*E1B5: E2 83          '..'                                                                      // No.: 10
*E1B7: E3 0D          '..'                                                                      // No.: 11
*E1B9: E3 AA          '..'                                                                      // No.: 12
*E1BB: E3 FB          '..'                                                                      // No.: 13
*E1BD: E4 E6          '..'                                                                      // No.: 14
*E1BF: F5 9E                                                                                    // No.: 15
*E1C1: F6 B2          '...'                                                                     // No.: 16
*E1C3: EA 94          '...'                                                                     // No.: 17
*E1C5: EA 94          '..'                                                                      // No.: 18
*E1C7: E8 53          '.S'                                                                      // No.: 19
*E1C9: EE D3          '..'                                                                      // No.: 20
*E1CB: E7 77          '.w'                                                                      // No.: 21
*E1CD: E7 92          '..'                                                                      // No.: 22
*E1CF: E7 B0          '..'                                                                      // No.: 23
*E1D1: E7 73          '.s'                                                                      // No.: 24
*E1D3: E7 D1          '..'                                                                      // No.: 25
*E1D5: E8 45          '.E'                                                                      // No.: 26
*E1D7: E8 1E          '..'                                                                      // No.: 27
*E1D9: FA 20                                                                                    // No.: 28
*E1DB: FC 93          '. .'                                                                     // No.: 29
*E1DD: FD 08          '.'                                                                       // No.: 30
*E1DF: EE CD          '..'                                                                      // No.: 31
*E1E1: E4 0A          '..'                                                                      // No.: 32
*E1E3: F0 01                                                                                    // No.: 33
*E1E5: EE DD       '...'                                                                        // No.: 34 -> Daten inkonsistent 0xEE9D
*E1E7: F0 06             '.'                                                                    // No.: 35
*E1E9: F3 77                                                                                    // No.: 36

// Call Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E1EB: CE 02 88                       ZE1EB   LDX     #M0288                                    // IX = 0x0288
*E1EE: DF 73                                  STX     M0073                                     // (0x0073) = IX; Zeiger auf 0x0288
*E1F0: 96 2F          's./'                   LDAA    M002f                                     // A = (0x002f); Zählervariable 0x1 ... 0x62
*E1F2: 97 6E          '.n'                    STAA    M006E                                     // (0x006e) = A
*E1F4: CE 01 90       '...'                   LDX     #M0190                                    // IX = 0x0190
*E1F7: 7A 00 6E       'z.n'           ZE1F7   DEC     >M006E <--\                               // (0x006e)--; Prüfen Zähler == 1
*E1FA: 27 0C          ''.'                    BEQ     ZE208     | >----\                        // if (z) == 1; Abbruch Zähler war 1
*E1FC: EC 00          '..'                    LDD     ,X        |      |                        // D = (0x0190)
*E1FE: 84 7F          '..'                    ANDA    #$7F      |      |                        // A &= 0x7f Maskiere; A = (0x0190), B = (0x0191)
*E200: D3 73          '.s'                    ADDD    M0073     |      |                        // D += 0x0288
*E202: DD 73          '.s'                    STD     M0073     |      |                        // (0x0073) = D
*E204: 08             '.'                     INX               |      |                        // IX++
*E205: 08             '.'                     INX               |      |                        // IX++; Adressen 0x0190 + n*2
*E206: 20 EF          ' .'                    BRA     ZE1F7 >---/      |                        // JR 0xe1f7; Schleife (0x0063) mal bzw. (0x002f)
// Abbruch                                                            |
*E208: DE 73          '.s'            ZE208   LDX     M0073       <----/                        // IX = (0x0073) Zeiger auf 0x0288 + n * (0x0190 + 0x0288)
*E20A: DC 73          '.s'                    LDD     M0073                                     // D = (0x0073)
*E20C: 39             '9'                     RTS                                               // RETURN

// Call Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E20D: 8D 03          '..'            ZE20D   BSR     ZE212                                     // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E20F: EE 00          '..'                    LDX     ,X                                        // IX = (IX)
*E211: 39             '9'                     RTS                                               // RETURN

// Call Adressberechnung Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E212: D6 2F          './'            ZE212   LDAB    M002F                                     // B = (0x002f); Zählervariable 0x1 ... 0x62
*E214: CE 01 8E       '...'                   LDX     #M018E                                    // IX = 0x018e
*E217: 3A             ':'                     ABX                                               // IX += B
*E218: 3A             ':'                     ABX                                               // IX += B
*E219: 39             '9'                     RTS                                               // RETURN

// Call Init Textlänge Zeiger; Übergabe (0x002f) = 0x1
*E21A: 8D F1          '..'            ZE21A   BSR     ZE20D                                     // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E21C: 26 12          '&.'                    BNE     ZE230 >-----\                             // if (z) == 0
*E21E: 8D CB          '..'                    BSR     ZE1EB       |                             // Call 0xe1eb; Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E220: DF 25          '.%'                    STX     M0025       |                             // (0x0025) = IX; Zeiger Textende
*E222: DF 27          '.''                    STX     M0027       |                             // (0x0027) = IX
*E224: DF 29          '.)'                    STX     M0029       |                             // (0x0029) = IX
*E226: DF 2B          '.+'                    STX     M002B       |                             // (0x002b) = IX; 4 Zeiger Text-Start-Ende-Länge-Position
*E228: 4F             'O'                     CLRA                |                             // A = 0
*E229: 97 34          '.4'                    STAA    M0034       |                             // (0x0034) = A
*E22B: 97 30          '.0'                    STAA    M0030       |                             // (0x0030) = A
*E22D: DF 2D          '.-'                    STX     M002D       |                             // (0x002d) = IX; Zeiger Text-Start-Ende-Länge
*E22F: 39             '9'                     RTS                 |                             // RETURN
                                                                  |
// JP                                                             |
*E230: 8D B9          '..'            ZE230   BSR     ZE1EB <-----/                             // Call 0xe1eb; Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E232: 08             '.'                     INX                                               // IX++
*E233: DF 25          '.%'                    STX     M0025                                     // (0x0025) = IX; Zeiger Text-Ende
*E235: DF 2D          '.-'                    STX     M002D                                     // (0x002d) = IX; Zeiger Text-Start
*E237: BD E2 12       '...'                   JSR     ZE212                                     // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
```

```
*E23A: EC 00          '..'                    LDD     ,X                      // D = (IX)
*E23C: 83 00 02       '...'                   SUBD    #M0002                  // D -= 0x0002
*E23F: 84 7F          '..'                    ANDA    #$7F                    // A &= 0x7f Maskiere
*E241: D3 25          '.%'                    ADDD    M0025                   // D += (0x0025); Zeiger Text-Ende
*E243: DD 27          '.''                    STD     M0027                   // (0x0027) = D
*E245: 7F 00 30       '..0'                   CLR     >M0030                  // (0x0030) = 0
*E248: BD E5 BC       '...'                   JSR     ZE5BC                   // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E24B: 7E E3 E7       '~..'                   JMP     ZE3E7                   // JP 0xe3e7

// A++ Bereich: 0x01 bis 0x063
*E24E: 96 2F          './'            ZE24E   LDAA    M002F                   // A = (0x002f); Zählervariable 0x1 ... 0x62
*E250: 4C             'L'                     INCA                            // A++
*E251: 81 63          '.c'                    CMPA    #$63                    // CMP A, 0x63
*E253: 23 02          '#.'                    BLS     ZE257 >----\            // if (c) | (z) == 1
*E255: 86 01          '..'                    LDAA    #$01       |            // A = 0x01
*E257: 97 2F          './'            ZE257   STAA    M002F <----/            // (0x002f) = A; Zählervariable 0x1 ... 0x62
*E259: 39             '9'                     RTS                             // RETURN

// Call Dekrement (0x002f), == 0 Return A = 0x63 (0x002f)
*E25A: 7A 00 2F       'z./'           ZE25A   DEC     >M002F                  // (0x002f)--; Zählervariable 0x1 ... 0x62
*E25D: 26 04          '&.'                    BNE     ZE263 >---\             // if (z) == 0 Return
*E25F: 86 63          '.c'                    LDAA    #$63       |            // A = 0x63
*E261: 97 2F          './'                    STAA    M002F      |            // (0x002f) = A; Zählervariable 0x1 ... 0x62
*E263: 39             '9'             ZE263   RTS             <---/           // RETURN

// Call Rückgabe Z-Flag == 1 wenn Taste "TEXT"
*E264: 8D B4          '..'            ZE264   BSR     ZE21A                   // Call 0xe21a  Init Textlänge Zeiger
*E266: BD EC A2       '...'                   JSR     ZECA2                   // Call 0xeca2
*E269: BD F3 F9       '...'                   JSR     ZF3F9                   // Call 0xf3f9; Tastendruck; Rückgabe A
*E26C: 81 96          '..'                    CMPA    #$96                    // CMP A, 0x96; Taste "TEXT"
*E26E: 39             '9'                     RTS                             // RETURN

// Call Sprungverteiler 5; Übergabe A; Rückgabe A Taste (TEXT)
*E26F: 8D DD          '..'            ZE26F   BSR     ZE24E <---\             // Call 0xe24e; A++ Bereich: 0x01 bis 0x063
*E271: 8D F1          '..'                    BSR     ZE264      |            // Call 0xe264; Rückgabe Z-Flag == 1 wenn Taste "TEXT"
*E273: 27 FA          ''.'                    BEQ     ZE26F >---/             // if (z) == 1
*E275: 7F 00 37       '..7'           ZE275   CLR     >M0037     <---\        // (0x0037) = 0
*E278: 7E E1 72       '~.r'                   JMP     ZE172           |       // JP 0xe172 Sprungverteiler
                                                                      |
// Call Sprungverteiler 6 Abfrage Taste "TEXT"                        |
*E27B: 8D DD          '..'            ZE27B   BSR     ZE25A <----\    |       // Call 0xe25a; Dekrement (0x002f), == 0 Return A = 0x63 (0x002f)
*E27D: 8D E5          '..'                    BSR     ZE264      |    |       // Call 0xe264; Rückgabe Z-Flag == 1 wenn Taste "TEXT"
*E27F: 27 FA          ''.'                    BEQ     ZE27B >----/    |       // if (z) == 1 Schleife
*E281: 20 F2          ' .'                    BRA     ZE275      >---/        // JR 0xe275


// Call Sprungverteiler 10, Abfrage "DELETE TEXT?"
*E283: BD E2 0D       '...'                   JSR     ZE20D                   // Call 0xe20d; Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E286: 26 03          '&.'                    BNE     ZE28B >---\             // if (z) == 0
*E288: 7E EA 94       '~..'                   JMP     ZEA94      |            // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E28B: CE F9 35       '..5'           ZE28B   LDX     #MF935 >--/             // IX = 0xF935; "DELETE TEXT  .. ? "
*E28E: C6 0C          '..'                    LDAB    #$0C                    // B = 0x0c
*E290: 96 2F          './'                    LDAA    M002F                   // A = (0x002f); Text Adresse 0x01 ... 0x62
*E292: BD EC CF       '...'                   JSR     ZECCF                   // Call 0xeccf; Übergabe D = 0x030c ... 0x630c, IX AusgabeText; Rückgabe D (0x0066)
*E295: BD F3 F9       '...'                   JSR     ZF3F9                   // Call 0xf3f9; Tastendruck; Rückgabe A
*E298: 88 99          '..'                    EORA    #$99                    // A XOR= 0x99 Taste "DEL TXT"
*E29A: 27 05          ''.'                    BEQ     ZE2A1 >---\             // if (z) == 1
*E29C: 88 0E          '..'                    EORA    #$0E       |            // A XOR= 0x0e
*E29E: 27 01          ''.'                    BEQ     ZE2A1 >----+            // if (z) == 1
*E2A0: 39             '9'                     RTS                |            // RETURN
                                                                 |
// Weiter aus 0xe39e Abfrage "DELETE TEXT?"                       |
*E2A1: 97 37          '.7'            ZE2A1   STAA    M0037 <---/             // (0x0037) = A
*E2A3: DE 25          '.%'                    LDX     M0025                   // IX = (0x0025); Zeiger Text-Ende
*E2A5: 09             '.'                     DEX                             // IX--
*E2A6: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX; Zeiger Text-Start
*E2A8: 7C 00 2F       '|./'                   INC     >M002F                  // (0x002f)++; Zählervariable 0x1 ... 0x62
*E2AB: BD E1 EB       '...'                   JSR     ZE1EB                   // Call 0xe1eb; Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E2AE: 7A 00 2F       'z./'                   DEC     >M002F                  // (0x002f)--; Zählervariable 0x1 ... 0x62
*E2B1: 93 2D          '.-'                    SUBD    M002D                   // D -= (0x002d); Zeiger Text-Start-Ende-Länge
*E2B3: BD EA 73       '..s'                   JSR     ZEA73                   // Call 0xea73; Übergabe D
*E2B6: BD E2 12       '...'                   JSR     ZE212                   // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E2B9: 6F 00          'o.'                    CLR     ,X                      // (IX) = 0
*E2BB: 6F 01          'o.'                    CLR     $01,X                   // (IX + 0x01) = 0
*E2BD: 7E E2 1A       '~..'                   JMP     ZE21A                   // JP 0xe21a -> mit Return Init Textlänge Zeiger

// Call Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*E2C0: BD E1 EB       '...'           ZE2C0   JSR     ZE1EB                   // Call 0xe1eb;  Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E2C3: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX; Zeiger Text-Start-Ende-Länge
*E2C5: C6 02          '..'                    LDAB    #$02                    // B = 0x02
*E2C7: BD EA 1E       '...'                   JSR     ZEA1E                   // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E2CA: DE 2D          '.-'                    LDX     M002D                   // IX = (0x002d); Zeiger Text-Start-Ende-Länge
*E2CC: 96 3D          '.='                    LDAA    M003D                   // A = (0x003d)
*E2CE: 97 3C          '.<'                    STAA    M003C                   // (0x003c) = A
*E2D0: A7 00          '..'                    STAA    ,X                      // (IX) = A; 0x002d = (0x003c) = (0x003d)
*E2D2: 08             '.'                     INX                             // IX++; 0x002e
*E2D3: 86 8D          '..'                    LDAA    #$8D                    // A = 0x8d (codiertes wr/zv 0x0d)
*E2D5: A7 00          '..'                    STAA    ,X                      // (IX) = A; (0x002e) = 0x8d
*E2D7: DF 25          '.%'                    STX     M0025                   // (0x0025) = IX = 0x002e; Zeiger Text-Ende
*E2D9: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX = 0x002e; Zeiger Text-Start
*E2DB: DF 27          '.''                    STX     M0027                   // (0x0027) = IX = 0x002e
*E2DD: BD E2 12       '...'                   JSR     ZE212                   // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E2E0: CC 00 02       '...'                   LDD     #M0002                  // D = 0x0002
*E2E3: ED 00          '..'                    STD     ,X                      // (IX) = D
*E2E5: 7E E5 BC       '~..'                   JMP     ZE5BC                   // JP 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b) mit Return
```

```
// Call Sprungverteiler 7; Clear Text; Ausgabe Freier Speicherplatz
*E2E8: 7F 00 37       '..7'                   CLR     >M0037                          // (0x0037) = 0
*E2EB: 8D 06          '..'                    BSR     ZE2F3 >---\                     // Call 0xe2f3; Prüfung Textspeicher verfügbar
*E2ED: BD E2 1A       '...'                   JSR     ZE21A     |                     // Call 0xe21a  Init Textlänge Zeiger
*E2F0: 7E ED EC       '~..'                   JMP     ZEDEC     |                     // JP 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe
                                                                |
// Call Prüfung Textspeicher verfügbar                          |
*E2F3: 96 2F          './'            ZE2F3   LDAA    M002F <---/                     // A = (0x002f); Zählervariable 0x1 ... 0x62
*E2F5: 97 72          '.r'                    STAA    M0072                           // (0x0072) = A
*E2F7: BD E2 4E       '..N'           ZE2F7   JSR     ZE24E <---\                     // Call 0xe24e; A++ Bereich: 0x01 bis 0x063
*E2FA: 91 72          '.r'                    CMPA    M0072     |                     // CMP A, (0x0072); Längenvergleich, max 0x63 = 99 bytes
*E2FC: 27 06          ''.'                    BEQ     ZE304     | >---\               // if (z) == 1; "No Free Text Variable"
*E2FE: BD E2 0D       '...'                   JSR     ZE20D     |     |               // Call 0xe20d; Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E301: 26 F4          '&.'                    BNE     ZE2F7 >---/     |               // if (z) == 0
*E303: 39             '9'                     RTS                     |               // RETURN
// Weiter-Abbruch                                                     |
*E304: CE F8 ED       '...'           ZE304   LDX     #ZF8ED      <---/               // IX = 0xf8ed; "No Free Text Vaiable... "
*E307: BD EC C2       '...'                   JSR     ZECC2                           // Call 0xecc2;  Ausgabe Fehler Text auf LCD; Übergabe IX
*E30A: 7E E1 5E       '~.^'                   JMP     ZE15E                           // JP 0xe15e

// Call Sprungverteiler 11; Clear all Text, Prüfe Tastendruck "Clear All"
*E30D: CE F8 DC       '...'                   LDX     #MF8DC                          // IX = 0xf8dc; "CLEAR ALL TEXT ? "
*E310: BD EC CA       '...'                   JSR     ZECCA                           // Call 0xecca; Textausgabe auf LCD, Warten auf Tastendruck; Übergabe IX
*E313: 81 9A          '..'                    CMPA    #$9A                            // CMP A, 0x9a; Taste "CLEAR ALL"
*E315: 27 01          ''.'                    BEQ     ZE318 >---\                     // if (z) == 1
*E317: 39             '9'             ZE317   RTS                 |   <---\        // RETURN
                                                                    |       |
// JP                                                               |       |
*E318: BD EA 99       '...'           ZE318   JSR     ZEA99 <---/         |       // Call 0xea99; Modem ON Übergabe A Port-1x, B Zeitschleife 60-100-60ms
*E31B: CE E3 2A       '..*'                   LDX     #ME32A              |       // IX = 0xe32a; 'ARE YOU SURE ? '
*E31E: BD EC CA       '...'                   JSR     ZECCA               |       // Call 0xecca; Textausgabe auf LCD, Warten auf Tastendruck; Übergabe IX
*E321: 81 9A          '..'                    CMPA    #$9A                |       // CMP A, 0x9a; Taste "Clear All"
*E323: 26 F2          '&.'                    BNE     ZE317           >---/       // if (z) == 0 RETURN
*E325: 8D 35          '.5'                    BSR     ZE35C                       // Call 0xe35c; "Clear All" Lösche und Fülle RAM mit Initalwerten
*E327: 7E E1 5E       '~.^'                   JMP     ZE15E                       // JP 0xe15e

// Data
*E32A: 41 52 45 20 59 4F 55 20        'ARE YOU SURE ? '             ME32A         // "ARE YOU SURE ? "
*E232: 53 55 52 45 20 3F A0

// JP Einsprung von 0xe000
*E339: 8D 0E          '..'            ZE339   BSR     ZE349 >---\                 // Warte auf Funktionstaste
                                                                |
                                                                |
// JP Einsprung aus LCD Initialisierung                         |
*E33B: 8D 0C          '..'            ZE33B   BSR     ZE349 >---+                 // Call 0xe349; Warte auf Funktionstaste?
*E33D: 8D 0A          '..'                    BSR     ZE349 >---+                 // Call 0xe349; Warte auf Funktionstaste?
*E33F: 7E F7 4A       '~.J'                   JMP     ZF74A     |                 // JP 0xf74a; TimerCSR auf 0, Warte auf Interrupt
                                                                |
*E342: 8D 18          '..'            ZE342   BSR     ZE35C     | >---\           // Call 0xe35c; "Clear All" Lösche und Fülle RAM mit Initalwerten
*E344: 8D 03          '..'                    BSR     ZE349 >---+     |           // Call 0xe349; Zeitschleifen Tastatur
*E346: 7E E1 5E       '~.^'                   JMP     ZE15E     |     |           // JP 0xe15e; Sprungverteiler
                                                                |     |
// Call LED ON/OFF                                              |     |
*E349: BD EA 94       '...'           ZE349   JSR     ZEA94 <---/     |           // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E34C: 71 7F 02       'q..'                   AIM     #$7F,M0002      |           // (Port-1 Data) &= 0x7f; Port-17 LED OFF
*E34F: C6 32          '.2'                    LDAB    #$32            |           // B = 0x32
*E351: BD EA CF       '...'                   JSR     ZEACF           |           // Call 0xeacf; Zeitschleife 10ms; 50mal
*E354: 72 80 02       'r..'                   OIM     #$80,M0002      |           // (0x0002) |= 0x80; Setze Bit7 Port-17 LED ON
*E357: C6 32          '.2'                    LDAB    #$32            |           // B = 0x32
*E359: 7E EA CF       '~..'                   JMP     ZEACF           |           // JP 0xeacf; Zeitschleife 10ms; 50mal
                                                                      |
// Call "Clear All" Lösche und Fülle RAM mit Initalwerten            |
*E35C: CE 01 90       '...'           ZE35C   LDX     #M0190      <---/           // IX = 0x0190
*E35F: 6F 00          'o.'            ZE35F   CLR     ,X    <---\                 // (IX) = 0
*E361: 08             '.'                     INX             |                   // IX++; 0x0191 ... 0x0255
*E362: 8C 02 56       '..V'                   CPX     #M0256    |                 // cmp IX, 0x0256
*E365: 26 F8          '&.'                    BNE     ZE35F >---/                 // if (z) == 0 Schleife Lösche (0x0190 ... 0x0255)
*E367: 86 28          '.('                    LDAA    #$28                        // A = 0x28
*E369: 97 3C          '.<'                    STAA    M003C                       // (0x003c) = A
*E36B: 97 3D          '.='                    STAA    M003D                       // (0x003d) = A
*E36D: CE 01 61       '..a'                   LDX     #M0161                      // IX = 0x0161
*E370: 86 30          '.0'                    LDAA    #$30                        // A = 0x30; "0"
*E372: C6 08          '..'                    LDAB    #$08                        // B = 0x08 Schleifenzähler
*E374: A7 00          '..'            ZE374   STAA    ,X    <----\                // (IX) = A
*E376: 4C             'L'                     INCA            |                   // A++
*E377: 08             '.'             ZE377   INX             |                   // IX++
*E378: 5A             'Z'                     DECB            |                   // B--
*E379: 26 F9          '&.'                    BNE     ZE374 >----/                // if (z) == 0 Schleife 8 Runden; Fülle 0x0161 bis 0x0168 mit "0" ... "7"
*E37B: CE 00 23       '..#'                   LDX     #M0023                      // IX = 0x0023
*E37E: CC 02 87       '...'                   LDD     #M0287                      // D = 0x0287
*E381: ED 00          '..'            ZE381   STD     ,X    <---\                 // (IX) = D
*E383: 08             '.'                     INX             |                   // IX++; 0x0024 ...
*E384: 08             '.'                     INX             |                   // IX++; 0x0025 ... 0x002d
*E385: 8C 00 2D       '..-'                   CPX     #M002D    |                 // cmp IX, 0x002d; Zeiger Text-Start-Ende-Länge
*E388: 23 F7          '#.'            ZE388   BLS     ZE381 >---/                 // if (c) | (z) == 1 Schleife Fülle 0x0023 ... 0x002c mit 0x0287
*E38A: 6F 00          'o.'            ZE38A   CLR     ,X    <---\                 // (IX) = 0; IX = 0x002d
*E38C: 08             '.'                     INX             |                   // IX++; 0x002e
*E38D: 8C 00 39       '..9'                   CPX     #M0039    |                 // cmp IX, 0x0039
*E390: 23 F8          '#.'                    BLS     ZE38A >---/                 // if (c) | (z) == 1 Schleife; Fülle 0x02e ... 0x0038 mit 0x0
*E392: BD E8 1E       '...'                   JSR     ZE81E                       // Call 0xe81e; Sprungverteiler 27; Fülle Speicherbereich 0x0100 ... 0x014f mit 0x0 bzw. 0xff
*E395: CE 00 46       '..F'                   LDX     #M0046                      // IX = 0x0046
*E398: 6F 00          'o.'            ZE398   CLR     ,X    <---\                 // (IX) = 0
*E39A: 08             '.'                     INX             |                   // IX++
```

```
*E39B: 8C 00 4E       '..N'                   CPX     #M004E    |                          // cmp IX, 0x004e
*E39E: 26 F8          '&.'                    BNE     ZE398 >---/                        // if (z) == 0 Schleife; Fülle 0x0046 bis 0x004d mit 0x0
*E3A0: 86 01          '..'                    LDAA    #$01                               // A = 0x01
*E3A2: 97 2F          './'                    STAA    M002F                              // (0x002f) = A; Zählervariable 0x1
*E3A4: 7E E2 1A       '~..'                   JMP     ZE21A                              // JP 0xe21a mit RETURN Init Textlänge Zeiger

// JP Sprung in den Verteiler Nr. 17
*E3A7: BD EA 94       '...'             ZE3A7 JSR     ZEA94 <---\                        // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                             |
// Call Sprungverteiler 12; SET MARGIN AT ...                |
*E3AA: BD EA DF       '...'                   JSR     ZEADF     |                        // Call 0xeadf; IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E3AD: 27 64          ''d'                    BEQ     ZE413     |  >---\                 // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E3AF: CE F9 6E       '..n'                   LDX     #MF96E    |      |                 // IX = 0xf96e; "SET RIGHT MARGIN AT .. "
*E3B2: C6 14          '..'                    LDAB    #$14      |      |                 // B = 0x14
*E3B4: 96 3C          '.<'                    LDAA    M003C     |      |                 // A = (0x003c)
*E3B6: BD EC CF       '...'                   JSR     ZECCF     |      |                 // Call 0xeccf; Übergabe D = 0x0314 ... 0x6314, IX = AusgabeText; Rückgabe D (0x0066)
*E3B9: CE 9B 9B       '...'                   LDX     #M9B9B    |      |                 // IX = 0x9b9b; Zeichenfolge
*E3BC: DF 5A          '.Z'                    STX     M005A     |      |                 // (0x005a) = IX
*E3BE: BD EA F5       '...'                   JSR     ZEAF5     |      |                 // Call 0xeaf5; LCD Ausgaben, Übergabe B
*E3C1: 81 0A          '..'                    CMPA    #$0A      |      |                 // CMP A, 0x0a
*E3C3: 25 E2          '%.'                    BCS     ZE3A7 >---+      |                 // if (c) == 1 -> Call 0xea94 weiter Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E3C5: 81 50          '.P'                    CMPA    #$50      |      |                 // CMP A, 0x50
*E3C7: 22 DE          '".'                    BHI     ZE3A7 >---/      |                 // if (c) & (z) == 0 -> Call 0xea94 weiter Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E3C9: 91 3C          '.<'                    CMPA    M003C            |                 // CMP A, (0x003c)
*E3CB: 27 19          ''.'                    BEQ     ZE3E6 >---\      |                 // if (z) == 1 RETURN
*E3CD: 97 3C          '.<'                    STAA    M003C     |      |                 // (0x003c) = A
*E3CF: 97 3D          '.='                    STAA    M003D     |      |                 // (0x003d) = A
*E3D1: BD E2 0D       '...'                   JSR     ZE20D     |      |                 // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E3D4: 27 10          ''.'                    BEQ     ZE3E6 >---+      |                 // if (z) == 1 RETURN
*E3D6: DE 25          '.%'                    LDX     M0025     |      |                 // IX = (0x0025); Zeiger Text-Ende
*E3D8: DF 29          '.)'                    STX     M0029     |      |                 // (0x0029) = IX
*E3DA: 09             '.'                     DEX               |      |                 // IX--
*E3DB: A7 00          '..'                    STAA    ,X        |      |                 // (IX) = A
*E3DD: BD EA 01       '...'                   JSR     ZEA01     |      |                 // Call 0xea01; Textanfang-Ende GTX gefunden?
*E3E0: 7F 00 31       '..1'                   CLR     >M0031    |      |                 // (0x0031) = 0
*E3E3: BD E7 6C       '..l'                   JSR     ZE76C     |      |                 // Call 0xe76c; (0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
*E3E6: 39             '9'               ZE3E6 RTS           <---/      |                 // RETURN
                                                                       |
// Call JP                                                             |
*E3E7: 3C             '<'               ZE3E7 PSHX                     |                 // Push IX
*E3E8: BD EA DF       '...'                   JSR     ZEADF            |                 // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E3EB: 27 0C          ''.'                    BEQ     ZE3F9  >---\     |                 // if (z) == 1 Pop IX  RETURN
*E3ED: BD E2 0D       '...'                   JSR     ZE20D      |     |                 // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E3F0: 27 07          ''.'                    BEQ     ZE3F9  >---+     |                 // if (z) == 1 Pop IX  RETURN
*E3F2: DE 25          '.%'                    LDX     M0025      |     |                 // IX = (0x0025); Zeiger Text-Ende
*E3F4: 09             '.'                     DEX                |     |                 // IX--
*E3F5: A6 00          '..'                    LDAA    ,X         |     |                 // A = (IX)
*E3F7: 97 3C          '.<'                    STAA    M003C      |     |                 // (0x003c) = A
// Call JP "Pop IX"                                          |     |
*E3F9: 38             '8'               ZE3F9 PULX          <---/     |                 // Pop IX
*E3FA: 39             '9'                     RTS                     |                 // RETURN
                                                                      |
// Call Sprungverteiler 13; Verlgeich (0x0027) mit(0x002d)->0xea94 Sprun|gverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E3FB: BD EA DF       '...'                   JSR     ZEADF            |                 // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E3FE: 27 13          ''.'                    BEQ     ZE413 >---\      |                 // if (z) == 1;-> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E400: DE 2D          '.-'                    LDX     M002D     |      |                 // IX = (0x002d)
*E402: 9C 27          '.''                    CPX     M0027     |      |                 // cmp IX, (0x0027)
*E404: 24 0D          '$.'                    BCC     ZE413 >---+      |                 // if (c) == 0; ->0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E406: 73 00 37       's.7'                   COM     >M0037    |      |                 // Not (0x0037)
*E409: 39             '9'                     RTS               |      |                 // RETURN
                                                                |      |
// Call Sprungverteiler 32                                      |      |
*E40A: BD EA DF       '...'                   JSR     ZEADF     |      |                 // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E40D: 27 04          ''.'                    BEQ     ZE413 >---+      |                 // if (z) == 1; ->0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E40F: 73 00 36       's.6'                   COM     >M0036    |      |                 // Not (0x0036); Tastatur z/s
*E412: 39             '9'                     RTS               |      |                 // RETURN
                                                                |      |
// JP                                                           |      |
*E413: 7E EA 94       '~..'             ZE413 JMP     ZEA94 <---+------/                 // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                             |
// Call Sprungverteiler 8                                    |
*E416: BD EA DF       '...'                   JSR     ZEADF     |                        // Call 0xeadf; IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E419: 27 F8          ''.'                    BEQ     ZE413 >---+                        // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E41B: BD E2 0D       '...'                   JSR     ZE20D     |                        // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E41E: 27 F3          ''.'                    BEQ     ZE413 >---+                        // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E420: DE 2D          '.-'                    LDX     M002D     |                        // IX = (0x002d); Zeiger Text-Start-Ende-Länge
*E422: 9C 25          '.%'                    CPX     M0025     |                        // cmp IX, (0x0025); Zeiger Text-Ende
*E424: 22 06          '".'                    BHI     ZE42C     |   >---\                // if (c) & (z) == 0
*E426: 9C 27          '.''                    CPX     M0027     |       |                // cmp IX, (0x0027)
*E428: 25 E9          '%.'                    BCS     ZE413 >---/       |                // if (c) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E42A: 20 37          ' 7'                    BRA     ZE463             |            >---\   // JR 0xe463
// JP                                                                   |                |
*E42C: D6 30          '.0'              ZE42C LDAB    M0030         <---/                |   // B = (0x0030)
*E42E: 27 0E          ''.'                    BEQ     ZE43E >---\                        |   // if (z) == 1
*E430: 5A             'Z'                     DECB              |                        |   // B--
*E431: D1 34          '.4'                    CMPB    M0034     |                        |   // cmp B, (0x0034)
*E433: 22 19          '".'                    BHI     ZE44E     | >---\                  |   // if (c) & (z) == 0
*E435: 26 07          '&.'                    BNE     ZE43E >---+     |                  |   // if (z) == 0
*E437: A6 00          '..'                    LDAA    ,X        |     |                  |   // A = (IX)
*E439: 81 8D          '..'                    CMPA    #$8D      |     |                  |   // CMP A, 0x8d
*E43B: 27 11          ''.'                    BEQ     ZE44E     | >---+                  |   // if (z) == 1
*E43D: 08             '.'                     INX               |     |                  |   // IX++
*E43E: 09             '.'               ZE43E DEX           <---/     |                  |   // IX--
*E43F: DF 2D          '.-'                    STX     M002D           |                  |   // (0x002d) = IX; Zeiger Text-Start-Ende-Länge
*E441: 7A 00 30       'z.0'                   DEC     >M0030          |                  |   // (0x0030)--
```

```
*E444: C6 01          '..'                    LDAB    #$01           |                  |       |       // B = 0x01
*E446: BD EA 63       '..c'                   JSR     ZEA63          |                  |       |       // Call 0xea63 Übergabe D
*E449: BD E6 65       '..e'                   JSR     ZE665          |                  |       |       // Call 0xe665
*E44C: 20 0B          ' .'                    BRA     ZE459 >---\    |                  |       |       // JR 0xe459
// JP                                                             |    |                  |       |
*E44E: D6 34          '.4'            ZE44E   LDAB    M0034     | <---/                  |       |       // B = (0x0034)
*E450: A6 00          '..'                    LDAA    ,X        |                        |       |       // A = (IX)
*E452: 81 8D          '..'                    CMPA    #$8D      |                        |       |       // CMP A, 0x8d
*E454: 27 01          ''.'                    BEQ     ZE457     | >---\                  |       |       // if (z) == 1
*E456: 5C             '\'                     INCB              |     |                  |       |       // B++
*E457: D7 30          '.0'            ZE457   STAB    M0030     | <---/                  |       |       // (0x0030) = B
*E459: 86 FF          '..'            ZE459   LDAA    #$FF  <---/                        |       |       // A = 0xff
*E45B: 97 38          '.8'                    STAA    M0038                              |       |       // (0x0038) = A
*E45D: DE 27          '.''                    LDX     M0027                              |       |       // IX = (0x0027)
*E45F: 9C 25          '.%'                    CPX     M0025                              |       |       // cmp IX, (0x0025); Zeiger Text-Ende
*E461: 22 0B          '".'                    BHI     ZE46E >---\                        |       |       // if (c) & (z) == 0 RETURN
// JP                                                           |                        |       |
*E463: BD E2 12       '...'           ZE463   JSR     ZE212     |                  <---+       |       // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E466: CC 00 02       '...'                   LDD     #M0002    |                              |       // D = 0x0002
*E469: ED 00          '..'                    STD     ,X        |                              |       // (IX) = D
*E46B: 7E E2 A1       '~..'                   JMP     ZE2A1     |                              |       // JP 0xe2a1
*E46E: 39             '9'             ZE46E   RTS           <---/                              |       // RETURN
                                                                                               |
// Call JP Sprungverteiler 9                                 ^                                  |
*E46F: BD E2 0D       '...'           ZE46F   JSR     ZE20D     /                        | <-\   |       // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E472: 27 9F          ''.'                    BEQ     ZE413 >---+                        |   |   |       // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E474: BD EA DF       '...'                   JSR     ZEADF     |                        |   |   |       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E477: 27 9A          ''.'                    BEQ     ZE413 >---/                        |   |   |       // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E479: 7D 00 30       '}.0'                   TST     >M0030                             |   |   |       // (0x0030) == 0?; MSB == 1 n-Flag (0x0030) == 0 z-Flag
*E47C: 26 15          '&.'                    BNE     ZE493 >---\                        |   |   |       // if (z) == 0
*E47E: DE 2B          '.+'                    LDX     M002B     |                        |   |   |       // IX = (0x002b); Zeiger Text-aktual Position
*E480: 9C 27          '.''                    CPX     M0027     |                        |   |   |       // cmp IX, (0x0027); Zeiger Text-aktual Position
*E482: 25 34          '%4'                    BCS     ZE4B8     |       >---\            |   |   |       // if (c) == 1
*E484: DE 29          '.)'                    LDX     M0029     |           |            |   |   |       // IX = (0x0029)
*E486: 9C 25          '.%'                    CPX     M0025     |           |            |   |   |       // cmp IX, (0x0025); Zeiger Text-Ende
*E488: 23 1A          '#.'                    BLS     ZE4A4     | >---\     |            |   |   |       // if (c) | (z) == 1
*E48A: 09             '.'                     DEX               |     |     |            |   |   |       // IX--
*E48B: A6 00          '..'                    LDAA    ,X        |     |     |            |   |   |       // A = (IX)
*E48D: 81 8D          '..'                    CMPA    #$8D      |     |     |            |   |   |       // CMP A, 0x8d
*E48F: 26 13          '&.'                    BNE     ZE4A4     | >---+     |            |   |   |       // if (z) == 0
*E491: 20 25          ' %'                    BRA     ZE4B8     |     | >---+            |   |   |       // JR 0xe4b8
// JP                                                           |     |     |            |   |
*E493: 96 30          '.0'            ZE493   LDAA    M0030 <---/     |     |            |   |   |       // A = (0x0030)
*E495: 91 34          '.4'                    CMPA    M0034           |     |            |   |   |       // CMP A, (0x0034)
*E497: 25 0B          '%.'                    BCS     ZE4A4       >---+     |            |   |   |       // if (c) == 1
*E499: DE 2B          '.+'                    LDX     M002B           |     |            |   |   |       // IX = (0x002b); Zeiger Text-aktual Position
*E49B: 9C 27          '.''                    CPX     M0027           |     |            |   |   |       // cmp IX, (0x0027); Zeiger Text-aktual Position
*E49D: 24 4C          '$L'                    BCC     ZE4EB           |     | >---\      |   |   |       // if (c) == 0; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E49F: 08             '.'                     INX                     |     |     |      |   |   |       // IX++
*E4A0: DF 2D          '.-'                    STX     M002D           |     |     |      |   |   |       // (0x002d) = IX; Zeiger Text-Start-Ende-Länge
*E4A2: 20 24          ' $'                    BRA     ZE4C8 >---\     |     |     |      |   |   |       // JR 0xe4c8
// JP                                                           |     |     |     |      |   |
*E4A4: DC 2B          '.+'            ZE4A4   LDD     M002B     | <---/     |     |      |   |   |       // D = (0x002b); Zeiger Text-aktual Position
*E4A6: 93 2D          '.-'                    SUBD    M002D     |           |     |      |   |   |       // D -= (0x002d); Zeiger Text-Start
*E4A8: 27 03          ''.'                    BEQ     ZE4AD     | >---\     |     |      |   |   |       // if (z) == 1
*E4AA: BD EA 63       '..c'                   JSR     ZEA63     |     |     |     |      |   |   |       // Call 0xea63 Übergabe D
*E4AD: DE 2D          '.-'            ZE4AD   LDX     M002D     | >---/     |     |      |   |   |       // IX = (0x002d); Zeiger Text-Start-Ende-Länge
*E4AF: 9C 27          '.''                    CPX     M0027     |           |     |      |   |   |       // cmp IX, (0x0027)
*E4B1: 24 15          '$.'                    BCC     ZE4C8 >---+           |     |      |   |   |       // if (c) == 0
*E4B3: 08             '.'                     INX               |           |     |      |   |   |       // IX++
*E4B4: DF 2D          '.-'                    STX     M002D     |           |     |      |   |   |       // (0x002d) = IX
*E4B6: 20 10          ' .'                    BRA     ZE4C8 >---+           |     |      |   |   |       // JR 0xe4c8
// JP                                                           |           |     |      |   |
*E4B8: DC 2B          '.+'            ZE4B8   LDD     M002B     |       <---/     |      |   |   |       // D = (0x002b); Zeiger Text-aktual Position
*E4BA: 93 2D          '.-'                    SUBD    M002D     |                 |      |   |   |       // D -= (0x002D); Zeiger Text-Start
*E4BC: 5C             '\'                     INCB              |                 |      |   |   |       // B++; B = 1 + (0x002b) - (0x002d)
*E4BD: BD EA 63       '..c'                   JSR     ZEA63     |                 |      |   |   |       // Call 0xea63 Übergabe D
*E4C0: DE 27          '.''                    LDX     M0027     |                 |      |   |   |       // IX = (0x0027)
*E4C2: 9C 2D          '.-'                    CPX     M002D     |                 |      |   |   |       // cmp IX, (0x002d); Zeiger Text-Start-Ende-Länge
*E4C4: 24 02          '$.'                    BCC     ZE4C8 >---+                 |      |   |   |       // if (c) == 0
*E4C6: DF 2D          '.-'                    STX     M002D     |                 |      |   |   |       // (0x002d) = IX
*E4C8: DE 27          '.''            ZE4C8   LDX     M0027 <---/                 |      |   |   |       // IX = (0x0027)
*E4CA: 9C 25          '.%'                    CPX     M0025                       |      |   |   |       // cmp IX, (0x0025); Zeiger Text-Ende
*E4CC: 23 95          '#.'                    BLS     ZE463                       |   >----/   |   |       // if (c) | (z) == 1 springe
*E4CE: BD E5 BC       '...'                   JSR     ZE5BC                       |            |   |       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E4D1: BD E6 65       '..e'                   JSR     ZE665                       |            |   |       // Call 0xe665
*E4D4: 7F 00 31       '..1'                   CLR     >M0031                      |            |   |       // (0x0031) = 0
*E4D7: BD EC A2       '...'                   JSR     ZECA2                       |            |   |       // Call 0xeca2
*E4DA: BD F3 F9       '...'                   JSR     ZF3F9                       |            |   |       // Call 0xf3f9; Tastendruck; Rückgabe A
*E4DD: 81 9A          '..'                    CMPA    #$9A                        |            |   |       // CMP A, 0x9a; Taste CLEAR ALL
*E4DF: 26 02          '&.'                    BNE     ZE4E3 >---\                 |            |   |       // if (z) == 0
*E4E1: 20 8C          ' .'                    BRA     ZE46F     |                 |        >---/   |       // JR 0e46f
// JP                                                           |                 |                |
*E4E3: 7E E1 90       '~..'           ZE4E3   JMP     ZE190 <---/                 |                |       // JP 0xe190->  LD SP = 0x0287; JP 0xe172;
                                                                                 |                |
// Call Sprungverteiler 14  "INSERT TEXT 00 ? "                                  |                |
*E4E6: BD EA DF       '...'                   JSR     ZEADF                       |                |       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E4E9: 26 03          '&.'                    BNE     ZE4EE >---\                 |                |       // if (z) == 0 "INSERT TEXT 00 ? "
*E4EB: 7E EA 94       '~..'           ZE4EB   JMP     ZEA94     |             <---+                |       // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
// weiter                                                       |                                  |
*E4EE: CE F9 24       '..$'           ZE4EE   LDX     #MF924 <--+                                  |       // IX = 0xf924; "INSERT TEXT 00 ? "
*E4F1: BD EC EA       '...'                   JSR     ZECEA     |                                  |       // Call 0xecea; Textausgabe LCD, Übergabe IX
*E4F4: C6 0C          '..'                    LDAB    #$0C      |                                  |       // B = 0x0c
*E4F6: CE 9C 9D       '...'                   LDX     #M9C9D    |                                  |       // IX = 0x9c9d
*E4F9: DF 5A          '.Z'                    STX     M005A     |                                  |       // (0x005a) = IX
```

```
*E4FB: BD EA F5       '...'                   JSR     ZEAF5     |                 |               // Call 0xeaf5; LCD Ausgaben, Übergabe B
*E4FE: 4D             'M'                     TSTA              |                 |               // A == 0?; MSB == 1 n-Flag A == 0 z-Flag
*E4FF: 27 EA          ''.'                    BEQ     ZE4EB >---+                 |               // if (z) == 1 Schleife
*E501: D6 2F          './'                    LDAB    M002F     |                 |               // B = (0x002f); Zählervariable 0x1 ... 0x62
*E503: 11             '.'                     CBA               |                 |               // cmp A  B
*E504: 27 E5          ''.'                    BEQ     ZE4EB >---+                 |               // if (z) == 1; Schleife
*E506: D7 72          '.r'                    STAB    M0072     |                 |               // (0x0072) = B
*E508: 97 2F          './'                    STAA    M002F     |                 |               // (0x002f) = A; Zählervariable 0x1 ... 0x62
*E50A: BD E2 0D       '...'                   JSR     ZE20D     |                 |               // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E50D: DF 70          '.p'                    STX     M0070     |                 |               // (0x0070) = IX
*E50F: 26 06          '&.'                    BNE     ZE517     | >---\           |               // if (z) == 0; -> 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E511: D6 72          '.r'                    LDAB    M0072     |     |           |               // B = (0x0072)
*E513: D7 2F          './'                    STAB    M002F     |     |           |               // (0x002f) = B; Zählervariable 0x1 ... 0x62
*E515: 20 D4          ' .'                    BRA     ZE4EB >---/     |           |               // JR 0xe4eb; Schleife
// Abbruch                                                           |           |
*E517: BD EA DF       '...'           ZE517   JSR     ZEADF       <---/           |               // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
// Tausch Inhalt (0x002f) <> (0x0072); Übergabe A                                  |
*E51A: 36             '6'                     PSHA                              |               // Push A
*E51B: 96 2F          './'                    LDAA    M002F                     |               // A = (0x002f); Zählervariable 0x1 ... 0x62
*E51D: D6 72          '.r'                    LDAB    M0072                     |               // B = (0x0072)
*E51F: 97 72          '.r'                    STAA    M0072                     |               // (0x0072) = A
*E521: D7 2F          './'                    STAB    M002F                     |               // (0x002f) = B; Zählervariable 0x1 ... 0x62
*E523: 32             '2'                     PULA                              |               // Pop A
*E524: 4D             'M'                     TSTA                              |               // A == 0?; MSB == 1 n-Flag A == 0 z-Flag
*E525: 26 07          '&.'                    BNE     ZE52E >---\               |               // if (z) == 0 weiter
*E527: BD E2 0D       '...'                   JSR     ZE20D     |               |               // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E52A: 27 57          ''W'                    BEQ     ZE583     |  >-----\      |               // if (z) == 1 springe
*E52C: 20 BD          ' .'                    BRA     ZE4EB     |        |  >---/               // JR 0xe4eb
// weiter                                                        |        |
*E52E: BD E2 0D       '...'           ZE52E   JSR     ZE20D <---/        |                     // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E531: 27 50          ''P'                    BEQ     ZE583          >---+                     // if (z) == 1 springe
*E533: 8D 0D          '..'                    BSR     ZE542             |                     // Call 0xe542; Übergabe A
*E535: D6 72          '.r'                    LDAB    M0072             |                     // B = (0x0072)
*E537: 96 2F          './'                    LDAA    M002F             |                     // A = (0x002f); Zählervariable 0x1 ... 0x62
*E539: D7 2F          './'                    STAB    M002F             |                     // (0x002f) = B; Zählervariable 0x1 ... 0x62
*E53B: 97 72          '.r'                    STAA    M0072             |                     // (0x0072) = A
*E53D: 8D 7D          '.}'                    BSR     ZE5BC             |                     // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E53F: 7E EA 01       '~..'                   JMP     ZEA01             |                     // JP 0xea01; Textanfang-Ende GTX gefunden?
                                                                        |
// Call Übergabe                                                        |
*E542: DC 70          '.p'            ZE542   LDD     M0070             |                     // D = (0x0070)
*E544: 84 7F          '..'                    ANDA    #$7F              |                     // A &= 0x7f Maskiere ASCII
*E546: 83 00 01       '...'                   SUBD    #M0001            |                     // D -= 0x0002
*E549: DD 70          '.p'                    STD     M0070             |                     // (0x0070) = D
*E54B: DE 29          '.)'                    LDX     M0029             |                     // IX = (0x0029)
*E54D: DF 2D          '.-'                    STX     M002D             |                     // (0x002d) = IX
*E54F: 7F 00 30       '..0'                   CLR     >M0030            |                     // (0x0030) = 0
*E552: BD EA 26       '..&'                   JSR     ZEA26             |                     // Call 0xea26; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E555: 96 72          '.r'                    LDAA    M0072             |                     // A = (0x0072)
*E557: D6 2F          './'                    LDAB    M002F             |                     // B = (0x002f); Zählervariable 0x1 ... 0x62
*E559: 97 2F          './'                    STAA    M002F             |                     // (0x002f) = A; Zählervariable 0x1 ... 0x62
*E55B: D7 72          '.r'                    STAB    M0072             |                     // (0x0072) = B; Tausch der Speicherinhalte (0x002f) <> (0x0072)
*E55D: 11             '.'                     CBA                       |                     // cmp A  B
*E55E: 24 05          '$.'                    BCC     ZE565 >---\       |                     // if (c) == 0
*E560: BD E1 EB       '...'                   JSR     ZE1EB     |       |                     // Call  0xe1eb; Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E563: 20 05          ' .'                    BRA     ZE56A     | >---\ |       |                     // JR 0xe56a
// JP                                                           |     | |       |
*E565: BD E1 EB       '...'           ZE565   JSR     ZE1EB <---/     | |       |                     // Call  0xe1eb; Tabelle Zeiger auf Text 1 ... Text n; Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger 0x0073; auf D = n * (0x0190) + (0x0288)
*E568: D3 70          '.p'                    ADDD    M0070           | |       |                     // D += (0x0070)
*E56A: DD 73          '.s'            ZE56A   STD     M0073       <---/ |       |                     // (0x0073) = D
*E56C: DC 2D          '.-'                    LDD     M002D             |       |                     // D = (0x002d)
*E56E: D3 70          '.p'                    ADDD    M0070             |       |                     // D += (0x0070)
*E570: DD 70          '.p'                    STD     M0070             |       |                     // (0x0070) = D
*E572: 9F 6C          '.l'                    STS     M006C             |       |                     // (0x0067) = SP; Sichere Stackpointer
*E574: 9E 73          '.s'                    LDS     M0073             |       |                     // SP = (0x0073) Lade SP mit Zeiger von (0x0073)
*E576: DE 2D          '.-'                    LDX     M002D             |       |                     // IX = (0x002d)
*E578: 33             '3'             ZE578   PULB          <---\     |       |                     // Pop B; Lade B mit Wert aus Zeiger von (0x0073)
*E579: E7 00          '..'                    STAB    ,X          |     |       |                     // (IX) = B; lege byte ab
*E57B: 08             '.'                     INX                 |     |       |                     // IX++
*E57C: 9C 70          '.p'                    CPX     M0070       |     |       |                     // cmp IX, (0x0070)
*E57E: 25 F8          '%.'                    BCS     ZE578 >---/         |       |                     // if (c) == 1; Schleife
*E580: 9E 6C          '.l'                    LDS     M006C               |       |                     // SP = (0x006c); Steackpointer wiederherstellen
*E582: 39             '9'                     RTS                         |       |                     // RETURN
                                                                          |       |
// JP                                                                     |       |
*E583: BD E2 C0       '...'           ZE583   JSR     ZE2C0           <---/       |                     // Call 0xe3c0; Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*E586: BD E5 42       '..B'                   JSR     ZE542                       |                     // Call 0xe542; Übergabe A
*E589: BD E1 EB       '...'                   JSR     ZE1EB                       |                     // Call 0xe1eb;  Übergabe Zähler n (0x002f) Rückgabe IX = Zeiger0x0073; auf D = n * (0x0190) + (0x0288)
*E58C: A6 00          '..'                    LDAA    ,X                          |                     // A = (IX)
*E58E: DE 25          '.%'                    LDX     M0025                       |                     // IX = (0x0025); Zeiger Text-Ende
*E590: 09             '.'                     DEX                                 |                     // IX--
*E591: A7 00          '..'                    STAA    ,X                          |                     // (IX) = A
*E593: BD EA DF       '...'                   JSR     ZEADF                       |                     // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E596: 36             '6'                     PSHA                                |                     // Push A
*E597: D6 72          '.r'                    LDAB    M0072                       |                     // B = (0x0072)
*E599: 96 2F          './'                    LDAA    M002F                       |                     // A = (0x002f); Zählervariable 0x1 ... 0x62
*E59B: D7 2F          './'                    STAB    M002F                       |                     // (0x002f) = B; Zählervariable 0x1 ... 0x62
*E59D: 97 72          '.r'                    STAA    M0072                       |                     // (0x0072) = A;  Tausche Speicherinhalte (0x002f) <> (0x0072)
*E59F: 32             '2'                     PULA                                |                     // Pop A
*E5A0: 4D             'M'                     TSTA                                |                     // A == 0?; MSB == 1 n-Flag, A == 0 z-Flag
*E5A1: 26 09          '&.'                    BNE     ZE5AC >---\                 |                     // if (z) == 0
*E5A3: BD E2 12       '...'                   JSR     ZE212     |                 |                     // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*E5A6: A6 00          '..'                    LDAA    ,X        |                 |                     // A = (IX)
*E5A8: 8A 80          '..'                    ORAA    #$80      |                 |                     // A |= 0x80; Setze Bit 7
*E5AA: A7 00          '..'                    STAA    ,X        |                 |                     // (IX) = A
```

```
*E5AC: DE 27          '.''             ZE5AC   LDX     M0027 <---/                              // IX = (0x0027)
*E5AE: DF 2D          '.-'                     STX     M002D                                    // (0x002d) = IX; Kopiere Zeiger um; Zeiger Text-Start-Ende-Länge
*E5B0: C6 01          '..'                     LDAB    #$01                                     // B = 0x01
*E5B2: BD EA 63       '..c'                    JSR     ZEA63                                    // Call 0xea63 Übergabe D
*E5B5: DE 25          '.%'                     LDX     M0025                                    // IX = (0x0025); Zeiger Text-Ende
*E5B7: DF 2D          '.-'                     STX     M002D                                    // (0x002d) = IX; Kopiere Zeiger um; Zeiger Text-Start-Ende
*E5B9: 7E E5 BC       '~..'                    JMP     ZE5BC  >---\                             // JP 0xe5bc mit Return; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b) mit Return
                                                               |
// Call JP Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E5BC: 36             '6'              ZE5BC   PSHA          <---/                              // Push A
*E5BD: DE 2D          '.-'                     LDX     M002D                                    // IX = (0x002d); Zeiger Text-Start
*E5BF: 9C 25          '.%'             ZE5BF   CPX     M0025 <----\                             // cmp IX, (0x0025); Zeiger Text-Ende
*E5C1: 27 06          ''.'                     BEQ     ZE5C9      | >----\                      // if (z) == 1; (0x002d) == (0x0025)
*E5C3: 09             '.'                      DEX               |      |                      // IX--
*E5C4: 6D 00          'm.'                     TST     ,X        |      |                      // (IX) == 0?; bit7 == 0
*E5C6: 2A F7          '*.'                     BPL     ZE5BF >---/      |                      // if (n) == 0;
*E5C8: 08             '.'                      INX                      |                      // IX++
*E5C9: DF 29          '.)'             ZE5C9   STX     M0029        <---/                      // (0x0029) = IX
*E5CB: 5F             '_'                      CLRB                                             // B = 0
*E5CC: 9C 27          '.''             ZE5CC   CPX     M0027 <----\                             // cmp IX, (0x0027)
*E5CE: 27 08          ''.'                     BEQ     ZE5D8      |   >---\                    // if (z) == 1; Weiter-Abbruch
*E5D0: 6D 00          'm.'                     TST     ,X         |       |                    // (IX) == 0?;  bit7 == 1
*E5D2: 2B 04          '+.'                     BNI     ZE5D8      |   >---+                    // if (n) == 1; Weiter-Abbruch
*E5D4: 5C             '\'                      INCB               |       |                    // B++; Zähler
*E5D5: 08             '.'                      INX                |       |                    // IX++
*E5D6: 20 F4          ' .'                     BRA     ZE5CC >----/       |                    // JR 0xe5cc; Schleife
// Weiter                                                                 |
*E5D8: D7 34          '.4'             ZE5D8   STAB    M0034         <----/                    // (0x0034) = B; Zähler
*E5DA: DF 2B          '.+'                     STX     M002B                                    // (0x002b) = IX; Zeiger Text-aktual Position
*E5DC: 32             '2'                      PULA                                             // Pop A
*E5DD: 39             '9'                      RTS                                              // RETURN

// Call Übergabe A
*E5DE: 81 0D          '..'             ZE5DE   CMPA    #$0D                                     // CMP A, 0x0d; WR/ZV Enter
*E5E0: 26 03          '&.'                     BNE     ZE5E5 >---\                              // if (z) == 0
*E5E2: 7E E6 80       '~..'                    JMP     ZE680     |                              // JP 0xe680
                                                               |
*E5E5: BD E6 B1       '...'            ZE5E5   JSR     ZE6B1 <---/                              // Call 0xe6b1; unverändert A
*E5E8: 7D 00 37       '}.7'                    TST     >M0037                                   // (0x0037) == 0; MSB == 1 n-Flag (0x0037) == 0 z-Flag
*E5EB: 26 25          '&%'                     BNE     ZE612 >---\                              // if (z) == 0
*E5ED: DE 2D          '.-'                     LDX     M002D     |                              // IX = (0x002d)
*E5EF: 9C 27          '.''                     CPX     M0027     |                              // cmp IX, (0x0027)
*E5F1: 24 1F          '$.'                     BCC     ZE612 >---+                              // if (c) == 0
*E5F3: D6 30          '.0'                     LDAB    M0030     |                              // B = (0x0030)
*E5F5: D1 3C          '.<'                     CMPB    M003C     |                              // cmp B, (0x003c)
*E5F7: 25 13          '%.'                     BCS     ZE60C     |          >---\               // if (c) == 1
*E5F9: DE 2D          '.-'                     LDX     M002D     |              |               // IX = (0x002d)
*E5FB: E6 00          '..'                     LDAB    ,X        |              |               // B = (IX)
*E5FD: C1 8D          '..'                     CMPB    #$8D      |              |               // cmp B, 0x8d
*E5FF: 26 06          '&.'                     BNE     ZE607     |    >---\     |               // if (z) == 0
*E601: C6 FF          '..'                     LDAB    #$FF      |        |     |               // B = 0xff
*E603: D7 37          '.7'                     STAB    M0037     |        |     |               // (0x0037) = B
*E605: 20 0B          ' .'                     BRA     ZE612 >---+        |     |               // JR 0xe612
                                                               |        |     |
*E607: 4D             'M'              ZE607   TSTA              |    <---/     |               // A == 0; bit7 == 1
*E608: 2B 1C          '+.'                     BNI     ZE626     |    >---\     |               // if (n) == 1; bit7 == 1
*E60A: D6 30          '.0'                     LDAB    M0030     |        |     |               // B = (0x0030)
*E60C: D1 34          '.4'             ZE60C   CMPB    M0034     |        | <---/               // cmp B, (0x0034)
*E60E: 25 0A          '%.'                     BCS     ZE61A     | >--\   |                     // if (c) == 1
*E610: 20 31          ' 1'                     BRA     ZE643     |    |   | >---\               // JR 0xe643
                                                               |    |   |     |
*E612: D6 34          '.4'             ZE612   LDAB    M0034 <---/    |   |     |               // B = (0x0034)
*E614: D1 30          '.0'                     CMPB    M0030          |   |     |               // cmp B, (0x0030)
*E616: 23 2B          '#+'                     BLS     ZE643          |   | >---+               // if (c) | (z) == 1
*E618: 20 3E          ' >'                     BRA     ZE658          |   |     | >---\         // JR 0xe658
                                                                      |   |     |     |
*E61A: DE 2D          '.-'             ZE61A   LDX     M002D      <---/   |     |     |         // IX = (0x002d)
*E61C: A7 00          '..'                     STAA    ,X                 |     |     |         // (IX) = A
*E61E: 08             '.'                      INX                        |     |     |         // IX++
*E61F: DF 2D          '.-'                     STX     M002D              |     |     |         // (0x002d) = IX
*E621: 7C 00 30       '|.0'                    INC     >M0030             |     |     |         // (0x0030)++
*E624: 20 3F          ' ?'                     BRA     ZE665   >---\      |     |     |         // JR 0xe665
                                                               |      |     |     |
*E626: D6 30          '.0'             ZE626   LDAB    M0030      |  <---/     |     |         // B = (0x0030)
*E628: D0 34          '.4'                     SUBB    M0034      |            |     |         // B -= (0x0034)
*E62A: 27 0B          ''.'                     BEQ     ZE637 >---\ |            |     |         // if (z) == 1
*E62C: 36             '6'                      PSHA              | |            |     |         // Push A
*E62D: BD EA 1E       '...'                    JSR     ZEA1E     | |            |     |         // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E630: BD EA 0F       '...'                    JSR     ZEA0F     | |            |     |         // Call 0xea0f; Speicher füllen mit " ", Übergabe B Rundenanzahl
*E633: 08             '.'                      INX               | |            |     |         // IX++
*E634: DF 2D          '.-'                     STX     M002D     | |            |     |         // (0x002d) = IX
*E636: 32             '2'                      PULA              | |            |     |         // Pop A
*E637: DE 2D          '.-'             ZE637   LDX     M002D <---/ |            |     |         // IX = (0x002d)
*E639: A7 00          '..'                     STAA    ,X          |            |     |         // (IX) = A
*E63B: 08             '.'                      INX                 |            |     |         // IX++
*E63C: DF 2D          '.-'                     STX     M002D       |            |     |         // (0x002d) = IX
*E63E: BD E5 BC       '...'                    JSR     ZE5BC       |            |     |         // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E641: 20 22          ' "'                     BRA     ZE665 >--+--/            |     |         // JR 0xe665
                                                               |                |     |
*E643: 36             '6'              ZE643   PSHA            |          <---/     |         // Push A
*E644: D6 30          '.0'                     LDAB    M0030   |                    |         // B = (0x0030)
*E646: D0 34          '.4'                     SUBB    M0034   |                    |         // B -= (0x0034)
*E648: 5C             '\'                      INCB            |                    |         // B++
*E649: BD EA 1E       '...'                    JSR     ZEA1E   |                    |         // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E64C: BD EA 0F       '...'                    JSR     ZEA0F   |                    |         // Call 0xea0f; Speicher füllen mit " ", Übergabe B Rundenanzahl
```

```
*E64F: 32            '2'                    PULA            |                   |           // Pop A
*E650: 09            '.'                    DEX             |                   |           // IX--
*E651: A7 00         '..'                   STAA    ,X      |                   |           // (IX) = A
*E653: 7C 00 30      '|.0'                  INC     >M0030  |                   |           // (0x0030)++
*E656: 20 0D         ' .'                   BRA     ZE665 >--+                   |           // JR 0xe665
                                                            |                   |
*E658: 36            '6'            ZE658   PSHA            |                   <---/       // Push A
*E659: C6 01         '..'                   LDAB    #$01    |                               // B = 0x01
*E65B: BD EA 1E      '...'                  JSR     ZEA1E   |                               // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E65E: 32            '2'                    PULA            |                               // Pop A
*E65F: 08            '.'                    INX             |                               // IX++
*E660: A7 00         '..'                   STAA    ,X      |                               // (IX) = A
*E662: 08            '.'                    INX             |                               // IX++
*E663: DF 2D         '.-'                   STX     M002D   |                               // (0x002d) = IX
                                                            |
// Call JP                                                  |
*E665: 96 30         '.0'           ZE665   LDAA    M0030 <--/                            // A = (0x0030)
*E667: 97 66         '.f'                   STAA    M0066                                 // (0x0066) = A
*E669: BD E9 C3      '...'                  JSR     ZE9C3                                 // Call 0xe9c3; Ermittle Textanfang-Ende GTX
*E66C: BD E5 BC      '...'                  JSR     ZE5BC                                 // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E66F: BD E7 6C      '..l'                  JSR     ZE76C                                 // Call 0xe76c;(0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
*E672: 96 30         '.0'                   LDAA    M0030                                 // A = (0x0030)
*E674: 91 66         '.f'                   CMPA    M0066                                 // CMP A, (0x0066)
*E676: 27 03         ''.'                   BEQ     ZE67B >--\                            // if (z) == 1
*E678: 7F 00 31      '..1'                  CLR     >M0031  |                             // (0x0031) = 0
*E67B: 86 FF         '..'           ZE67B   LDAA    #$FF  <--/                            // A = 0xff
*E67D: 97 38         '.8'                   STAA    M0038                                 // (0x0038) = A
*E67F: 39            '9'                    RTS                                           // RETURN

// JP
*E680: 7D 00 37      '}.7'          ZE680   TST     >M0037                                // (0x0037) == 0?; MSB == 1 n-Flag (0x0037) == 0 z-Flag
*E683: 26 14         '&.'                   BNE     ZE699 >---\                           // if (z) == 0
*E685: DE 2B         '.+'                   LDX     M002B     |                           // IX = (0x002b); Zeiger Text-aktual Position
*E687: DF 2D         '.-'                   STX     M002D     |                           // (0x002d) = IX; Zeiger Text-Start
*E689: 96 34         '.4'                   LDAA    M0034     |                           // A = (0x0034)
*E68B: 97 30         '.0'                   STAA    M0030     |                           // (0x0030) = A
*E68D: 9C 27         '.''                   CPX     M0027     |                           // cmp IX, (0x0027)
*E68F: 24 05         '$.'                   BCC     ZE696     |       >---\               // if (c) == 0
*E691: BD F5 D9      '...'                  JSR     ZF5D9     |           |               // Call 0xf5d9; Prüfe (0x0031) == (0x0070)
*E694: 20 0C         ' .'                   BRA     ZE6A2     | >---\     |               // JR 0xe6a2
                                                             |     |     |
*E696: BD F6 11      '...'          ZE696   JSR     ZF611     |     | <---/             // Call 0xf611
*E699: C6 01         '..'           ZE699   LDAB    #$01  <---/     |                   // B = 0x01
*E69B: BD EA 1E      '...'                  JSR     ZEA1E           |                   // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E69E: 86 8D         '..'                   LDAA    #$8D            |                   // A = 0x8d
*E6A0: A7 01         '..'                   STAA    $01,X           |                   // (IX) = A
*E6A2: DE 2D         '.-'           ZE6A2   LDX     M002D       <---/                 // IX = (0x002d)
*E6A4: 08            '.'                    INX                                       // IX++
*E6A5: DF 2D         '.-'                   STX     M002D                             // (0x002d) = IX
*E6A7: 7F 00 30      '..0'                  CLR     >M0030                            // (0x0030) = 0
*E6AA: BD E5 BC      '...'                  JSR     ZE5BC                             // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E6AD: BD EA D6      '...'                  JSR     ZEAD6                             // Call 0xead6 Zeitschleife 10 ms
*E6B0: 39            '9'                    RTS                                       // RETURN

// Call Unverändert A
*E6B1: D6 3C         '.<'           ZE6B1   LDAB    M003C                             // B = (0x003c)
*E6B3: D0 30         '.0'                   SUBB    M0030                             // B -= (0x0030)
*E6B5: C1 08         '..'                   CMPB    #$08                              // cmp B, 0x08
*E6B7: 26 05         '&.'                   BNE     ZE6BE >---\                       // if (z) == 0 RETURN
*E6B9: 36            '6'                    PSHA            |                         // Push A
*E6BA: BD EA AA      '...'                  JSR     ZEAAA   |                         // Call 0xeaaa; Modem ON Übergabe A Port-14, B Zeitschleife  30ms
*E6BD: 32            '2'                    PULA            |                         // Pop A
*E6BE: 39            '9'            ZE6BE   RTS         <---/                         // RETURN

// Call Sprungverteiler 1
*E6BF: BD EA DF      '...'                  JSR     ZEADF                             // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E6C2: 27 68         ''h'                   BEQ     ZE72C                 >---\       // if (z) == 1 springe
*E6C4: 73 00 38      's.8'                  COM     >M0038                    |       // Not (0x0038)
*E6C7: DE 2D         '.-'                   LDX     M002D                     |       // IX = (0x002d); Zeiger Text-Start
*E6C9: 9C 25         '.%'                   CPX     M0025                     |       // cmp IX, (0x0025); Zeiger Text-Ende
*E6CB: 22 05         '".'                   BHI     ZE6D2 >---\               |       // if (c) & (z) == 0
*E6CD: 7D 00 30      '}.0'                  TST     >M0030    |               |       // (0x0030) == 0?; MSB == 1 n-Flag (0x0030) == 0 z-Flag
*E6D0: 27 5A         ''Z'                   BEQ     ZE72C     |           <---+       // if (z) == 1 springe
*E6D2: 96 30         '.0'           ZE6D2   LDAA    M0030 <---/               |       // A = (0x0030)
*E6D4: 26 0F         '&.'                   BNE     ZE6E5 >---\               |       // if (z) == 0
*E6D6: 96 3C         '.<'                   LDAA    M003C     |               |       // A = (0x003c)
*E6D8: 97 30         '.0'                   STAA    M0030     |               |       // (0x0030) = A
*E6DA: 09            '.'                    DEX             |               |       // IX--
*E6DB: DF 2D         '.-'                   STX     M002D     |               |       // (0x002d) = IX
*E6DD: BD E5 BC      '...'                  JSR     ZE5BC     |               |       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E6E0: 7F 00 31      '..1'                  CLR     >M0031    |               |       // (0x0031) = 0
*E6E3: 20 0A         ' .'                   BRA     ZE6EF     | >---\         |       // JR -> ZE72B -> RETURN
                                                             |     |         |       |
*E6E5: 91 34         '.4'           ZE6E5   CMPA    M0034 <---/     |         |       // CMP A, (0x0034)
*E6E7: 22 03         '".'                   BHI     ZE6EC >---\     |         |       // if (c) & (z) == 0
*E6E9: 09            '.'                    DEX             |     |         |       // IX--
*E6EA: DF 2D         '.-'                   STX     M002D   |     |         |       // (0x002d) = IX
*E6EC: 4A            'J'            ZE6EC   DECA        <---/     |         |       // A--
*E6ED: 97 30         '.0'                   STAA    M0030         |         |       // (0x0030) = A
*E6EF: 20 3A         ' :'           ZE6EF   BRA     ZE72B     <---/         |       // JR -> RETURN
                                                                           |
                                                                           |
// Call Sprungverteiler 1                                                  |
*E6F1: BD EA DF      '...'                  JSR     ZEADF                     |       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E6F4: 27 36         ''6'                   BEQ     ZE72C  >---\              |       // if (z) == 1; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
```

```
*E6F6: BD E2 0D       '...'                   JSR     ZE20D       |                |                // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E6F9: 26 03          '&.'                    BNE     ZE6FE       |  >---\         |                // if (z) == 0
*E6FB: BD E2 C0       '...'                   JSR     ZE2C0       |      |         |                // Call 0xE2C0; Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*E6FE: 73 00 38       's.8'           ZE6FE   COM     >M0038      |  <---/         |                // Not (0x0038)
*E701: 96 30          '.0'                    LDAA    M0030       |                |                // A = (0x0030)
*E703: 91 3C          '.<'                    CMPA    M003C       |                |                // CMP A, (0x003c)
*E705: 2B 0F          '+.'                    BNI     ZE716       |  >---\         |                // if (n) == 1; bit7 == 0
*E707: DE 2D          '.-'                    LDX     M002D       |      |         |                // IX = (0x002d)
*E709: 9C 27          '.''                    CPX     M0027       |      |         |                // cmp IX, (0x0027)
*E70B: 24 1F          '$.'                    BCC     ZE72C  >----+      |         |                // if (c) == 0; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E70D: 08             '.'                     INX                        |         |                // IX++
*E70E: DF 2D          '.-'                    STX     M002D              |         |                // (0x002d) = IX
*E710: BD E5 BC       '...'                   JSR     ZE5BC              |         |                // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E713: 4F             'O'                     CLRA                       |         |                // A = 0
*E714: 20 0A          ' .'                    BRA     ZE720              |         | >---\  |       // JR 0xe720
                                                                         |         |     |  |
*E716: 91 34          '.4'            ZE716   CMPA    M0034       |  <---/         |  |  |       // CMP A, (0x0034)
*E718: 24 05          '$.'                    BCC     ZE71F       |  >---\         |  |  |       // if (c) == 0
*E71A: DE 2D          '.-'                    LDX     M002D       |      |         |  |  |       // IX = (0x002d)
*E71C: 08             '.'                     INX                 |      |         |  |  |       // IX++
*E71D: DF 2D          '.-'                    STX     M002D       |      |         |  |  |       // (0x002d) = IX
*E71F: 4C             'L'             ZE71F   INCA                |  <---/         |  |  |       // A++
*E720: 97 30          '.0'            ZE720   STAA    M0030       |            <---/  |  |       // (0x0030) = A
*E722: DE 2D          '.-'                    LDX     M002D       |                   |  |       // IX = (0x002d)
*E724: 9C 27          '.''                    CPX     M0027       |                   |  |       // cmp IX, (0x0027)
*E726: 25 03          '%.'                    BCS     ZE72B       | >---\             |  |       // if (c) == 1 RETURN
*E728: 7F 00 37       '..7'                   CLR     >M0037      |     |             |  |       // (0x0037) = 0
*E72B: 39             '9'             ZE72B   RTS                 | <---+             |  |       // RETURN
                                                                  |     |             |  |
*E72C: 7E EA 94       '~..'           ZE72C   JMP     ZEA94  <----+     |    <------+  |       // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                        |     |        |
// Call Sprungverteiler 3                                               |     |        |
*E72F: BD EA DF       '...'                   JSR     ZEADF       |     |          |       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E732: 27 F8          ''.'                    BEQ     ZE72C >-----+     |          |       // if (z) == 1; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E734: 73 00 38       's.8'                   COM     >M0038      |     |          |       // Not (0x0038)
*E737: DE 29          '.)'                    LDX     M0029       |     |          |       // IX = (0x0029); Zeiger Text-aktual Position
*E739: 9C 25          '.%'                    CPX     M0025       |     |          |       // cmp IX, (0x0025); Zeiger Text-Ende
*E73B: 23 EF          '#.'                    BLS     ZE72C  >----+     |          |       // if (c) | (z) == 1; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E73D: 09             '.'                     DEX                       |          |       // IX--
*E73E: DF 2D          '.-'            ZE73E   STX     M002D       |     |          |  <----\ // (0x002d) = IX
*E740: BD E5 BC       '...'                   JSR     ZE5BC       |     |          |       | // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E743: 7F 00 31       '..1'                   CLR     >M0031      |     |          |       | // (0x0031) = 0
*E746: 8D 16          '..'                    BSR     ZE75E       |     |          |       | // Call 0xe57e Übergabe (0x0030);  If (0x0034) != B then (0x002d) = (0x0029) + B
*E748: 7F 00 37       '..7'                   CLR     >M0037      |     |          |       | // (0x0037) = 0
*E74B: 20 DE          ' .'                    BRA     ZE72B       | >---/          |       | // JR -> RETURN
                                                                  |                |       |
// Call Sprungverteiler 4                                         |                |       |
*E74D: BD EA DF       '...'                   JSR     ZEADF       |                |       |  // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E750: 27 DA          ''.'                    BEQ     ZE72C  >---/                |       |  // if (z) == 1; -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E752: 73 00 38       's.8'                   COM     >M0038                      |       |  // Not (0x0038)
*E755: DE 2B          '.+'                    LDX     M002B                       |       |  // IX = (0x002b); Zeiger Text-aktual Position
*E757: 9C 27          '.''                    CPX     M0027                       |       |  // cmp IX, (0x0027); Zeiger Text-aktual Position
*E759: 24 D1          '$.'                    BCC     ZE72C                 >-----/       |  // if (c) == 0; ->  0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E75B: 08             '.'                     INX                                       |  // IX++
*E75C: 20 E0          ' .'                    BRA     ZE73E                       >-----/  // JR in den Sprungverteiler 3

// Call If (0x0034) != (0x0030) then (0x002d) = (0x0029) + (0x0030)
*E75E: D6 30          '.0'            ZE75E   LDAB    M0030                               // B = (0x0030)
*E760: D1 34          '.4'                    CMPB    M0034                               // cmp B, (0x0034)
*E762: 23 02          '#.'                    BLS     ZE766 >---\                         // if (c) | (z) == 1
*E764: D6 34          '.4'                    LDAB    M0034     |                         // B = (0x0034)
*E766: DE 29          '.)'            ZE766   LDX     M0029 <---/                         // IX = (0x0029)
*E768: 3A             ':'                     ABX                                         // IX += B
*E769: DF 2D          '.-'                    STX     M002D                               // (0x002d) = IX
*E76B: 39             '9'                     RTS                                         // RETURN

// Call (0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
*E76C: DC 2D          '.-'            ZE76C   LDD     M002D                               // D = (0x002d); Zeiger Text-Start
*E76E: 93 29          '.)'                    SUBD    M0029                               // D -= (0x0029); Zeiger Text-aktual Position
*E770: D7 30          '.0'                    STAB    M0030                               // (0x0030) = B; Länge
*E772: 39             '9'                     RTS                                         // RETURN

// Call Sprungverteiler 24; IX = (0x0029)
*E773: DE 29          '.)'                    LDX     M0029                               // IX = (0x0029)
*E775: 20 02          ' .'                    BRA     ZE779 >---\                         // JR Sprungverteiler 21
                                                              |
// Call Sprungverteiler 21; IX = (0x0025)                     |
*E777: DE 25          '.%'                    LDX     M0025     |                         // IX = (0x0025); Zeiger Text-Ende
// JP aus Sprungverteiler 24                                  |
*E779: 3C             '<'             ZE779   PSHX          <---/                         // Push IX; (0x0029) oder (0x0025)
*E77A: BD EA DF       '...'                   JSR     ZEADF                               // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E77D: 38             '8'                     PULX                                        // Pop IX; (0x0029) oder (0x0025)
*E77E: 27 4E          ''N'                    BEQ     ZE7CE   >---\                       // if (z) == 1; ->  0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E780: 9C 2D          '.-'                    CPX     M002D       |                       // cmp IX, (0x002d)
*E782: 25 05          '%.'                    BCS     ZE789 >---\ |                       // if (c) == 1
*E784: 7D 00 30       '}.0'                   TST     >M0030    | |                       // (0x0030) == 0?;  MSB == 1 n-Flag (0x0030) == 0 z-Flag
*E787: 27 45          ''E'                    BEQ     ZE7CE >-- | +                       // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E789: DF 2D          '.-'            ZE789   STX     M002D <---/ |                       // (0x002d) = IX
*E78B: BD E5 BC       '...'                   JSR     ZE5BC       |                       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E78E: 7F 00 30       '..0'                   CLR     >M0030      |                       // (0x002d) = 0
*E791: 39             '9'                     RTS                 |                       // RETURN
                                                                  |
// Call Sprungverteiler 22                                        |
*E792: BD EA DF       '...'                   JSR     ZEADF       |                       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E795: 27 37          ''7'                    BEQ     ZE7CE   >---+                       // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
```

```
*E797: DE 2B          '.+'                    LDX     M002B       |                       // IX = (0x002b); Zeiger Text-aktual Position
*E799: 9C 2D          '.-'                    CPX     M002D       |                       // cmp IX, (0x002d); Zeiger Text-Start
*E79B: 22 06          '".'                    BHI     ZE7A3       | >---\                 // if (c) & (z) == 0
*E79D: D6 30          '.0'                    LDAB    M0030       |     |                 // B = (0x0030)
*E79F: D1 3C          '.<'                    CMPB    M003C       |     |                 // cmp B, (0x003c)
*E7A1: 24 2B          '$+'                    BCC     ZE7CE    >----+     |                 // if (c) == 0 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E7A3: DF 2D          '.-'            ZE7A3   STX     M002D       | <---/                 // (0x002d) = IX
*E7A5: 96 3C          '.<'                    LDAA    M003C       |                       // A = (0x003c)
*E7A7: 97 30          '.0'                    STAA    M0030       |                       // (0x0030) = A
*E7A9: BD E5 BC       '...'                   JSR     ZE5BC       |                       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E7AC: 7F 00 37       '..7'                   CLR     >M0037      |                       // (0x0037) = 0
*E7AF: 39             '9'                     RTS                 |                       // RETURN
                                                                  |
// Call Sprungverteiler 23                                        |
*E7B0: BD EA DF       '...'                   JSR     ZEADF       |                       // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E7B3: 27 19          ''.'                    BEQ     ZE7CE    >----+                     // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E7B5: DE 27          '.''                    LDX     M0027       |                       // IX = (0x0027)
*E7B7: 9C 2D          '.-'                    CPX     M002D       |                       // cmp IX, (0x002d)
*E7B9: 22 06          '".'                    BHI     ZE7C1       | >---\                 // if (c) & (z) == 0
*E7BB: D6 30          '.0'                    LDAB    M0030       |     |                 // B = (0x0030)
*E7BD: D1 34          '.4'                    CMPB    M0034       |     |                 // cmp B, (0x0034)
*E7BF: 27 0D          ''.'                    BEQ     ZE7CE    >----+     |                 // if (z) == 1 -> 0xea94 Sprungverteiler 17, 18; Tastaturabfrage Zi/Zeichen/Sonderzeichen
*E7C1: DF 2D          '.-'            ZE7C1   STX     M002D       | <---/                 // (0x002d) = IX
*E7C3: BD E5 BC       '...'                   JSR     ZE5BC       |                       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E7C6: 96 34          '.4'                    LDAA    M0034       |                       // A = (0x0034)
*E7C8: 97 30          '.0'                    STAA    M0030       |                       // (0x0030) = A
*E7CA: 7F 00 37       '..7'                   CLR     >M0037      |                       // (0x0037) = 0
*E7CD: 39             '9'                     RTS                 |                       // RETURN
                                                                  |
                                                                 /
*E7CE: 7E EA 94       '~..'           ZE7CE   JMP     ZEA94 <---+---------------------\   // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                |                     |
// Call Sprungverteiler 25                                      |                     |
*E7D1: BD EA DF       '...'                   JSR     ZEADF       |                     |   // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E7D4: 27 F8          ''.'                    BEQ     ZE7CE >---+ |                     |   // if (z) == 1 ->0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E7D6: 73 00 38       's.8'                   COM     >M0038    | |                     |   // Not (0x0038)
*E7D9: BD E2 0D       '...'                   JSR     ZE20D     | |                     |   // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E7DC: 26 03          '&.'                    BNE     ZE7E1     |  >---\              |   // if (z) == 0
*E7DE: BD E2 C0       '...'                   JSR     ZE2C0     |      |              |   // Call 0xE2C0; Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*E7E1: 7D 00 37       '}.7'           ZE7E1   TST     >M0037    |  <---/              |   // (0x0037) == 0?;  MSB == 1 n-Flag (0x0037) == 0 z-Flag
*E7E4: 27 06          ''.'                    BEQ     ZE7EC     |  >---\              |   // if (z) == 1
*E7E6: 96 30          '.0'                    LDAA    M0030     |      |              |   // A = (0x0030)
*E7E8: 91 34          '.4'                    CMPA    M0034     |      |              |   // CMP A, (0x0034)
*E7EA: 25 0A          '%.'                    BCS     ZE7F6     |      | >---\        |   // if (c) == 1
*E7EC: 8D 19          '..'            ZE7EC   BSR     ZE807     |  <---/     |        |   // Call 0xe807; Übergabe (0x0030), Rückgabe B = 0x7f und Flag
*E7EE: 22 DE          '".'                    BHI     ZE7CE >---/            |        |   // if (c) & (z) == 0 -> 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E7F0: D7 30          '.0'                    STAB    M0030                  |        |   // (0x0030) = B
*E7F2: BD E7 5E       '..^'                   JSR     ZE75E                  |        |   // Call 0xe57e Übergabe (0x0030);  If (0x0034) != B then (0x002d) = (0x0029) + B
*E7F5: 39             '9'             ZE7F5   RTS           <---\            |        |   // RETURN
                                                                |            |        |
// JP                                                           |            |        |
*E7F6: 8D 0F          '..'            ZE7F6   BSR     ZE807     |        <---/        |   // Call 0xe807; Übergabe (0x0030), Rückgabe B = 0x7f und Flag
*E7F8: 22 FB          '".'                    BHI     ZE7F5 >---+                     |   // if (c) & (z) == 0 RETURN
*E7FA: D0 30          '.0'                    SUBB    M0030     |                     |   // B -= (0x0030)
*E7FC: BD EA 1E       '...'                   JSR     ZEA1E     |                     |   // Call 0xea1e; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*E7FF: BD EA 0F       '...'                   JSR     ZEA0F     |                     |   // Call 0xea0f; Speicher füllen mit " ", Übergabe B Rundenanzahl
*E802: BD E6 65       '..e'                   JSR     ZE665     |                     |   // Call 0xe665
*E805: 20 EE          ' .'                    BRA     ZE7F5 >---/                     |   // JR RETURN
                                                                                      |
// CALL Übergabe (0x0030), Rückgabe B = 0x7f und Flag                                 |
*E807: D6 30          '.0'            ZE807   LDAB    M0030                           |   // B = (0x0030); Schrittweite?
*E809: CE 01 00       '...'                   LDX     #M0100                          |   // IX = 0x0100
*E80C: 5C             '\'                     INCB                                    |   // B++; Schrittweite
*E80D: 3A             ':'                     ABX                                     |   // IX += B; 0x0100 + (0x0030)
*E80E: 6D 00          'm.'            ZE80E   TST     ,X          <---\               |   // (IX) == 0?;  MSB == 1 n-Flag (IX) == 0 z-Flag
*E810: 26 09          '&.'                    BNE     ZE81B >---\     |               |   // if (z) == 0
*E812: 5C             '\'                     INCB               |     |               |   // B++
*E813: 08             '.'                     INX                |     |               |   // IX++; 0x0100 + B * (0x0130); Schrittweite * 2
*E814: 8C 01 50       '..P'                   CPX     #M0150     |     |               |   // cmp IX, 0x0150; maximal 80/2 = 40 Runden
*E817: 23 F5          '#.'                    BLS     ZE80E      | >---/               |   // if (c) | (z) == 1; Schleife bis IX = 0x0150
*E819: C6 7F          '..'                    LDAB    #$7F       |                     |   // B = 0x7f
*E81B: D1 3C          '.<'            ZE81B   CMPB    M003C <---/                     |   // cmp B, (0x003c)
*E81D: 39             '9'                     RTS                                     |   // RETURN
                                                                                      |
// Call Sprungverteiler 27 Fülle Speicherbereich 0x0100 ... 0x014f mit 0x0 bzw. 0xff  |
*E81E: BD EA DF       '...'           ZE81E   JSR     ZEADF                           |   // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E821: 27 AB          ''.'                    BEQ     ZE7CE                    >----+   // if (z) == 1  -> 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E823: CE 01 00       '...'                   LDX     #M0100                          |   // IX = 0x0100
*E826: 6F 00          'o.'            ZE826   CLR     ,X      <---\                   |   // (IX) = 0
*E828: 08             '.'                     INX                |                    |   // IX++
*E829: 8C 01 50       '..P'                   CPX     #M0150     |                    |   // cmp IX, 0x0150
*E82C: 23 F8          '#.'                    BLS     ZE826 >---/                     |   // if (c) | (z) == 1 Schleife Fülle 0x0100 bis 0x014f mit 0x0
*E82E: CE 01 08       '...'                   LDX     #M0108                          |   // IX = 0x0108
*E831: 86 FF          '..'                    LDAA    #$FF                            |   // A = 0xff
*E833: C6 08          '..'                    LDAB    #$08                            |   // B = 0x08 Schrittweite
*E835: A7 00          '..'            ZE835   STAA    ,X      <---\                   |   // (IX) = A
*E837: 3A             ':'                     ABX                |                    |   // IX += B
*E838: 8C 01 50       '..P'                   CPX     #M0150     |                    |   // cmp IX, 0x0150
*E83B: 25 F8          '%.'                    BCS     ZE835 >---/                     |   // if (c) == 1 Schleife Fülle 0x0108, Schrittweite 8 bis 0x014F mit 0xff
*E83D: DE 29          '.)'                    LDX     M0029                           |   // IX = (0x0029)
*E83F: DF 2D          '.-'                    STX     M002D                           |   // (0x002d) = IX
*E841: 7F 00 30       '..0'                   CLR     >M0030                          |   // (0x0030) = 0
*E844: 39             '9'                     RTS                                     |   // RETURN
                                                                                      |
// Call Sprungverteiler 26                                                            |
```

```
*E845: BD EA DF       '...'                   JSR     ZEADF                           |           // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E848: 27 84          ''.'                    BEQ     ZE7CE                        >---/          // if (z) == 1  -> 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*E84A: D6 30          '.0'                    LDAB    M0030                                       // B = (0x0030)
*E84C: CE 01 00       '...'                   LDX     #M0100                                      // IX = 0x0100
*E84F: 3A             ':'                     ABX                                                 // IX += B; 0x0100 + (0x0030) Schrittweite
*E850: 63 00          'c.'                    COM     ,X                                          // Complement (IX)
*E852: 39             '9'                     RTS                                                 // RETURN

// Call Sprungverteiler 19 "Suche"
*E853: CE F8 51       '..Q'                   LDX     #ZF851                                      // IX = 0xf851; "Search for ...."
*E856: BD ED 82       '...'                   JSR     ZED82                                       // Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
*E859: BD ED CD       '...'                   JSR     ZEDCD                                       // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
// Zeiger Intialisieren
*E85C: CE 00 55       '..U'                   LDX     #M0055                                      // IX = 0x0055
*E85F: DF AB          '..'                    STX     M00AB                                       // (0x00ab) = IX; Zeiger auf 0x0055
*E861: CE 00 4D       '..M'                   LDX     #M004D                                      // IX = 0x004d
*E864: DF A9          '..'                    STX     M00A9                                       // (0x00a9) = IX; Zeiger auf 0x00a9
*E866: CE 00 08       '...'                   LDX     #M0008                                      // IX = 0x0008
*E869: DF AD          '..'                    STX     M00AD                                       // (0x00ad) = IX; Zeiger auf 0x0008
*E86B: BD F4 EB       '...'                   JSR     ZF4EB                                       // Call 0xf4eb; Kopiere (0x00a9)->byte->(0x00ab); Adresse--; Schleifenzähler: (0x00ad)--
*E86E: 7F 00 45       '..E'                   CLR     >M0045                                      // (0x0045) = 0
*E871: BD E9 A5       '...'           ZE871   JSR     ZE9A5                           <---\       // Call 0xe9a5
*E874: 86 29          '.)'                    LDAA    #$29                                |       // A = 0x29; 40 Elemente
*E876: 97 32          '.2'                    STAA    M0032                               |       // (0x0032) = A
*E878: BD EC F9       '...'                   JSR     ZECF9                               |       // Call 0xecf9; Übergabe (0x0032)
*E87B: CE 00 4E       '..N'                   LDX     #M004E                              |       // IX = 0x004e
*E87E: 86 0B          '..'                    LDAA    #$0B                                |       // A = 0x0b
*E880: 6D 00          'm.'            ZE880   TST     ,X              <---\               |       // (IX) == 0?; MSB == 1 n-Flag (IX) == 0 z-Flag
*E882: 27 09          ''.'                    BEQ     ZE88D >---\     |   |               |       // if (z) == 1 Abbruch
*E884: 8C 00 56       '..V'                   CPX     #M0056    |     |   |               |       // cmp IX, (0x0056); von 0x00e4 ... 0x0056
*E887: 27 04          ''.'                    BEQ     ZE88D >---+     |   |               |       // if (z) == 1; Abbruch
*E889: 4C             'L'                     INCA              |     |   |               |       // A++
*E88A: 08             '.'                     INX               |     |   |               |       // IX++
*E88B: 20 F3          ' .'                    BRA     ZE880     | >---/   |               |       // JR 0xe880; Schleife
// Weiter Übergabe A                                            |         |               |
*E88D: 97 32          '.2'            ZE88D   STAA    M0032 <---/         |               |       // (0x0032) = A
*E88F: BD F3 F9       '...'           ZE88F   JSR     ZF3F9                               |       // Call 0xf3f9; Tastendruck; Rückgabe A
*E892: 81 A2          '..'                    CMPA    #$A2                                |       // CMP A, 0xa2; Taste "SEARCH"
*E894: 27 3F          ''?'                    BEQ     ZE8D5           >---\               |       // if (z) == 1
*E896: 81 97          '..'                    CMPA    #$97                |               |       // CMP A, 0x97; Taste DEL
*E898: 27 25          ''%'                    BEQ     ZE8BF               | >---\       |       // if (z) == 1
*E89A: 81 90          '..'                    CMPA    #$90                |     |       |       // CMP A, 0x90; Taste "<-"
*E89C: 25 01          '%.'                    BCS     ZE89F >---\         |     |       |       // if (c) == 1
*E89E: 39             '9'                     RTS               |         |     |       |       // RETURN
                                                                |         |     |       |
// JP Taste "<-" 0x90;                                          |         |     |       |
*E89F: 7D 00 45       '}.E'           ZE89F   TST     >M0045 <--/         |     |       |       // (0x0045) == 0?;  MSB == 1 n-Flag (0x0045) == 0 z-Flag
*E8A2: 26 06          '&.'                    BNE     ZE8AA  >---\        |     |       |       // if (z) == 0
*E8A4: BD EB DA       '...'                   JSR     ZEBDA      |        |     |       |       // Call 0xebda; Lösche 0x004e ... 0x0055
*E8A7: 73 00 45       's.E'                   COM     >M0045     |        |     |       |       // Complement (0x0045)
*E8AA: CE 00 4E       '..N'           ZE8AA   LDX     #M004E <---/        |     |       |       // IX = 0x004e
*E8AD: 6D 00          'm.'            ZE8AD   TST     ,X          <---\   |     |       |       // (IX) == 0?; MSB == 1 n-Flag (IX) == 0 z-Flag
*E8AF: 27 0A          ''.'                    BEQ     ZE8BB >---\     |   |     |       |       // if (z) == 1
*E8B1: 08             '.'                     INX               |     |   |     |       |       // IX++
*E8B2: 8C 00 56       '..V'                   CPX     #M0056    |     |   |     |       |       // cmp IX, 0x0056
*E8B5: 23 F6          '#.'                    BLS     ZE8AD     | >---/   |     |       |       // if (c) | (z) == 1
*E8B7: 8D 6D          '.m'                    BSR     ZE926     |         |     |       |       // Call 0xe926
*E8B9: 20 D4          ' .'                    BRA     ZE88F     |         |     |       |       // JR 0xe88f -> Call 0xf3f9; Tastendruck; Rückgabe A
                                                                |         |     |       |
*E8BB: A7 00          '..'            ZE8BB   STAA    ,X    <---/         |     |       |       // (IX) = A
*E8BD: 20 B2          ' .'                    BRA     ZE871               |     |   >---+       // JR -> Call 0xe9a5
                                                                          |     |               |
//JP Taste DEL 0x97;                                                      |     |               |
*E8BF: CE 00 4F       '..O'           ZE8BF   LDX     #M004F              | <---/               |       // IX = 0x004f
*E8C2: 6D 00          'm.'            ZE8C2   TST     ,X          <---\   |                     |       // (IX) == 0?; MSB == 1 n-Flag (IX) == 0 z-Flag
*E8C4: 27 06          ''.'                    BEQ     ZE8CC >---\     |   |                     |       // if (z) == 1; gefunden
*E8C6: 08             '.'                     INX               |     |   |                     |       // IX++
*E8C7: 8C 00 56       '..V'                   CPX     #M0056    |     |   |                     |       // cmp IX, 0x0056
*E8CA: 23 F6          '#.'                    BLS     ZE8C2     | >---/   |                     |       // if (c) | (z) == 1; Schleife 0x004f ... 0x0056
*E8CC: 09             '.'             ZE8CC   DEX           <---/         |                     |       // IX--
*E8CD: 6F 00          'o.'                    CLR     ,X                  |                     |       // (IX) = 0
*E8CF: 86 FF          '..'                    LDAA    #$FF                |                     |       // A = 0xff
*E8D1: 97 45          '.E'                    STAA    M0045               |                     |       // (0x0045) = A
*E8D3: 20 9C          ' .'                    BRA     ZE871               |           >---/     |       // JR -> Call 0xe9a5
// JP "SEARCH"                                                            |                     |
*E8D5: 96 4E          '.N'            ZE8D5   LDAA    M004E           <---/                     |       // A = (0x004e)
*E8D7: 27 4D          ''M'                    BEQ     ZE926  >---\                              |       // if (z) == 1
*E8D9: BD EB BF       '...'                   JSR     ZEBBF      |                              |       // Call 0xebbf; Init 0x0046 .. 0x004d; Zeiger
*E8DC: 7F 00 08       '...'                   CLR     >M0008     |                              |       // (0x0008) = 0; Port21 Output, TCSR1 Timer Control Status 1
*E8DF: 0E             '.'                     CLI                |                              |       // Clear Interrupt Mask = 0
*E8E0: 96 2F          './'            ZE8E0   LDAA    M002F      |                    <---\     |       // A = (0x002f); Zählervariable 0x1 ... 0x62
*E8E2: 97 72          '.r'                    STAA    M0072      |                        |     |       // (0x0072) = A
*E8E4: DE 2D          '.-'                    LDX     M002D      |                        |     |       // IX = (0x002d)
*E8E6: DF 70          '.p'                    STX     M0070      |                        |     |       // (0x0070) = IX
*E8E8: BD E2 0D       '...'           ZE8E8   JSR     ZE20D      |                        | <---\  // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*E8EB: 27 49          ''I'                    BEQ     ZE936      |   >---\                |     |  // if (z) == 1
*E8ED: BD EA DF       '...'                   JSR     ZEADF      |       |                |     |  // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*E8F0: 27 44          ''D'                    BEQ     ZE936      |   >---+                |     |  // if (z) == 1
*E8F2: DE 2D          '.-'                    LDX     M002D      |       |                |     |  // IX = (0x002d)
*E8F4: 9C 27          '.''            ZE8F4   CPX     M0027 <--- | \     |                |     |  // cmp IX, (0x0027)
*E8F6: 24 3E          '$>'                    BCC     ZE936      | | >---+                |     |  // if (c) == 0
*E8F8: 96 4E          '.N'                    LDAA    M004E      | |     |                |     |  // A = (0x004e)
*E8FA: BD E9 9A       '...'                   JSR     ZE99A      | |     |      >---\     |     |  // Call 0xe99a; Übergabe A, Rückgabe A &= 0xdf; if A > 0x60 and < 0x7b then "a ... z" -> "A ... Z"
*E8FD: 16             '.'                     TAB                | |     |          |     |     |  // B = A
*E8FE: A6 00          '..'                    LDAA    ,X         | |     |          |     |     |  // A = (IX)
```

```
*E900: BD E9 9A       '...'                    JSR     ZE99A       | |     |      >---+     |     |  // Call 0xe99a; Übergabe A, Rückgabe A &= 0xdf; if A > 0x60 and < 0x7b then "a ... z" -> "A ... Z"
*E903: 11             '.'                      CBA                 | |     |          |     |     |  // cmp A  B
*E904: 26 23          '&#'                     BNE     ZE929       | |     | >---\    |     |     |  // if (z) == 0
*E906: 8D 5B          '.['                     BSR     ZE963       | |     |     |    |     |     |  // Call 0xe963
*E908: 26 1F          '&.'                     BNE     ZE929       | |     | >---+    |     |     |  // if (z) == 0
*E90A: DF 2D          '.-'                     STX     M002D       | |     |     |    |     |     |  // (0x002d) = IX
*E90C: BD E5 BC       '...'                    JSR     ZE5BC       | |     |     |    |     |     |  // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*E90F: BD E7 6C       '..l'                    JSR     ZE76C       | |     |     |    |     |     |  // Call 0xe76c; (0x002d) - (0x0029) Rückgabe Low-teil (0x0030)
*E912: BD EC 74       '..t'                    JSR     ZEC74       | |     |     |    |     |     |  // Call 0xec74 Rückgabe  A = (0x0030) bzw. (0x0031)
*E915: BD ED EC       '...'                    JSR     ZEDEC       | |     |     |    |     |     |  // Call 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe
*E918: 0F             '.'                      SEI                 | |     |     |    |     |     |  // Set Interrupt Mask / NMI
*E919: 72 04 08       'r..'                    OIM     #$04,M0008  | |     |     |    |     |     |  // (0x0008) |= 0x04; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*E91C: BD F3 F9       '...'                    JSR     ZF3F9       | |     |     |    |     |     |  // Call 0xf3f9; Tastendruck; Rückgabe A
*E91F: 81 A2          '..'                     CMPA    #$A2        | |     |     |    |     |     |  // CMP A, 0xa2 Taste SEARCH
*E921: 27 BD          ''.'                     BEQ     ZE8E0       | |     |     |    | >---/     |  // if (z) == 1
*E923: 7E E1 90       '~..'                    JMP     ZE190       | |     |     |    |           |  // JP 0xe190->  LD SP = 0x0287; JP 0xe172;
                                                                   | |     |     |    V           |
// Call JP                                                         | |     |     |                |
*E926: 7E EA 94       '~..'            ZE926   JMP     ZEA94  <---/ |     |     |                |  // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                     |     |     |                |
// JP                                                                |     |     |                |
*E929: 08             '.'              ZE929   INX                   |     |     |                |  // IX++
*E92A: 9C 70          '.p'                     CPX     M0070       |     |     |                |  // cmp IX, (0x0070)
*E92C: 26 C6          '&.'                     BNE     ZE8F4    >---+     |     |                |  // if (z) == 0
*E92E: 96 2F          './'                     LDAA    M002F       |     |     |                |  // A = (0x002f); Zählervariable 0x1 ... 0x62
*E930: 91 72          '.r'                     CMPA    M0072       |     |     |                |  // CMP A, (0x0072)
*E932: 26 C0          '&.'                     BNE     ZE8F4    >---/     |     |                |  // if (z) == 0
*E934: 20 12          ' .'                     BRA     ZE948  >---\       |     |                |  // JR 0xe948
                                                                 |       |     |                |
// JP                                                            |       |     |                |
*E936: BD E2 4E       '..N'            ZE936   JSR     ZE24E      |   <---/     |                |  // Call 0xe24e; A++ Bereich: 0x01 bis 0x063
*E939: BD E2 1A       '...'                    JSR     ZE21A      |             |                |  // Call 0xe21a  Init Textlänge Zeiger
*E93C: DE 2D          '.-'                     LDX     M002D      |             |                |  // IX = (0x002d)
*E93E: 9C 70          '.p'                     CPX     M0070      |             |                |  // cmp IX, (0x0070)
*E940: 26 A6          '&.'                     BNE     ZE8E8      |             |          >---+  // if (z) == 0
*E942: 96 2F          './'                     LDAA    M002F      |             |              |  // A = (0x002f); Zählervariable 0x1 ... 0x62
*E944: 91 72          '.r'                     CMPA    M0072      |             |              |  // CMP A, (0x0072)
*E946: 26 A0          '&.'                     BNE     ZE8E8      |             |          >---/  // if (z) == 0
*E948: 0F             '.'              ZE948   SEI           <---/                              // Set Interrupt Mask / NMI
*E949: 72 04 08       'r..'                    OIM     #$04,M0008                               // (0x0008) |= 0x04; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*E94C: CE F8 1E       '...'                    LDX     #MF81E                                   // IX = 0xf81e; "< ........ NOT FOUND >
*E94F: BD ED 82       '...'                    JSR     ZED82                                    // Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
*E952: BD ED CD       '...'                    JSR     ZEDCD                                    // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*E955: CE 00 CF       '...'                    LDX     #M00CF                                   // IX = 0x00cf; Überflüssig wird überschrieben
*E958: 8D 4E          '.N'                     BSR     ZE9A8                                    // Call 0xe9a8; Fülle 0x00d8 ... 0x00df mit (0x004e ...) oder 0x20 wenn (0x004e ...) == 0
*E95A: BD EC F9       '...'                    JSR     ZECF9                                    // Call 0xecf9; Übergabe (0x0032)
*E95D: BD E9 26       '..&'                    JSR     ZE926                                    // Call 0xe926
*E960: 7E EA CD       '~..'                    JMP     ZEACD                                    // JP 0xeacd; Zeitschleife * 0x64;->RETURN
```

```
// JP Übergabe IX; Rückgabe A = 0x0 oder 0xff; IX = (0x00a9) oder wie Übergabe
*E963: 3C             '<'              ZE963   PSHX                                             // Push IX
*E964: DF A9          '..'                     STX     M00A9                                    // (0x00a9) = IX
*E966: CE 00 4E       '..N'                    LDX     #M004E                                   // IX = 0x004e
*E969: DF AB          '..'                     STX     M00AB                                    // (0x00ab) = IX; Zeiger auf 0x004e
*E96B: DE AB          '..'             ZE96B   LDX     M00AB               <---\                // IX = (0x00ab); hole Zeiger 0x004e oder
*E96D: 8C 00 56       '..V'                    CPX     #M0056                  |                // cmp IX, 0x0056
*E970: 27 1E          ''.'                     BEQ     ZE990 >-----\           |                // if (z) == 1; Geradziehen SP, Lösche A, Init IX
*E972: A6 00          '..'                     LDAA    ,X          |           |                // A = (IX)
*E974: 27 1A          ''.'                     BEQ     ZE990 >-----+           |                // if (z) == 1; Geradziehen SP, Lösche A, Init IX
*E976: 08             '.'                      INX                 |           |                // IX++
*E977: DF AB          '..'                     STX     M00AB       |           |                // (0x00ab) = IX; Lege Zeiger 0x0055, 0x0056 ab
*E979: BD E9 9A       '...'                    JSR     ZE99A       |           |  >---\         // Call 0xe99a; Übergabe A, Rückgabe A &= 0xdf; if A > 0x60 and < 0x7b then "a ... z" -> "A ... Z"
*E97C: 16             '.'                      TAB                 |           |      |         // B = A
*E97D: DE A9          '..'                     LDX     M00A9       |           |      |         // IX = (0x00a9); Zeiger auf 0x004d
*E97F: A6 00          '..'                     LDAA    ,X          |           |      |         // A = (IX)
*E981: BD E9 9A       '...'                    JSR     ZE99A       |           |  >---+         // Call 0xe99a; Übergabe A, Rückgabe A &= 0xdf; if A > 0x60 and < 0x7b then "a ... z" -> "A ... Z"
*E984: 11             '.'                      CBA                 |           |      |         // cmp A  B
*E985: 26 0F          '&.'                     BNE     ZE996       | >---\     |      |         // if (z) == 0; Pop IX, A = 0xff, RETURN;
*E987: 9C 27          '.''                     CPX     M0027       |     |     |      |         // cmp IX, (0x0027)
*E989: 27 0B          ''.'                     BEQ     ZE996       | >---+     |      |         // if (z) == 1; Pop IX, A = 0xff, RETURN;
*E98B: 08             '.'                      INX                 |     |     |      |         // IX++
*E98C: DF A9          '..'                     STX     M00A9       |     |     |      |         // (0x00a9) = IX;  Zeiger 0x004d + n
*E98E: 20 DB          ' .'                     BRA     ZE96B       |     | >---/      |         // JR wiederholen bis Treffer oder Abbruch
// Geradziehen SP, Lösche A, Init IX                               |     |            |
*E990: 31             '1'              ZE990   INS           <-----/     |            |         // SP++
*E991: 31             '1'                      INS                       |            |         // SP++; stelle SP wieder her
*E992: DE A9          '..'                     LDX     M00A9             |            |         // IX = (0x00a9)
*E994: 4F             'O'                      CLRA                      |            |         // A = 0
*E995: 39             '9'                      RTS                       |            |         // RETURN; Rückgabe A = 0; IX = (0x00a9)
                                                                         |            |
// JP Rückgabe A = 0xff IX unverändert                                   |            |
*E996: 38             '8'              ZE996   PULX                <-----/            |         // Pop IX
*E997: 86 FF          '..'                     LDAA    #$FF                          |         // A = 0xff
*E999: 39             '9'                      RTS                                   |         // RETURN; A = 0xff
                                                                                     |
// Call Übergabe A, Rückgabe A &= 0xdf; Klein- -> Großschreibung                     |
*E99A: 81 61          '.a'             ZE99A   CMPA    #$61                    <---/         // CMP A, 0x61; "a"
*E99C: 25 06          '%.'                     BCS     ZE9A4 >---\                           // if (c) == 1 RETURN
*E99E: 81 7A          '.z'                     CMPA    #$7A      |                           // CMP A, 0x7a; "z"
*E9A0: 22 02          '".'                     BHI     ZE9A4 >---+                           // if (c) & (z) == 0 RETURN
*E9A2: 84 DF          '..'                     ANDA    #$DF      |                           // A &= 0xdf Maskiere-> Lösche bit 5
*E9A4: 39             '9'              ZE9A4   RTS           <---/                           // RETURN

// Call Fülle 0x00d8 ... 0x00df mit (0x004e ...) oder 0x20 wenn (0x004e ...) == 0
*E9A5: CE 00 D8       '...'            ZE9A5   LDX     #M00D8                                // IX = 0x00d8
```

```
*E9A8: DF 73          '.s'              ZE9A8   STX     M0073                          // (0x0073) = IX
*E9AA: CE 00 4E       '..N'                     LDX     #M004E                         // IX = 0x004e
*E9AD: A6 00          '..'              ZE9AD   LDAA    ,X      <---\                  // A = (IX)
*E9AF: 3C             '<'                       PSHX              |                  // Push IX
*E9B0: 26 02          '&.'                      BNE     ZE9B4 >---\ |                  // if (z) == 0; A == 0 then A = 0x20
*E9B2: 86 20          '. '                      LDAA    #$20      | |                  // A = 0x20
*E9B4: DE 73          '.s'              ZE9B4   LDX     M0073 <---/ |                  // IX = (0x0073); 0x00d8
*E9B6: A7 00          '..'                      STAA    ,X          |                  // (IX) = A; 0x00d8 = > 0 oder 0x20
*E9B8: 08             '.'                       INX               |                  // IX++; 0x00d9
*E9B9: DF 73          '.s'                      STX     M0073       |                  // (0x0073) = IX
*E9BB: 38             '8'                       PULX              |                  // Pop IX; 0x004e ...
*E9BC: 08             '.'                       INX               |                  // IX++; 0x004f ...
*E9BD: 8C 00 56       '..V'                     CPX     #M0056      |                  // cmp IX, 0x0056; bi x00056
*E9C0: 26 EB          '&.'                      BNE     ZE9AD   >---/                  // if (z) == 0 Schleife, von 0x004e ... 0x0056
*E9C2: 39             '9'                       RTS                                    // RETURN

// Call Ermittle Textanfang-Ende GTX
*E9C3: DE 29          '.)'              ZE9C3   LDX     M0029                          // IX = (0x0029); Zeiger Text-aktual Position
*E9C5: 9C 25          '.%'              ZE9C5   CPX     M0025 <---\                    // cmp IX, (0x0025); Zeiger Text-Ende
*E9C7: 23 08          '#.'                      BLS     ZE9D1     |  >---\              // if (c) | (z) == 1
*E9C9: 09             '.'                       DEX               |      |             // IX--
*E9CA: A6 00          '..'                      LDAA    ,X        |      |             // A = (IX)
*E9CC: 81 8D          '..'                      CMPA    #$8D      |      |             // CMP A, 0x8d; Klartext-Zeichen Textende
*E9CE: 26 F5          '&.'                      BNE     ZE9C5 >---/      |             // if (z) == 0 Schleife
*E9D0: 08             '.'                       INX                      |             // IX++
*E9D1: 4F             'O'               ZE9D1   CLRA          <----------+             // A = 0
*E9D2: 5F             '_'                       CLRB                     |             // B = 0
*E9D3: DD 6E          '.n'                      STD     M006E            |             // (0x006e) = D = 0x0000
*E9D5: A6 00          '..'              ZE9D5   LDAA    ,X      <----\   |             // A = (IX)
*E9D7: 81 8D          '..'                      CMPA    #$8D         |   |             // CMP A, 0x8d
*E9D9: 26 01          '&.'                      BNE     ZE9DC >---\  |   |             // if (z) == 0
*E9DB: 39             '9'                       RTS               |  |   |             // RETURN
                                                                  |  |   |
// JP                                                             |  |   |
*E9DC: 84 7F          '..'              ZE9DC   ANDA    #$7F  <---/  |   |             // A &= 0x7f Maskiere
*E9DE: A7 00          '..'                      STAA    ,X           |   |             // (IX) = A
*E9E0: 81 20          '. '                      CMPA    #$20         |   |             // CMP A, 0x20
*E9E2: 26 02          '&.'                      BNE     ZE9E6 >---\  |   |             // if (z) == 0
*E9E4: DF 6E          '.n'                      STX     M006E     |  |   |             // (0x006e) = IX
*E9E6: 08             '.'               ZE9E6   INX           <---/  |   |             // IX++
*E9E7: 5C             '\'                       INCB                 |   |             // B++
*E9E8: D1 3C          '.<'                      CMPB    M003C        |   |             // cmp B, (0x003c)
*E9EA: 23 E9          '#.'                      BLS     ZE9D5    >---/   |             // if (c) | (z) == 1 Schleife
*E9EC: 3C             '<'                       PSHX                     |             // Push IX                          <-------\
*E9ED: DE 6E          '.n'                      LDX     M006E            |             // IX = (0x006e)                            |
*E9EF: 26 05          '&.'                      BNE     ZE9F6    >----\  |             // if (z) == 0                             |
*E9F1: 38             '8'                       PULX                 |  |             // Pop IX                                  |
*E9F2: 09             '.'                       DEX                  |  |             // IX--                                     |
*E9F3: 09             '.'                       DEX                  |  |             // IX--                                     |
*E9F4: 20 02          ' .'                      BRA     ZE9F8 >---\  |  |             // JR 0xe9f8                               |
                                                                  |  |  |                                                        |
// JP                                                             |  |  |                                                        |
*E9F6: 33             '3'               ZE9F6   PULB          <-/  |  |             // Pop B                                   |
*E9F7: 33             '3'                       PULB                |  |             // Pop B; Realisiert das Pop IX-LOW Teil --/
*E9F8: A6 00          '..'              ZE9F8   LDAA    ,X    <---/  |             // A = (IX)
*E9FA: 8A 80          '..'                      ORAA    #$80         |             // A |= 0x80; Setze Bit 7
*E9FC: A7 00          '..'                      STAA    ,X           |             // (IX) = A
*E9FE: 08             '.'                       INX                  |             // IX++
*E9FF: 20 D0          ' .'                      BRA     ZE9D1     >---/             // JR 0e9d1

// Call JP  Textanfang-Ende GTX gefunden?
*EA01: 8D C0          '..'              EA01    BSR     ZE9C3 <----\                   // Call 0xe9c3; Ermittle Textanfang-Ende GTX
*EA03: 9C 27          '.''                      CPX     M0027      |                   // cmp IX, (0x0027)
*EA05: 24 05          '$.'                      BCC     ZEA0C      | >---\             // if (c) == 0; Abbruch
*EA07: 08             '.'                       INX                |     |             // IX++
*EA08: DF 29          '.)'                      STX     M0029      |     |             // (0x0029) = IX
*EA0A: 20 F5          ' .'                      BRA     ZEA01 >----/     |             // JR Schleife
                                                                         |
*EA0C: 7E E5 BC       '~..'             EA0C    JMP     ZE5BC       <----/             // JP 0xe5bc Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b) RETURN

// Call Speicher füllen mit " ", Übergabe B Rundenanzahl
*EA0F: DE 2D          '.-'              EA0F    LDX     M002D                          // IX = (0x002d)
*EA11: 36             '6'                       PSHA                               // Push A
*EA12: 86 20          '. '                      LDAA    #$20                       // A = 0x20
*EA14: A7 00          '..'              EA14    STAA    ,X    <---\                // (IX) = A; Fülle IX + B mit 0x20 " "
*EA16: 08             '.'                       INX             |                  // IX++
*EA17: 5A             'Z'                       DECB            |                  // B--
*EA18: 26 FA          '&.'                      BNE     EA14 >---/                 // if (z) == 0 Schleife B
*EA1A: DF 2D          '.-'                      STX     M002D                      // (0x002d) = IX
*EA1C: 32             '2'                       PULA                               // Pop A
*EA1D: 39             '9'                       RTS                                // RETURN

// Call Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*EA1E: 4F             'O'               EA1E    CLRA                               // A = 0
*EA1F: DD 6E          '.n'                      STD     M006E                      // (0x006e) = D &= 0x00ff
*EA21: DE 23          '.#'                      LDX     M0023                      // IX = (0x0023)
*EA23: 3A             ':'                       ABX                                // IX += B
*EA24: 20 07          ' .'                      BRA     EA2D >---\                 // JR 0xea2s
// Call Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D  |
*EA26: DD 6E          '.n'              EA26    STD     M006E     |                // (0x006e) = D
*EA28: D3 23          '.#'                      ADDD    M0023     |                // D += (0x0023)
*EA2A: 37             '7'                       PSHB              |                // Push B
*EA2B: 36             '6'                       PSHA              |                // Push A
*EA2C: 38             '8'                       PULX              |                // Pop IX; IX = B, A
*EA2D: 8C 1F FF       '...'             EA2D    CPX     #M1FFF <--/                // cmp IX, 0x1fff
```

```
*EA30: 22 1B          '".'                    BHI     ZEA4D >---\                     // if (c) & (z) == 0 Prüfung Textspeicher voll?
*EA32: 9F 6C          '.l'                    STS     M006C     |                     // (0x006C) = SP;  Sicher Stackpointer
*EA34: 35             '5'                     TXS               |                     // SP = (IX); Setze SP neu auf ...
*EA35: 31             '1'                     INS               |                     // SP++; 2 Byte Adresse
*EA36: DC 27          '.''                    LDD     M0027     |                     // D = (0x0027)
*EA38: D3 6E          '.n'                    ADDD    M006E     |                     // D += (0x006e)
*EA3A: DD 27          '.''                    STD     M0027     |                     // (0x0027) += (0x063)
*EA3C: DE 23          '.#'                    LDX     M0023     |                     // IX = (0x0023)
*EA3E: 9F 23          '.#'                    STS     M0023     |                     // (0x0023) = SP;  (0x0023) = berechneter IX
*EA40: E6 00          '..'            ZEA40   LDAB    ,X        | <---\               // B = (IX)
*EA42: 37             '7'                     PSHB              |     |               // Push B
*EA43: 09             '.'                     DEX               |     |               // IX--
*EA44: 9C 2D          '.-'                    CPX     M002D     |     |               // cmp IX, (0x002d)
*EA46: 24 F8          '$.'                    BCC     ZEA40     | >---/               // if (c) == 0
*EA48: 9E 6C          '.l'                    LDS     M006C     |                     // SP = (0x006c)
*EA4A: DC 6E          '.n'                    LDD     M006E     |                     // D = (0x006e)
*EA4C: 39             '9'                     RTS               |                     // RETURN
                                                                |
// JP Textspeicher voll                                         |
*EA4D: 8E 02 87       '...'           ZEA4D   LDS     #M0287 <--/                     // SP = 0x0287; SP neu festlegen
*EA50: 86 04          '..'                    LDAA    #$04                            // A = 0x04
*EA52: 97 08          '..'                    STAA    M0008                           // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*EA54: BD EA 99       '...'                   JSR     ZEA99                           // Call 0xea99; Modem ON Übergabe A Port-1x, B Zeitschleife 60-100-60ms
*EA57: CE F8 B2       '...'                   LDX     #MF8B2                          // IX = 0xf8b2; "*** MEMORY FULL ***"
*EA5A: BD EC C5       '...'                   JSR     ZECC5                           // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*EA5D: 72 80 02       'r..'                   OIM     #$80,M0002                      // (Port-1 Data) |= 0x80; Port-17 LED ON
*EA60: 7E E1 5E       '~.^'                   JMP     ZE15E                           // JP 0xe15e

// Call Übergabe D
*EA63: 4F             'O'             ZEA63   CLRA                                    // A = 0
*EA64: DD 6E          '.n'                    STD     M006E                           // (0x006e) = D
*EA66: DC 27          '.''                    LDD     M0027                           // D = (0x0027); aktual Position
*EA68: 93 6E          '.n'                    SUBD    M006E                           // D -= (0x006e); Länge
*EA6A: DD 27          '.''                    STD     M0027                           // (0x0027) = D; (0x0027) -= (0x006e) & 0x00ff)
*EA6C: DC 6E          '.n'                    LDD     M006E                           // D = (0x006e)
*EA6E: DE 2D          '.-'                    LDX     M002D                           // IX = (0x002d)
*EA70: 3A             ':'                     ABX                                     // IX += B
*EA71: 20 09          ' .'                    BRA     ZEA7C >--\                      // JR 0xea7c
                                                              |
// Call Übergabe D                                            |
*EA73: DD 6E          '.n'            ZEA73   STD     M006E     |                     // (0x006e) = D
*EA75: DC 2D          '.-'                    LDD     M002D     |                     // D = (0x002d); Zeiger Text-Start
*EA77: D3 6E          '.n'                    ADDD    M006E     |                     // D += (0x006e); Text-Ende
*EA79: 37             '7'                     PSHB              |                     // Push B
*EA7A: 36             '6'                     PSHA              |                     // Push A
*EA7B: 38             '8'                     PULX              |                     // Pop IX; IX = D
*EA7C: DC 23          '.#'            ZEA7C   LDD     M0023 <--/                      // D = (0x0023)
*EA7E: 93 6E          '.n'                    SUBD    M006E                           // D -= (0x006e); Länge
*EA80: DD 23          '.#'                    STD     M0023                           // (0x0023) = D; (0x0023) -= (0x002d) + (0x006e)
*EA82: 9F 6C          '.l'                    STS     M006C                           // (0x006c) = SP; Sichere SP
*EA84: 35             '5'                     TXS                                     // SP = (IX); Aus Push B, A -> Pop IX
*EA85: DE 2D          '.-'                    LDX     M002D                           // IX = (0x002d); Zeiger Text-Ende
*EA87: 33             '3'             ZEA87   PULB          <--\                      // Pop B; Wert aus SP = (IX) siehe 0xea7b
*EA88: E7 00          '..'                    STAB    ,X        |                     // (IX) = B
*EA8A: 08             '.'                     INX               |                     // IX++
*EA8B: 9C 23          '.#'                    CPX     M0023     |                     // cmp IX, (0x0023)
*EA8D: 23 F8          '#.'                    BLS     ZEA87 >--/                      // if (c) | (z) == 1 Schleife
*EA8F: 9E 6C          '.l'                    LDS     M006C                           // SP = (0x006c); SP Wiederherstellen
*EA91: DC 6E          '.n'                    LDD     M006E                           // D = (0x006e)
*EA93: 39             '9'                     RTS                                     // RETURN

// Call Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*EA94: 4F             'O'             ZEA94   CLRA                                    // A = 0x00 Data an Modem
*EA95: C6 0A          '..'                    LDAB    #$0A                            // B = 0x0a Zeitschleife
*EA97: 20 1B          ' .'                    BRA     ZEAB4 >---\                     // JR 0xeab4; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                              |
// Call Modem ON Übergabe A Port-1x, B Zeitschleife 60-100-60ms |
*EA99: 86 10          '..'            ZEA99   LDAA    #$10      |                     // A = 0x10 Data an Modem
*EA9B: C6 06          '..'                    LDAB    #$06      |                     // B = 0x06 Zeitschleife 60ms
*EA9D: 8D 15          '..'                    BSR     ZEAB4 >---+                     // Call 0xeab4; Modem ON Übergabe A Port-1x, B Zeitschleife 60ms
*EA9F: 4F             'O'                     CLRA              |                     // A = 0x00 Data an Modem
*EAA0: C6 0A          '..'                    LDAB    #$0A      |                     // B = 0x0a Zeitschleife 100ms
*EAA2: 8D 10          '..'                    BSR     ZEAB4 >---+                     // Call 0xeab4; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*EAA4: 86 0A          '..'                    LDAA    #$0A      |                     // A = 0x0a Data an Modem
*EAA6: C6 06          '..'                    LDAB    #$06      |                     // B = 0x06 Zeitschleife 60ms
*EAA8: 20 0A          ' .'                    BRA     ZEAB4 >---+                     // JR 0xeab4; Modem ON Übergabe A Port-1x, B Zeitschleife 60ms
                                                              |
// Call Modem ON Übergabe A Port-14, B Zeitschleife  30ms     |
*EAAA: 86 10          '..'            ZEAAA   LDAA    #$10      |                     // A = 0x10 Data an Modem
*EAAC: C6 03          '..'                    LDAB    #$03      |                     // B = 0x03 Zeitschleife 30ms
*EAAE: 20 04          ' .'                    BRA     ZEAB4 >---+                     // JR 0xeab4; Modem ON Übergabe A Port-1x, B Zeitschleife 30ms
                                                              |
// Call Modem ON Übergabe A Port-14, B Zeitschleife 10ms      |
*EAB0: 86 10          '..'            ZEAB0   LDAA    #$10      |                     // A = 0x10
*EAB2: C6 01          '..'                    LDAB    #$01      |                     // B = 0x01 Wiederholung Zeitschleife 10ms
                                                              |
// Call JR Modem ON Übergabe A Port-1x, B Zeitschleife *10ms  |
*EAB4: 97 33          '.3'            ZEAB4   STAA    M0033 <---/                     // (0x0033) = A; 0x00, 0x0a, 0x10
*EAB6: 7D 00 3B       '}.;'                   TST     >M003B                          // (0x003b) == 0?;  MSB == 1 n-Flag (0x003b) == 0 z-Flag
*EAB9: 27 11          ''.'                    BEQ     ZEACC     >---\                 // if (z) == 1 RETURN
*EABB: 71 F9 02       'q..'                   AIM     #$F9,M0002    |                 // (Port-1 Data) &= 0xf9; außer Port-11 Modem ON/OFF, Port-12 Speed Baud
*EABE: 96 02          '..'                    LDAA    M0002         |                 // A = Port-1 Data
*EAC0: 84 EF          '..'                    ANDA    #$EF          |                 // A &= 0xef; außer Port-14 TX
*EAC2: 9A 33          '.3'                    ORAA    M0033         |                 // A |= (0x0033); A |= 0x00; 0x0a; 0x10; Port-10 Port-12 Port-14;
*EAC4: 97 02          '..'                    STAA    M0002         |                 // (Port-1 Data) = A; entsprechend Port "0" oder "1"
```

```
*EAC6: BD EA CF       '...'                   JSR     ZEACF       |                   // Call 0xeacf Zeitschleife 10ms; Übergabe B
*EAC9: 72 02 02       'r..'                   OIM     #$02,M0002  |                   // (Port-1 Data) |= 0x02 Port-11 Modem ON
*EACC: 39             '9'             ZEACC   RTS             <---/                   // RETURN

// Call JP Zeitschleife 0x64 * 10 ms
*EACD: C6 64          '.d'            ZEACD   LDAB    #$64                            // B = 0x64; Zeitschleife 1 Sekunde
// Call JP Zeitschleife B * Wiederholungen; Übergabe B
*EACF: BD EA D6       '...'           ZEACF   JSR     ZEAD6 <---\                     // Call 0xead6;  Zeitschleife 10ms;
*EAD2: 5A             'Z'                     DECB            |                       // B--
*EAD3: 26 FA          '&.'                    BNE     ZEACF >---/                     // if (z) == 0; Schleifenzähler  Wiederholungen der Zeitschleife
*EAD5: 39             '9'                     RTS                                     // RETURN

// Call/JP Zeitschleife 10 ms
*EAD6: 3C             '<'             ZEAD6   PSHX                                    // Push IX
*EAD7: CE 09 BE       '...'                   LDX     #M09BE                          // IX = 0x09be
*EADA: 09             '.'             ZEADA   DEX             <---\                   // IX--
*EADB: 26 FD          '&.'                    BNE     ZEADA >---/                     // if (z) == 0
*EADD: 38             '8'                     PULX                                    // Pop IX
*EADE: 39             '9'                     RTS                                     // RETURN

// Call IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*EADF: BD E2 12       '...'           ZEADF   JSR     ZE212                           // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*EAE2: 6D 00          'm.'                    TST     ,X                              // (IX) == 0; bit7 == 1
*EAE4: 2B 03          '+.'                    BNI     ZEAE9 >---\                     // if (n) == 1  A = 0x00 RETURN
*EAE6: 86 FF          '..'                    LDAA    #$FF        |                   // A = 0xff
*EAE8: 39             '9'                     RTS                 |                   // RETURN Rückgabe 0xff
                                                                  |
// A = 0x00; Return                                               |
*EAE9: 4F             'O'             ZEAE9   CLRA            <---/                   // A = 0
*EAEA: 39             '9'                     RTS                                     // RETURN

// Call A Division 10; Übergabe A, Rückgabe A, B; A = "0" ... "9"; B Anzahl der Division
*EAEB: C6 2F          './'            ZEAEB   LDAB    #$2F                            // B = 0x2f
*EAED: 5C             '\'             ZEAED   INCB            <---\                   // B++
*EAEE: 80 0A          '..'                    SUBA    #$0A        |                   // A-= 0x0a
*EAF0: 24 FB          '$.'                    BCC     ZEAED >---/                     // if (c) == 0 Schleife; Abbruch < 0 bzw. neagtives Ergebnis
*EAF2: 8B 3A          '.:'                    ADDA    #$3A                            // A += 0x3a; bei 0 A = "0" 0x30
*EAF4: 39             '9'                     RTS                                     // RETURN

// Call LCD Ausgaben, Übergabe B, (0x005a);
*EAF5: D7 66          '.f'            ZEAF5   STAB    M0066                           // (0x0066) = B; 0x0c, 0x0e, 0x14
*EAF7: 96 37          '.7'                    LDAA    M0037                           // A = (0x0037)
*EAF9: 36             '6'                     PSHA                                    // Push A
*EAFA: 86 FF          '..'                    LDAA    #$FF                            // A = 0xff
*EAFC: 97 37          '.7'                    STAA    M0037                           // (0x0037) = A
*EAFE: 96 66          '.f'            ZEAFE   LDAA    M0066               <---\       // A = (0x0066); Wert aus Übergabe; 0x0c, 0x0e, 0x14
*EB00: 4C             'L'                     INCA                            |       // A++; 0x0d, 0x0f, 0x15
*EB01: 97 32          '.2'                    STAA    M0032                   |       // (0x0032) = A
*EB03: 8A 80          '..'                    ORAA    #$80                    |       // A |= 0x80; Setze Bit7; DD-RAM LCD, 0x8d, 0x8f, 0x95
*EB05: BD EC 37       '..7'                   JSR     ZEC37                   |       // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*EB08: BD F3 F9       '...'                   JSR     ZF3F9                   |       // Call 0xf3f9; Tastendruck; Rückgabe A
*EB0B: 91 5A          '.Z'                    CMPA    M005A                   |       // CMP A, (0x005a)
*EB0D: 27 21          ''!'                    BEQ     ZEB30       >--\        |       // if (z) == 1
*EB0F: 91 5B          '.['                    CMPA    M005B          |        |       // CMP A, (0x005b)
*EB11: 27 1D          ''.'                    BEQ     ZEB30       >--+        |       // if (z) == 1
*EB13: 81 30          '.0'                    CMPA    #$30           |        |       // CMP A, 0x30; "0"
*EB15: 25 2F          '%/'                    BCS     ZEB46 >--\     |        |       // if (c) == 1 Sprungverteiler 17, 18 Tastaturabfrage
*EB17: 81 39          '.9'                    CMPA    #$39     |     |        |       // CMP A, 0x39; "9"
*EB19: 22 2B          '"+'                    BHI     ZEB46 >--+     |        |       // if (c) & (z) == 0 Sprungverteiler 17, 18 Tastaturabfrage
*EB1B: CE 00 CD       '...'                   LDX     #M00CD   |     |        |       // IX = 0x00cd
*EB1E: D6 66          '.f'                    LDAB    M0066    |     |        |       // B = (0x0066)
*EB20: 3A             ':'                     ABX              |     |        |       // IX += B; 0x00cd + B
*EB21: E6 01          '..'                    LDAB    $01,X    |     |        |       // B = (IX + 0x01); 0x00cd + B + 1
*EB23: E7 00          '..'                    STAB    ,X       |     |        |       // (IX) = B; (0x00cd) = B
*EB25: A7 01          '..'                    STAA    $01,X    |     |        |       // (IX + 0x01) = (0x00cd + B + 1) = A
*EB27: 86 29          '.)'                    LDAA    #$29     |     |        |       // A = 0x29; 40 Elemente
*EB29: 97 32          '.2'                    STAA    M0032    |     |        |       // (0x0032) = A
*EB2B: BD EC F9       '...'                   JSR     ZECF9    |     |        |       // Call 0xecf9; Übergabe (0x0032)
*EB2E: 20 CE          ' .'                    BRA     ZEAFE    |     |  >---+        // JR 0xeafe; Schleife, Abbruch
                                                               |     |               |
// JP  (0x00nn)                                                |     |               |
*EB30: CE 00 CD       '...'           ZEB30   LDX     #M00CD   | <---/               |       // IX = 0x00cd
*EB33: D6 66          '.f'                    LDAB    M0066    |                     |       // B = (0x0066)
*EB35: 3A             ':'                     ABX              |                     |       // IX += B: 0x00cd + (0x0066)
*EB36: A6 00          '..'                    LDAA    ,X       |                     |       // A = (IX)
*EB38: 84 0F          '..'                    ANDA    #$0F     |                     |       // A &= 0x0f Maskiere
*EB3A: C6 0A          '..'                    LDAB    #$0A     |                     |       // B = 0x0a
*EB3C: 3D             '='                     MUL              |                     |       // D = A * 10    Schrittanzahl dekadisch; 0x00, 0x10, 0x20, ... 0xf0
*EB3D: A6 01          '..'                    LDAA    $01,X    |                     |       // A = (IX + 0x01)
*EB3F: 84 0F          '..'                    ANDA    #$0F     |                     |       // A &= 0x0f Maskiere
*EB41: 1B             '.'                     ABA              |                     |       // A += B
*EB42: 33             '3'                     PULB             |                     |       // Pop B; mit Push A gesicherter Wert aus (0x0037)
*EB43: D7 37          '.7'                    STAB    M0037    |                     |       // (0x0037) = B
*EB45: 39             '9'                     RTS              |                     |       // RETURN
                                                               |                     |
// JP "0" "9"                                                  |                     |
*EB46: BD EA 94       '...'           ZEB46   JSR     ZEA94 <--/                     |       // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*EB49: 20 B3          ' .'                    BRA     ZEAFE                  >---/       // JR 0xeafe

// Call Hex-Eingabe, Übergabe B
*EB4B: D7 66          '.f'            ZEB4B   STAB    M0066                           // (0x0066) = B
*EB4D: 96 36          '.6'                    LDAA    M0036                           // A = (0x0036); Tastatur z/s
*EB4F: 36             '6'                     PSHA                                    // Push A; 0x0036 auf den Stack
*EB50: 96 37          '.7'                    LDAA    M0037                           // A = (0x0037)
*EB52: 36             '6'                     PSHA                                    // Push A; 0x0037 auf den Stack
```

```
*EB53: 86 FF          '..'                    LDAA    #$FF                                    // A = 0xff
*EB55: 97 36          '.6'                    STAA    M0036                                   // (0x0036) = A; fülle mit 0xff
*EB57: 97 37          '.7'                    STAA    M0037                                   // (0x0037) = A; fülle mit 0xff
*EB59: BD F4 37       '..7'           ZEB59   JSR     ZF437 <---\                             // Call 0xf437; Tastaturabfrage Rückgabe A+B, IX = 0x009d, z-Flag
*EB5C: 26 FB          '&.'                    BNE     ZEB59 >---/                             // if (z) == 0 Warte auf Flag Änderung aus Funktion 0xf437
// JP Wiederhole Hex-Eingabe
*EB5E: 96 66          '.f'            ZEB5E   LDAA    M0066           <---\                   // A = (0x0066)
*EB60: 4C             'L'                     INCA                        |                   // A++
*EB61: 97 32          '.2'                    STAA    M0032               |                   // (0x0032) = A
*EB63: 8A 80          '..'                    ORAA    #$80                |                   // A |= 0x80; setze Bit7, DD-RAM LCD
*EB65: BD EC 37       '..7'                   JSR     ZEC37               |                   // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*EB68: BD F3 F9       '...'                   JSR     ZF3F9               |                   // Call 0xf3f9; Tastendruck; Rückgabe A
*EB6B: 81 0D          '..'                    CMPA    #$0D                |                   // CMP A, 0x0d; WR/ZV Enter
*EB6D: 27 25          ''%'                    BEQ     ZEB94   >---\       |                   // if (z) == 1
*EB6F: 81 30          '.0'                    CMPA    #$30        |       |                   // CMP A, 0x30; "0"
*EB71: 25 47          '%G'                    BCS     ZEBBA       |       |   >---\           // if (c) == 1
*EB73: 81 39          '.9'                    CMPA    #$39        |       |       |           // CMP A, 0x39; "9"
*EB75: 23 08          '#.'                    BLS     ZEB7F >---\ |       |       |           // if (c) | (z) == 1
*EB77: 81 41          '.A'                    CMPA    #$41      | |       |       |           // CMP A, 0x41; "A"
*EB79: 25 3F          '%?'                    BCS     ZEBBA     | |       |   >---+           // if (c) == 1
*EB7B: 81 46          '.F'                    CMPA    #$46      | |       |       |           // CMP A, 0x46; "F"
*EB7D: 22 3B          '";'                    BHI     ZEBBA     | |       |   >---+           // if (c) & (z) == 0
                                                                | |       |       |
// JP "9"                                                       | |       |       |
*EB7F: CE 00 CD       '...'           ZEB7F   LDX     #M00CD <--/ |       |       |           // IX = 0x00cd
*EB82: D6 66          '.f'                    LDAB    M0066       |       |       |           // B = (0x0066)
*EB84: 3A             ':'                     ABX                 |       |       |           // IX += B
*EB85: E6 01          '..'                    LDAB    $01,X       |       |       |           // B = (IX + 0x01)
*EB87: E7 00          '..'                    STAB    ,X          |       |       |           // (IX) = B
*EB89: A7 01          '..'                    STAA    $01,X       |       |       |           // (IX + 0x01) = A; "1" ... "9"
*EB8B: 86 29          '.)'                    LDAA    #$29        |       |       |           // A = 0x29; 40 Elemente
*EB8D: 97 32          '.2'                    STAA    M0032       |       |       |           // (0x0032) = A
*EB8F: BD EC F9       '...'                   JSR     ZECF9       |       |       |           // Call 0xecf9; Übergabe (0x0032)
*EB92: 20 CA          ' .'                    BRA     ZEB5E       |   >---+       |           // JR 0xeb5e; Hex-Eingabe
                                                                  |       |       |
// JP "WR/ZV"                                                     |       |       |
*EB94: CE 00 CD       '...'           ZEB94   LDX     #M00CD  <---/       |       |           // IX = 0x00cd
*EB97: D6 66          '.f'                    LDAB    M0066               |       |           // B = (0x0066)
*EB99: 3A             ':'                     ABX                         |       |           // IX += B
*EB9A: A6 00          '..'                    LDAA    ,X                  |       |           // A = (IX)
*EB9C: 80 30          '.0'                    SUBA    #$30                |       |           // A-= 0x30
*EB9E: 81 09          '..'                    CMPA    #$09                |       |           // CMP A, 0x09; "9" = 0x39
*EBA0: 23 02          '#.'                    BLS     ZEBA4 >---\         |       |           // if (c) | (z) == 1
*EBA2: 80 07          '..'                    SUBA    #$07      |         |       |           // A-= 0x07; "A" ... "F"
*EBA4: 48             'H'             ZEBA4   ASLA          <---/         |       |           // c <- A7 ... A0 <- 0 Shift left
*EBA5: 48             'H'                     ASLA                        |       |           // c <- A7 ... A0 <- 0 Shift left
*EBA6: 48             'H'                     ASLA                        |       |           // c <- A7 ... A0 <- 0 Shift left
*EBA7: 48             'H'                     ASLA                        |       |           // c <- A7 ... A0 <- 0 Shift left
*EBA8: E6 01          '..'                    LDAB    $01,X               |       |           // B = (IX + 0x01)
*EBAA: C0 30          '.0'                    SUBB    #$30                |       |           // B -= 0x30
*EBAC: C1 09          '..'                    CMPB    #$09                |       |           // cmp B, 0x09; "9"
*EBAE: 23 02          '#.'                    BLS     ZEBB2 >---\         |       |           // if (c) | (z) == 1
*EBB0: C0 07          '..'                    SUBB    #$07      |         |       |           // B -= 0x07; "A" ... "F"
*EBB2: 1B             '.'             ZEBB2   ABA           <---/         |       |           // A += B
*EBB3: 33             '3'                     PULB                        |       |           // Pop B
*EBB4: D7 37          '.7'                    STAB    M0037               |       |           // (0x0037) = B
*EBB6: 33             '3'                     PULB                        |       |           // Pop B
*EBB7: D7 36          '.6'                    STAB    M0036               |       |           // (0x0036) = B; Tastatur z/s
*EBB9: 39             '9'                     RTS                         |       |           // RETURN
                                                                          |       |
// JP "0" oder "A" oder "F"                                               |       |
*EBBA: BD EA 94       '...'           ZEBBA   JSR     ZEA94               |   <---/           // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*EBBD: 20 9F          ' .'                    BRA     ZEB5E           >---/                   // JR 0xeb5e; Hex-Eingabe

// Call Init 0x0046 .. 0x004d; Zeiger
*EBBF: CE 00 46       '..F'           ZEBBF   LDX     #M0046                                  // IX = 0x0046
*EBC2: 6F 00          'o.'            ZEBC2   CLR     ,X      <---\                           // (IX) = 0
*EBC4: 08             '.'                     INX                 |                           // IX++
*EBC5: 8C 00 4E       '..N'                   CPX     #M004E      |                           // cmp IX, 0x004e
*EBC8: 26 F8          '&.'                    BNE     ZEBC2   >---/                           // if (z) == 0 Schleife Fülle 0x0046 ... 0x004d mit 0x0
*EBCA: 09             '.'                     DEX                                             // IX--; 0x004d
*EBCB: DF AB          '..'                    STX     M00AB                                   // (0x00ab) = IX; Zeiger 0x00ab
*EBCD: CE 00 55       '..U'                   LDX     #M0055                                  // IX = 0x0055
*EBD0: DF A9          '..'                    STX     M00A9                                   // (0x00a9) = IX; Zeiger 0x0055
*EBD2: CE 00 08       '...'                   LDX     #M0008                                  // IX = 0x0008
*EBD5: DF AD          '..'                    STX     M00AD                                   // (0x00ad) = IX; Zeiger 0x0008
*EBD7: 7E F4 EB       '~..'                   JMP     ZF4EB                                   // JP 0xf4eb; Kopiere (0x00a9)->byte->(0x00ab); Adresse--; Schleifenzähler: (0x00ad)--

// Call Lösche 0x004e ... 0x0055
*EBDA: CE 00 4E       '..N'           ZEBDA   LDX     #M004E                                  // IX = 0x004e
*EBDD: 6F 00          'o.'            ZEBDD   CLR     ,X      <---\                           // (IX) = 0
*EBDF: 08             '.'                     INX                 |                           // IX++
*EBE0: 8C 00 56       '..V'                   CPX     #M0056      |                           // cmp IX, 0x0056
*EBE3: 26 F8          '&.'                    BNE     ZEBDD >---/                             // if (z) == 0; Schleife 0x004e ... 0x0056; Lösche Speicherbereich
*EBE5: 39             '9'                     RTS                                             // RETURN

// Call Initalisierung LCD mit Sonderzeichen
*EBE6: 0F             '.'             ZEBE6   SEI                                             // Set Interrupt Mask / NMI
*EBE7: CE F9 CA       '...'                   LDX     #MF9CA                                  // IX = 0xf9ca; Initialisierung LCD Port HD44100
*EBEA: A6 00          '..'            ZEBEA   LDAA    ,X      <---\                           // A = (0xf9ca)
*EBEC: B7 80 00       '...'                   STAA    M8000       |                           // (0x8000) = A LCD PORT-COMMAND
*EBEF: BD EA D6       '...'                   JSR     ZEAD6       |                           // Call 0xead6 Zeitschleife 10 ms
*EBF2: 08             '.'                     INX                 |                           // IX++
*EBF3: 8C F9 D6       '...'                   CPX     #MF9D6      |                           // cmp IX, 0xf9d6
*EBF6: 25 F2          '%.'                    BCS     ZEBEA >---/                             // if (c) == 1 Schleife LCD Port Initialsierung
```

```
*EBF8: 86 40          '.@'                    LDAA    #$40                                    // A = 0x40; Command LCD Sonderzeichen-Adresse
*EBFA: BD EC 3B       '..;'                   JSR     ZEC3B                                   // Call 0xec3b; LCD COMMAND; Übergabe A
*EBFD: CE F9 D6       '...'                   LDX     #MF9D6                                  // IX = 0xf9d6; Data Sonderzeichen für LCD
*EC00: 7F 00 5A       '..Z'                   CLR     >M005A                                  // (0x005a) = 0x00
*EC03: 3C             '<'             ZEC03   PSHX            <---------\                     // Push IX
*EC04: CE 00 0A       '...'                   LDX     #M000A          |                       // IX = 0x000a
*EC07: 09             '.'             ZEC07   DEX                <---\|                       // IX--
*EC08: 26 FD          '&.'                    BNE     ZEC07 >---/     ||                      // if (z) == 0 Zeitschleife 10 * 1,6 ms
*EC0A: CE 00 40       '..@'                   LDX     #M0040          ||                      // IX = 0x0040
*EC0D: B6 80 00       '...'           ZEC0D   LDAA    M8000           |<---\                    // A = (0x8000) LCD-Port-COMMAND BUSY
*EC10: F6 80 00       '...'                   LDAB    M8000           |    |                    // B = (0x8000) LCD-Port-COMMAND BUSY
*EC13: 4D             'M'                     TSTA                    |    |                    // A == 0; bit7 == 0
*EC14: 2A 06          '*.'                    BPL     ZEC1C  >----\   |    |                    // if (n) == 0; Warte auf 1 -> 0 wechsel
*EC16: 09             '.'                     DEX                 |   |    |                    // IX--
*EC17: 26 F4          '&.'                    BNE     ZEC0D       |   |    |>---/               // if (z) == 0 Schleife 64 Runden Abfrage LCD-Port
*EC19: 7E E3 3B       '~.;'           ZEC19   JMP     ZE33B       |   |    |<---\               // JP 0xe33b
                                                                  |   |    |    |
// JP                                                             |   |    |    |
*EC1C: 54             'T'             ZEC1C   LSRB            <---/   |    |    |               // 0 -> B7 ... B0 -> c
*EC1D: 54             'T'                     LSRB                    |    |    |               // 0 -> B7 ... B0 -> c
*EC1E: 54             'T'                     LSRB                    |    |    |               // 0 -> B7 ... B0 -> c
*EC1F: 54             'T'                     LSRB                    |    |    |               // 0 -> B7 ... B0 -> c; B hight -> B low
*EC20: 84 30          '.0'                    ANDA    #$30            |    |    |               // A &= 0x30; Maskiere Bit 4, 5
*EC22: 1B             '.'                     ABA                     |    |    |               // A += B
*EC23: 91 5A          '.Z'                    CMPA    M005A           |    |    |               // CMP A, (0x005a)
*EC25: 26 F2          '&.'                    BNE     ZEC19           |    |>---/               // if (z) == 0 -> JP 0xe33b
*EC27: 38             '8'                     PULX                    |    |                    // Pop IX; 0xf9d6
*EC28: A6 00          '..'                    LDAA    ,X              |    |                    // A = (IX)
*EC2A: BD EC 57       '..W'                   JSR     ZEC57           |    |                    // Call 0xec57; LCD Data; Übergabe A
*EC2D: 7C 00 5A       '|.Z'                   INC     >M005A          |    |                    // (0x005a)++
*EC30: 08             '.'                     INX                     |    |                    // IX++
*EC31: 8C FA 16       '...'                   CPX     #ZFA16          |    |                    // cmp IX, 0xfa16; IX == 0xf9d6 bis 0xfa15
*EC34: 25 CD          '%.'                    BCS     ZEC03 >---------/    |                    // if (c) == 1 -> 0xec03, Schleife Lade Sonderzeichen in LCD
*EC36: 39             '9'                     RTS                          |                    // RETURN
                                                                           |
// CALL Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND       |
*EC37: 36             '6'             ZEC37   PSHA                         |                    // Push A
*EC38: 8D 2C          '.,'                    BSR     ZEC66                |                    // Call 0xec66; Warte auf Busy LCD-Controller
*EC3A: 32             '2'                     PULA                         |                    // Pop A
                                                                           |
// Call LCD COMMAND; Übergabe A                                            |
*EC3B: 36             '6'             ZEC3B   PSHA                         |                    // Push A
*EC3C: 84 F0          '..'                    ANDA    #$F0                 |                    // A &= 0xf0 Maskiere; Kommandos: 0x8d, 0x8f, 0x95
*EC3E: B7 80 00       '...'                   STAA    M8000                |                    // (0x8000) = A hight an LCD-Port-COMMAND Senden 4 bit
*EC41: 32             '2'                     PULA                         |                    // Pop A
*EC42: 36             '6'                     PSHA                         |                    // Push A
*EC43: 48             'H'                     ASLA                         |                    // c <- A7 ... A0 <- 0 Shift left
*EC44: 48             'H'                     ASLA                         |                    // c <- A7 ... A0 <- 0 Shift left
*EC45: 48             'H'                     ASLA                         |                    // c <- A7 ... A0 <- 0 Shift left
*EC46: 48             'H'                     ASLA                         |                    // c <- A7 ... A0 <- 0 Shift left
*EC47: B7 80 00       '...'                   STAA    M8000                |                    // (0x8000) = A low an LCD-Port-COMMAND Senden 4 bit
*EC4A: 32             '2'                     PULA                         |                    // Pop A
*EC4B: 81 01          '..'                    CMPA    #$01                 |                    // CMP A, 0x01
*EC4D: 26 16          '&.'                    BNE     ZEC65  >---\         |                    // if (z) == 0 RETURN
*EC4F: 7E EA D6       '~..'                   JMP     ZEAD6      |         |                    // JP 0xead6 Zeitschleife 10 ms; -> RETURN
                                                                 |         |
// Call JP LCD Data nach Busy; Übergabe A                        |         |
*EC52: 36             '6'             ZEC52   PSHA               |         |                    // Push A
*EC53: BD EC 66       '..f'                   JSR     ZEC66      |         |                    // Call 0xec66; Warte auf Busy LCD-Controller
*EC56: 32             '2'                     PULA               |         |                    // Pop A
                                                                 |         |
// Call LCD Data; Übergabe A                                     |         |
*EC57: 36             '6'             ZEC57   PSHA               |         |                    // Push A
*EC58: 84 F0          '..'                    ANDA    #$F0       |         |                    // A &= 0xf0 Maskiere
*EC5A: B7 80 01       '...'                   STAA    M8001      |         |                    // (0x8001) = A high an LCD-Port-Data Senden 4 bit
*EC5D: 32             '2'                     PULA               |         |                    // Pop A
*EC5E: 48             'H'                     ASLA               |         |                    // c <- A7 ... A0 <- 0 Shift left
*EC5F: 48             'H'                     ASLA               |         |                    // c <- A7 ... A0 <- 0 Shift left
*EC60: 48             'H'                     ASLA               |         |                    // c <- A7 ... A0 <- 0 Shift left
*EC61: 48             'H'                     ASLA               |         |                    // c <- A7 ... A0 <- 0 Shift left
*EC62: B7 80 01       '...'                   STAA    M8001      |         |                    // (0x8001) = A low an LCD-Port-Data Senden 4 bit
*EC65: 39             '9'             ZEC65   RTS           <---/         |                    // Return
                                                                           |
// CALL Warte auf Busy LCD-Controller                                      |
*EC66: 36             '6'             ZEC66   PSHA                         |                    // Push A
*EC67: 37             '7'                     PSHB                         |                    // Push B
*EC68: B6 80 00       '...'           ZEC68   LDAA    M8000 <----\         |                    // A = (0x8000) LCD-Port-COMMAND
*EC6B: F6 80 00       '...'                   LDAB    M8000      |         |                    // B = (0x8000) LCD-Port-COMMAND
*EC6E: 4D             'M'                     TSTA               |         |                    // A == 0; bit7 == 1
*EC6F: 2B F7          '+.'                    BNI     ZEC68 >----/         |                    // if (n) == 1 Schleife bis Busy Flag auf "0" geht
*EC71: 33             '3'                     PULB                         |                    // Pop B
*EC72: 32             '2'                     PULA                         |                    // Pop A
*EC73: 39             '9'                     RTS                          |                    // Return
                                                                           |
// CALL  Rückgabe  A = (0x0030) oder (0x0031)                              |
*EC74: 96 3C          '.<'            ZEC74   LDAA    M003C                |                    // A = (0x003c)
*EC76: 80 28          '.('                    SUBA    #$28                 |                    // A-= 0x28
*EC78: 22 05          '".'                    BHI     ZEC7F >---\          |                    // if (c) & (z) == 0
*EC7A: 7F 00 31       '..1'                   CLR     >M0031    |          |                    // (0x0031) = 0
*EC7D: 20 22          ' "'                    BRA     ZECA1     |          |                    // JR 0xeca1 -> RETURN
                                                                |          |
*EC7F: 4C             'L'             ZEC7F   INCA          <---/          |                    // A++
*EC80: 91 31          '.1'                    CMPA    M0031                |                    // CMP A, (0x0031)
*EC82: 24 02          '$.'                    BCC     ZEC86 >---\          |                    // if (c) == 0
*EC84: 97 31          '.1'                    STAA    M0031     |          |                    // (0x0031) = A
```

```
*EC86: 96 31          '.1'              ZEC86   LDAA    M0031 <---/                       // A = (0x0031)
*EC88: 2A 03          '*.'                      BPL     ZEC8D >---\                       // if (n) == 0; bit7 == 0
*EC8A: 7F 00 31       '..1'                     CLR     >M0031    |                       // (0x0031) = 0
*EC8D: 96 30          '.0'              ZEC8D   LDAA    M0030 <---/                       // A = (0x0030)
*EC8F: 80 27          '.'                       SUBA    #$27                              // A-= 0x27
*EC91: 23 06          '#.'                      BLS     ZEC99 >---\                       // if (c) | (z) == 1
*EC93: 91 31          '.1'                      CMPA    M0031     |                       // CMP A, (0x0031)
*EC95: 23 02          '#.'                      BLS     ZEC99 >---+                       // if (c) | (z) == 1
*EC97: 97 31          '.1'                      STAA    M0031     |                       // (0x0031) = A
*EC99: 96 30          '.0'              ZEC99   LDAA    M0030 <---/                       // A = (0x0030)
*EC9B: 91 31          '.1'                      CMPA    M0031                             // CMP A, (0x0031)
*EC9D: 24 02          '$.'                      BCC     ZECA1  >---\                      // if (c) == 0 RETURN A = (0x0030)
*EC9F: 97 31          '.1'                      STAA    M0031      |                      // (0x0031) = A
*ECA1: 39             '9'               ZECA1   RTS            <---/                     // RETURN A = (0x0031)

// Call  Rückgabe A für Berechnung Sprungverteiler
*ECA2: BD E2 0D       '...'             ZECA2   JSR     ZE20D                             // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*ECA5: 27 16          ''.'                      BEQ     ZECBD >---\                       // if (z) == 1
*ECA7: DC 27          '.''                      LDD     M0027     |                       // D = (0x0027)
*ECA9: 93 25          '.%'                      SUBD    M0025     |                       // D -= (0x0025); Zeiger Text-Ende
*ECAB: C3 00 02       '...'                     ADDD    #M0002    |                       // D += 0x0002
*ECAE: DD 6E          '.n'                      STD     M006E     |                       // (0x006e) = D
*ECB0: BD E2 12       '...'                     JSR     ZE212     |                       // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*ECB3: DC 6E          '.n'                      LDD     M006E     |                       // D = (0x006e)
*ECB5: 6D 00          'm.'                      TST     ,X        |                       // (IX) == 0; bit7 == 0
*ECB7: 2A 02          '*.'                      BPL     ZECBB     | >---\                 // if (n) == 0
*ECB9: 8A 80          '..'                      ORAA    #$80      |     |                 // A |= 0x80; Setze Bit 7
*ECBB: ED 00          '..'              ZECBB   STD     ,X        | <---/                 // (IX) = D
*ECBD: 8D B5          '..'              ZECBD   BSR     ZEC74 <---/                       // Call 0xec74; Rückgabe  A = (0x0030) bzw. (0x0031)
*ECBF: 7E ED EC       '~..'                     JMP     ZEDEC                             // JP 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe

// Call Ausgabe Fehler Text auf LCD; Übergabe IX
*ECC2: BD EA 94       '...'             ZECC2   JSR     ZEA94                             // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms

// CALL JP Ausgabe Text auf LCD; Übergabe IX
*ECC5: 8D 23          '.#'              ZECC5   BSR     ZECEA                             // Call 0xecea; Textausgabe LCD, Übergabe IX
*ECC7: 7E EA CD       '~..'                     JMP     ZEACD                             // JP 0xeacd; Zeitschleife * 0x64;->RETURN

// Call Textausgabe auf LCD, Warten auf Tastendruck; Übergabe IX
*ECCA: 8D 1E          '..'              ZECCA   BSR     ZECEA                             // Call 0xecea; Textausgabe LCD, Übergabe IX
*ECCC: 7E F3 F9       '~..'                     JMP     ZF3F9                             // JP 0xf3f9; Tastendruck; Rückgabe A

// Call Übergabe D, IX AusgabeText; Rückgabe D (0x0066)
*ECCF: DD 66          '.f'              ZECCF   STD     M0066                             // (0x0066) = D (0x..0c, 0x..10, 0x..14)
*ECD1: BD ED CD       '...'                     JSR     ZEDCD                             // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*ECD4: DC 66          '.f'                      LDD     M0066                             // D = (0x0066); Bsp. aus Chiff/Dechiff: A = 0x00 ... 0x63 B = 0x10, 0x0c, 0x14
*ECD6: CE 00 CD       '...'                     LDX     #M00CD                            // IX = 0x00cd
*ECD9: 3A             ':'                       ABX                                       // IX += B; 0x10, 0x0c, 0x14
*ECDA: BD EA EB       '...'                     JSR     ZEAEB                             // Call 0xeaeb; A Division 10; Übergabe A, Rückgabe A, B; A = "0" ... "9"; B Anzahl der Division
*ECDD: E7 00          '..'                      STAB    ,X                                // (IX) = B
*ECDF: A7 01          '..'                      STAA    $01,X                             // (IX + 0x01) = A; D = Länge Dezimal
*ECE1: 86 29          '.)'                      LDAA    #$29                              // A = 0x29; 40 Elemente
*ECE3: 97 32          '.2'                      STAA    M0032                             // (0x0032) = A
*ECE5: 8D 12          '..'                      BSR     ZECF9 >---\                       // Call 0xecf9; Übergabe (0x0032)
*ECE7: DC 66          '.f'                      LDD     M0066     |                       // D = (0x0066)
*ECE9: 39             '9'                       RTS               |                       // RETURN
                                                                  |
// Call Übergabe IX                                               |
*ECEA: BD ED CD       '...'             ZECEA   JSR     ZEDCD     |                       // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*ECED: 86 29          '.)'                      LDAA    #$29      |                       // A = 0x29; 40 Elemente
*ECEF: 97 32          '.2'                      STAA    M0032     |                       // (0x0032) = A
*ECF1: 7F 00 57       '..W'                     CLR     >M0057    |                       // (0x0057) = 0
*ECF4: 20 03          ' .'                      BRA     ZECF9 >---+                       // JR; Übergabe (0x0032)
                                                                  |
*ECF6: BD EE AE       '...'             ZECF6   JSR     ZEEAE     |                       // Call 0xeeae; Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
                                                                  |
// CALL JP Übergabe ( 0x0032, 0x00cd ) Init LCD und Textausgabe   |
*ECF9: 86 01          '..'              ZECF9   LDAA    #$01  <---/                       // A = 0x01; Command LCD Clear Display
*ECFB: BD EC 37       '..7'                     JSR     ZEC37                             // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*ECFE: 86 80          '..'                      LDAA    #$80                              // A = 0x80; DD-RAM LCD
*ED00: BD EC 37       '..7'                     JSR     ZEC37                             // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*ED03: CE 00 CD       '...'                     LDX     #M00CD                            // IX = 0x00cd
*ED06: C6 28          '.('                      LDAB    #$28                              // B = 0x28; 40 Runden -> 40 Zeichen pro Zeile
*ED08: A6 00          '..'              ZED08   LDAA    ,X    <---\                       // A = (IX)
*ED0A: BD EC 52       '..R'                     JSR     ZEC52     |                       // Call 0xec52; LCD Data nach Busy; Übergabe A
*ED0D: 08             '.'                       INX               |                       // IX++
*ED0E: 5A             'Z'                       DECB              |                       // B--
*ED0F: 26 F7          '&.'                      BNE     ZED08 >---/                       // if (z) == 0 Schleife 40 Runden -> 40 Zeichen pro Zeile
*ED11: 96 3A          '.:'                      LDAA    M003A                             // A = (0x003a)
*ED13: 81 C0          '..'                      CMPA    #$C0                              // CMP A, 0xc0; LCD Kopfzeile Telefonhörer
*ED15: 25 0E          '%.'                      BCS     ZED25 >---\                       // if (c) == 1
*ED17: 81 C5          '..'                      CMPA    #$C5      |                       // CMP A, 0xc5; LCD Kopfzeile FORMAT:
*ED19: 22 0A          '".'                      BHI     ZED25 >---+                       // if (c) & (z) == 0
*ED1B: 36             '6'               ZED1B   PSHA              | <---\                 // Push A
*ED1C: BD EE C5       '...'                     JSR     ZEEC5     |     |                 // Call 0xeec5; LCD Data mit Busy;
*ED1F: 32             '2'                       PULA              |     |                 // Pop A
*ED20: 4A             'J'                       DECA              |     |                 // A--
*ED21: 81 C1          '..'                      CMPA    #$C1      |     |                 // CMP A, 0xc1; Level 1
*ED23: 24 F6          '$.'                      BCC     ZED1B     | >---/                 // if (c) == 0 Schleife 5 Runden - 0xc1 ... 0xc5
*ED25: 86 C7          '..'              ZED25   LDAA    #$C7  <---/                       // A = 0xc7
*ED27: 8D 56          '.V'                      BSR     ZED7F                       >---\ // Call 0xed7f ---> 0xeec5; Übergabe A
*ED29: 96 3C          '.<'                      LDAA    M003C                           |     // A = (0x003c)
*ED2B: 81 29          '.)'                      CMPA    #$29                            |     // CMP A, 0x29; 40 Elemente
*ED2D: 25 06          '%.'                      BCS     ZED35 >---\                     |     // if (c) == 1
*ED2F: 86 CA          '..'                      LDAA    #$CA      |                     |     // A = 0xca
```

```
*ED31: 8D 4C          '.L'                    BSR     ZED7F     |                    >---+      // Call 0xed7f ---> 0xeec5; Übergabe A
*ED33: 20 04          ' .'                    BRA     ZED39     |  >---\                  |      // JR 0xed39
                                                                |      |                  |
*ED35: 86 CB          '..'            ZED35   LDAA    #$CB  <---/      |                  |      // A = 0xcb; LCD Kopfzeile "L"
*ED37: 8D 46          '.F'                    BSR     ZED7F            |                  >---+      // Call 0xed7f ---> 0xeec5; Übergabe A
                                                                       |                      |
// JP                                                                  |                      |
*ED39: 7D 00 36       '}.6'           ZED39   TST     >M0036       <---/                      |      // (0x0036) == 0?;  MSB == 1 n-Flag (0x0036) == 0 z-Flag
*ED3C: 27 04          ''.'                    BEQ     ZED42  >---\                            |      // if (z) == 1
*ED3E: 86 D2          '..'                    LDAA    #$D2       |                            |      // A = 0xd2; LCD Kopfzeile
*ED40: 8D 3D          '.='                    BSR     ZED7F      |                            >---+      // Call 0xed7f ---> 0xeec5; Übergabe A
*ED42: 7D 00 37       '}.7'           ZED42   TST     >M0037 <---/                            |      // (0x0037) == 0?;  MSB == 1 n-Flag (0x0037) == 0 z-Flag
*ED45: 27 04          ''.'                    BEQ     ZED4B  >---\                            |      // if (z) == 1
*ED47: 86 D9          '..'                    LDAA    #$D9       |                            |      // A = 0xd9; LCD Kopfzeile
*ED49: 8D 34          '.4'                    BSR     ZED7F      |                            >---+      // Call 0xed7f ---> 0xeec5; Übergabe A
*ED4B: DE 23          '.#'            ZED4B   LDX     M0023  <---/                            |      // IX = (0x0023)
*ED4D: 86 DD          '..'                    LDAA    #$DD                                    |      // A = 0xdd; LCD M:_XXXX_ % Anzeige; 4*0x075e
*ED4F: 8C 1F FF       '...'                   CPX     #M1FFF                                  |      // cmp IX, 0x1fff; > 99%
*ED52: 24 13          '$.'                    BCC     ZED67 >---\                             |      // if (c) == 0; Übergabe A = 0xdd
*ED54: 4C             'L'                     INCA              |                             |      // A++; LCD M: XXXX  % Anzeige; 3*0x075e
*ED55: 8C 18 A1       '...'                   CPX     #M18A1    |                             |      // cmp IX, 0x18a1; > 75%
*ED58: 24 0D          '$.'                    BCC     ZED67 >---+                             |      // if (c) == 0; Übergabe A = 0xde
*ED5A: 4C             'L'                     INCA              |                             |      // A++; LCD M: XXXX  % Anzeige; 2*0x075e
*ED5B: 8C 11 43       '..C'                   CPX     #M1143    |                             |      // cmp IX, 0x1143; > 50%
*ED5E: 24 07          '$.'                    BCC     ZED67 >---+                             |      // if (c) == 0; Übergabe A = 0xdf
*ED60: 4C             'L'                     INCA              |                             |      // A++; LCD M:_XXXX_ % Anzeige; 1*0x075e
*ED61: 8C 09 E5       '...'                   CPX     #M09E5    |                             |      // cmp IX, 0x09e5; > 25%
*ED64: 24 01          '$.'                    BCC     ZED67 >---+                             |      // if (c) == 0; Übergabe A = 0xe0
*ED66: 4C             'L'                     INCA              |                             |      // A++; ; LCD M:_XXXX_ % Anzeige; > 0x075e
*ED67: 36             '6'             ZED67   PSHA          <---+                             |      // Push A
*ED68: BD EE C5       '...'                   JSR     ZEEC5     |                             |      // Call 0xeec5; LCD Data mit Busy;
*ED6B: 32             '2'                     PULA              |                             |      // Pop A
*ED6C: 4A             'J'                     DECA              |                             |      // A--
*ED6D: 81 DD          '..'                    CMPA    #$DD      |                             |      // CMP A, 0xdd
*ED6F: 24 F6          '$.'                    BCC     ZED67 >---/                             |      // if (c) == 0; Übergabe A = 0xde ...
*ED71: 96 3C          '.<'                    LDAA    M003C                                   |      // A = (0x003c)
*ED73: 81 21          '.!'                    CMPA    #$21                                    |      // CMP A, 0x21
*ED75: 23 06          '#.'                    BLS     ZED7D >---\                             |      // if (c) | (z) == 1
*ED77: 96 32          '.2'                    LDAA    M0032     |                             |      // A = (0x0032)
*ED79: 81 1A          '..'                    CMPA    #$1A      |                             |      // CMP A, 0x1a
*ED7B: 2C 14          ',.'                    BGE     ZED91     |  >---\                      |      // if (n) == (v) == 1/0; RETURN
*ED7D: 86 E5          '..'            ZED7D   LDAA    #$E5  <---/      |                      |      // A = 0xe5
// Abrruch                                                             |                      |
*ED7F: 7E EE C5       '~..'           ZED7F   JMP     ZEEC5            |                      <---/      // JP 0xeec5; LCD Data mit Busy; -> Return
                                                                       |
// Call Fülle 0x00cd ... 0x00f5 mit 0x20 " "                           |
*ED82: 3C             '<'             ZED82   PSHX                     |                             // Push IX
*ED83: CE 00 CD       '...'                   LDX     #M00CD           |                             // IX = 0x00cd
*ED86: 86 20          '. '                    LDAA    #$20             |                             // A = 0x20 " "
*ED88: C6 28          '.('                    LDAB    #$28             |                             // B = 0x28
*ED8A: A7 00          '..'            ZED8A   STAA    ,X    <---\      |                             // (IX) = A
*ED8C: 08             '.'                     INX               |      |                             // IX++
*ED8D: 5A             'Z'                     DECB              |      |                             // B--
*ED8E: 26 FA          '&.'                    BNE     ZED8A >---/      |                             // if (z) == 0 Schleife 40 Runden; Fülle 0x00cd ... 0x00f4 mit 0x20 " "
*ED90: 38             '8'                     PULX                     |                             // Pop IX
*ED91: 39             '9'             ZED91   RTS                   >---/                             // RETURN

// Call
*ED92: 3C             '<'             ZED92   PSHX                                                  // Push IX
*ED93: CE 01 00       '...'                   LDX     #M0100                                        // IX = 0x0100
*ED96: D6 31          '.1'                    LDAB    M0031                                         // B = (0x0031)
*ED98: 3A             ':'                     ABX                                                   // IX += B; 0x0100 + B
*ED99: DF 66          '.f'                    STX     M0066                                         // (0x0066) = IX
*ED9B: C6 00          '..'                    LDAB    #$00                                          // B = 0x00
*ED9D: DE 66          '.f'            ZED9D   LDX     M0066 <---\                                   // IX = (0x0066) 0x0100+B
*ED9F: 3A             ':'                     ABX               |                                   // IX += B; 0x0100+B + Schleifenzähler B
*EDA0: A6 00          '..'                    LDAA    ,X        |                                   // A = (IX)
*EDA2: 84 02          '..'                    ANDA    #$02      |                                   // A &= 0x02; Bit 1
*EDA4: 8B 04          '..'                    ADDA    #$04      |                                   // A += 0x04; setze Bit 2; A 0x4 oder 0x6
*EDA6: CE 00 CD       '...'                   LDX     #M00CD    |                                   // IX = 0x00cd
*EDA9: 3A             ':'                     ABX               |                                   // IX += B; IX + Schleifenzähler B
*EDAA: A7 00          '..'                    STAA    ,X        |                                   // (IX) = A; 0x4 oder 0x6
*EDAC: 5C             '\'                     INCB              |                                   // B++
*EDAD: C1 28          '.('                    CMPB    #$28      |                                   // cmp B, 0x28
*EDAF: 25 EC          '%.'                    BCS     ZED9D >---/                                   // if (c) == 1 Schleife 40 Runden

*EDB1: D6 3C          '.<'                    LDAB    M003C                                         // B = (0x003c)
*EDB3: C1 28          '.('                    CMPB    #$28                                          // cmp B, 0x28
*EDB5: 24 14          '$.'                    BCC     ZEDCB >---\                                   // if (c) == 0 -> Pop IX RETURN
*EDB7: 86 07          '..'                    LDAA    #$07      |                                   // A = 0x07
*EDB9: CE 00 CD       '...'                   LDX     #M00CD    |                                   // IX = 0x00cd
*EDBC: 3A             ':'                     ABX               |                                   // IX += B
*EDBD: A7 00          '..'                    STAA    ,X        |                                   // (IX) = A
*EDBF: 86 20          '. '                    LDAA    #$20      |                                   // A = 0x20
*EDC1: 08             '.'             ZEDC1   INX               | <---\                             // IX++; IX = 0x00cd + B (0x00 .. 0x27)
*EDC2: 8C 00 F5       '...'                   CPX     #M00F5    |     |                             // cmp IX, 0x00f5
*EDC5: 24 04          '$.'                    BCC     ZEDCB >---+     |                             // if (c) == 0 maximal 48 Zeichen
*EDC7: A7 00          '..'                    STAA    ,X        |     |                             // (IX) = A
*EDC9: 20 F6          ' .'                    BRA     ZEDC1     | >---/                             // JR Schleife mit 0x20 " " füllen
                                                                |
*EDCB: 38             '8'             ZEDCB   PULX          <---/                                   // Pop IX
*EDCC: 39             '9'                     RTS                                                   // RETURN

// Call Übergabe IX (Textadresse, Textlänge)
*EDCD: BD ED 82       '...'           ZEDCD   JSR     ZED82                                         // Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
```

```
*EDD0: 9F 6C          '.l'                    STS     M006C                                   // (0x006c) = SP
*EDD2: 35             '5'                     TXS                                             // SP = (IX)
*EDD3: CE 00 CD       '...'                   LDX     #M00CD                                  // IX = 0x00cd
*EDD6: 32             '2'             ZEDD6   PULA                    <---\                   // Pop A
*EDD7: 16             '.'                     TAB                         |                   // B = A
*EDD8: 84 7F          '..'                    ANDA    #$7F                |                   // A &= 0x7f Maskiere
*EDDA: A7 00          '..'                    STAA    ,X                  |                   // (IX) = A
*EDDC: 08             '.'                     INX                         |                   // IX++
*EDDD: 5D             ']'                     TSTB                        |                   // B == 0; bit7 == 0
*EDDE: 2A F6          '*.'                    BPL     ZEDD6           >---+                   // if (n) == 0; Schleife bis Textende
*EDE0: C1 A0          '..'                    CMPB    #$A0                |                   // cmp B, 0xa0
*EDE2: 26 05          '&.'                    BNE     ZEDE9 >---\         |                   // if (z) == 0
*EDE4: 8E F9 BC       '...'                   LDS     #MF9BC    |         |                   // SP = 0xf9bc; Nutze SP als Register
*EDE7: 20 ED          ' .'                    BRA     ZEDD6     |   >---/                   // JR Wiederhole bis Textende bit 7 == 1
*EDE9: 9E 6C          '.l'            ZEDE9   LDS     M006C <---/                             // SP = (0x006c);  SP wiederherstellen
*EDEB: 39             '9'                     RTS                                             // RETURN

// Call JP Ausgabe Freier Speicherplatz
*EDEC: 8D A4          '..'            ZEDEC   BSR     ZED92                                   // Call 0xed92
*EDEE: BD E2 0D       '...'                   JSR     ZE20D                                   // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*EDF1: 26 03          '&.'                    BNE     ZEDF6 >---\                             // if (z) == 0
*EDF3: 7E EE 86       '~..'                   JMP     ZEE86     |                             // JP 0xee86 Ausgabe auf LCD: < FREE >
// Call Textnr. Chiffriert                                      |
*EDF6: BD EA DF       '...'           ZEDF6   JSR     ZEADF <---/                             // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*EDF9: 26 03          '&.'                    BNE     ZEDFE >---\                             // if (z) == 0
*EDFB: 7E EE 92       '~..'                   JMP     ZEE92     |                             // JP 0xee92; Textausgabe "ENCRYPTED TEXT"
                                                              |                               |
// Weiter-Abbruch                                             |                               |
*EDFE: DE 29          '.)'            ZEDFE   LDX     M0029 <---/                             // IX = (0x0029); Zeiger Text-aktual Position
*EE00: D6 31          '.1'                    LDAB    M0031                                   // B = (0x0031)
*EE02: 3A             ':'                     ABX                                             // IX += B
*EE03: C6 00          '..'                    LDAB    #$00                                    // B = 0x00
*EE05: 9C 2B          '.+'            ZEE05   CPX     M002B       <---\                       // cmp IX, (0x002b); Zeiger Text-aktual Position
*EE07: 24 14          '$.'                    BCC     ZEE1D >---\     |                       // if (c) == 0
*EE09: A6 00          '..'                    LDAA    ,X        |     |                       // A = (IX)
*EE0B: 3C             '<'                     PSHX              |     |                       // Push IX
*EE0C: CE 00 CD       '...'                   LDX     #M00CD    |     |                       // IX = 0x00cd
*EE0F: 3A             ':'                     ABX               |     |                       // IX += B
*EE10: 84 7F          '..'                    ANDA    #$7F      |     |                       // A &= 0x7f Maskiere
*EE12: A7 00          '..'                    STAA    ,X        |     |                       // (IX) = A
*EE14: 38             '8'                     PULX              |     |                       // Pop IX
*EE15: 08             '.'                     INX               |     |                       // IX++
*EE16: 5C             '\'                     INCB              |     |                       // B++
*EE17: C1 28          '.('                    CMPB    #$28      |     |                       // cmp B, 0x28
*EE19: 26 EA          '&.'                    BNE     ZEE05     | >---/                       // if (z) == 0 Schleife 40 Runden
*EE1B: 20 14          ' .'                    BRA     ZEE31     | >--\                        // JR 0xee31
// Weiter                                                       |    |
*EE1D: 22 12          '".'            ZEE1D   BHI     ZEE31 <---/ >--+                        // if (c) & (z) == 0
*EE1F: D1 3C          '.<'                    CMPB    M003C          |                        // cmp B, (0x003c)
*EE21: 27 0E          ''.'                    BEQ     ZEE31      >---+                        // if (z) == 1
*EE23: A6 00          '..'                    LDAA    ,X             |                        // A = (IX)
*EE25: 81 8D          '..'                    CMPA    #$8D           |                        // CMP A, 0x8d
*EE27: 27 08          ''.'                    BEQ     ZEE31      >---+                        // if (z) == 1
*EE29: CE 00 CD       '...'                   LDX     #M00CD         |                        // IX = 0x00cd
*EE2C: 3A             ':'                     ABX                    |                        // IX += B
*EE2D: 84 7F          '..'                    ANDA    #$7F           |                        // A &= 0x7f Maskiere
*EE2F: A7 00          '..'                    STAA    ,X             |                        // (IX) = A
// Einsprung                                                    |
*EE31: D6 30          '.0'            ZEE31   LDAB    M0030      <---/                        // B = (0x0030)
*EE33: D0 31          '.1'                    SUBB    M0031                                   // B -= (0x0031)
*EE35: D1 3C          '.<'                    CMPB    M003C                                   // cmp B, (0x003c)
*EE37: 26 05          '&.'                    BNE     ZEE3E >---\                             // if (z) == 0
*EE39: C1 28          '.('                    CMPB    #$28      |                             // cmp B, 0x28
*EE3B: 26 01          '&.'                    BNE     ZEE3E >---+                             // if (z) == 0
*EE3D: 5A             'Z'                     DECB              |                             // B--
*EE3E: D7 32          '.2'            ZEE3E   STAB    M0032 <---/                             // (0x0032) = B
*EE40: 96 3C          '.<'                    LDAA    M003C                                   // A = (0x003c)
*EE42: 81 21          '.!'                    CMPA    #$21                                    // CMP A, 0x21
*EE44: 23 04          '#.'                    BLS     ZEE4A           >---\                   // if (c) | (z) == 1
*EE46: C1 1A          '..'                    CMPB    #$1A                |                   // cmp B, 0x1a
*EE48: 24 39          '$9'                    BCC     ZEE83               |   >---\           // if (c) == 0 -> 0xecf9
// Einsprung aus Textausgabe "<Free>"                                       |       |
*EE4A: 86 01          '..'            ZEE4A   LDAA    #$01            <---+       |           // A = 0x01
*EE4C: DE 29          '.)'                    LDX     M0029               |       |           // IX = (0x0029)
*EE4E: 9C 25          '.%'                    CPX     M0025               |       |           // cmp IX, (0x0025); Zeiger Text-Ende
*EE50: 27 19          ''.'                    BEQ     ZEE6B           >---\ |       |           // if (z) == 1
*EE52: 09             '.'                     DEX                       | |       |           // IX--
*EE53: DF 6E          '.n'                    STX     M006E             | |       |           // (0x006e) = IX
*EE55: DE 25          '.%'                    LDX     M0025             | |       |           // IX = (0x0025); Zeiger Text-Ende
*EE57: 36             '6'                     PSHA                      | |       |           // Push A
*EE58: 09             '.'                     DEX                       | |       |           // IX--
*EE59: 08             '.'             ZEE59   INX                       | |       |           // IX++; Prüfung auf NULL
*EE5A: 6D 00          'm.'                    TST     ,X                | |       |           // (IX) == 0; bit7 == 0
*EE5C: 2A FB          '*.'                    BPL     ZEE59 <-------\   | |       |           // if (n) == 0; Flag aus INC/Dec IX
*EE5E: 32             '2'                     PULA                  |   | |       |           // Pop A
*EE5F: 4C             'L'                     INCA                  |   | |       |           // A++
*EE60: 81 64          '.d'                    CMPA    #$64          |   | |       |           // CMP A, 0x64
*EE62: 25 01          '%.'                    BCS     ZEE65 >---\   |   | |       |           // if (c) == 1; Zähler: 0 ... 0x64
*EE64: 4F             'O'                     CLRA              |   |   | |       |           // A = 0
*EE65: 36             '6'             ZEE65   PSHA          <---/   |   | |       |           // Push A
*EE66: 9C 6E          '.n'                    CPX     M006E         |   | |       |           // cmp IX, (0x006e)
*EE68: 26 EF          '&.'                    BNE     ZEE59 >-------/   | |       |           // if (z) == 0
*EE6A: 32             '2'                     PULA                      | |       |           // Pop A; A = 0x00
*EE6B: BD EA EB       '...'           ZEE6B   JSR     ZEAEB         <---/ |       |           // Call 0xeaeb; A Division 10; Übergabe A, Rückgabe A, B; A = "0" ... "9"; B Anzahl der Division
*EE6E: D7 F0          '..'                    STAB    M00F0               |       |           // (0x00f0) = B = 0x30 "0"
```

```
*EE70: 97 F1          '..'                    STAA    M00F1             |      |                // (0x00f1) = A = 0x30 "0"
*EE72: 86 20          '. '                    LDAA    #$20              |      |                // A = 0x20
*EE74: 97 EF          '..'                    STAA    M00EF             |      |                // (0x00ef) = A
*EE76: 86 2D          '.-'                    LDAA    #$2D              |      |                // A = 0x2d
*EE78: 97 F2          '..'                    STAA    M00F2             |      |                // (0x00f2) = A
*EE7A: 96 2F          './'            ZEE7A   LDAA    M002F <---\       |      |                // A = (0x002f); Zählervariable 0x1 ... 0x62
*EE7C: BD EA EB       '...'                   JSR     ZEAEB     |       |      |                // Call 0xeaeb; A Division 10; Übergabe A, Rückgabe A, B; A = "0" ... "9"; B Anzahl der Division
*EE7F: D7 F3          '..'                    STAB    M00F3     |       |      |                // (0x00f3) = B
*EE81: 97 F4          '..'                    STAA    M00F4     |       |      |                // (0x00f4) = A
*EE83: 7E EC F9       '~..'           ZEE83   JMP     ZECF9     |       |  <---/                // JP 0xecf9; Übergabe (0x0032)
                                                                |       |
/// JP                                                           |       |
*EE86: CE F7 6A       '..j'           ZEE86   LDX     #MF76A    |       |                       // IX = 0xf76a; "< FREE >
*EE89: BD ED CD       '...'                   JSR     ZEDCD     |       |                       // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*EE8C: 86 00          '..'                    LDAA    #$00      |       |                       // A = 0x00
*EE8E: 97 32          '.2'                    STAA    M0032     |       |                       // (0x0032) = A
*EE90: 20 B8          ' .'                    BRA     ZEE4A     |   >---/                       // JR 0xee4a
                                                                |
// JP                                                           |
*EE92: CE F7 A9       '...'           ZEE92   LDX     #MF7A9    |                               // IX = 0xf7a9; "ENCRYPTED TEXT, LENGTH /:99 BYTE"
*EE95: BD ED CD       '...'                   JSR     ZEDCD     |                               // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*EE98: BD E2 12       '...'                   JSR     ZE212     |                               // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*EE9B: EC 00          '..'                    LDD     ,X        |                               // D = (IX)
*EE9D: 84 7F          '..'                    ANDA    #$7F      |                               // A &= 0x7f Maskiere
*EE9F: 8D 0A          '..'                    BSR     ZEEAB     |                               // Call 0xeeab; Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*EEA1: 86 29          '.)'                    LDAA    #$29      |                               // A = 0x29; 40 Elemente
*EEA3: 97 32          '.2'                    STAA    M0032     |                               // (0x0032) = A
*EEA5: 7F 00 57       '..W'                   CLR     >M0057    |                               // (0x0057) = 0
*EEA8: 7E EE 7A       '~.z'                   JMP     ZEE7A >---/                               // JP 0xee7a

// Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*EEAB: CE 00 E4       '...'           ZEEAB   LDX     #M00E4                                    // IX = 0x00e4; Anzahl der Tausender
// Call Längenberechnung Übergabe D, IX; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*EEAE: 6C 00          'l.'            ZEEAE   INC     ,X    <----\                              // IX++; Anzahl 1000
*EEB0: 83 03 E8       '...'                   SUBD    #M03E8     |                              // D -= 1000
*EEB3: 24 F9          '$.'                    BCC     ZEEAE >----/                              // if (c) == 0; D größer 1000
*EEB5: 6A 01          'j.'            ZEEB5   DEC     $01,X <----\                              // (IX + 0x01)--; Anzahl 100
*EEB7: C3 00 64       '..d'                   ADDD    #M0064     |                              // D += 100
*EEBA: 24 F9          '$.'                    BCC     ZEEB5 >----/                              // if (c) == 0; D größer 0?
*EEBC: 17             '.'                     TBA                                               // A = B
*EEBD: BD EA EB       '...'                   JSR     ZEAEB                                     // Call 0xEAEB; A Division 10; Übergabe A, Rückgabe A, B; A = "0" ... "9"; B Anzahl der Division
*EEC0: E7 02          '..'                    STAB    $02,X                                     // (0x00e6) = B; Hunderter
*EEC2: A7 03          '..'                    STAA    $03,X                                     // (0x00e7) = A; Zehner
*EEC4: 39             '9'                     RTS                                               // RETURN

// Call JP LCD Data mit Busy; Übergabe A LCD-Command und Anzeige Cursor
*EEC5: BD EC 37       '..7'           ZEEC5   JSR     ZEC37                                     // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*EEC8: 86 05          '..'                    LDAA    #$05                                      // A = 0x05; Sonderzeichen Unterstrich/Cursor
*EECA: 7E EC 52       '~.R'                   JMP     ZEC52                                     // JP 0xec52; LCD Data nach Busy; Übergabe A; -> Return

// Call Sprungverteiler 31
*EECD: 86 4C          '.L'                    LDAA    #$4C                                      // A = 0x4c
*EECF: C6 FF          '..'                    LDAB    #$FF                                      // B = 0xff
*EED1: 20 09          ' .'                    BRA     ZEEDC      >---\                          // JR Sprungverteiler 34/2
                                                                      |
                                                                      |
// Call Sprungverteiler 2                                             |
*EED3: 86 48          '.H'                    LDAA    #$48           |                          // A = 0x48
*EED5: C6 01          '..'                    LDAB    #$01           |                          // B = 0x01
*EED7: 20 03          ' .'                    BRA     ZEEDC >--------+                          // JR Sprungverteiler 34/2 "Send"
// Sprungverteiler 34                                                 |
*EED9: 5F             '_'                     CLRB                   |                          // B = 0x00
*EEDA: 86 4E          '.N'                    LDAA    #$4E           |                          // A = 0x4e
                                                                     |
// Call JP Sprungverteiler 34/2 Übergabe A, B Rückgabe:              |
*EEDC: D7 39          '.9'            ZEEDC   STAB    M0039        <-/                          // (0x0039) = B (0x00, 0x01, 0xff)
*EEDE: 36             '6'                     PSHA                                              // Push A       (0x4e, 0x48, 0x4c)
*EEDF: BD E2 0D       '...'                   JSR     ZE20D                                     // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*EEE2: 26 03          '&.'                    BNE     ZEEE7 >----+                              // if (z) == 0; "Send ..."
*EEE4: 7E EA 94       '~..'                   JMP     ZEA94      |                              // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                 |
*EEE7: CE F9 85       '...'           ZEEE7   LDX     #MF985 <---/                              // IX = 0xf985; "READY TO SEND [.] - PRESS SENDKEY (/:999"
*EEEA: 71 7F 02       'q..'                   AIM     #$7F,M0002                                // (Port-1 Data)&= 0x7f; Maskiere
*EEED: 86 29          '.)'                    LDAA    #$29                                      // A = 0x29; 40 Elemente
*EEEF: 97 32          '.2'                    STAA    M0032                                     // (0x0032) = A
*EEF1: BD ED CD       '...'                   JSR     ZEDCD                                     // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*EEF4: 32             '2'                     PULA                                              // Pop A -> Übergabe (0x4e, 0x48, 0x4c)
*EEF5: 97 DC          '..'                    STAA    M00DC                                     // (0x00dc) = A
*EEF7: DC 27          '.''                    LDD     M0027                                     // D = (0x0027)
*EEF9: 93 25          '.%'                    SUBD    M0025                                     // D -= (0x0025); Zeiger Text-Ende
*EEFB: C3 00 02       '...'                   ADDD    #M0002                                    // D += 0x0002; Kompromat 0x0d und 0x28
*EEFE: CE 00 F0       '...'                   LDX     #M00F0                                    // IX = 0x00f0
*EF01: BD EC F6       '...'                   JSR     ZECF6                                     // Call 0xecf6 -> Call 0xeeae; Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*EF04: BD F3 F9       '...'           ZEF04   JSR     ZF3F9 <---\                               // Call 0xf3f9; Tastendruck; Rückgabe A
*EF07: 81 B1          '..'                    CMPA    #$B1      |                               // CMP A, 0xb1; Taste SEND
*EF09: 26 F9          '&.'                    BNE     ZEF04 >---/                               // if (z) == 0 Warte auf 0xb1-Taste SEND
*EF0B: 7F 00 08       '...'                   CLR     >M0008                                    // (0x0008) = 0; Port21 Output, TCSR1 Timer Control Status 1
*EF0E: CE F9 AD       '...'                   LDX     #MF9AD                                    // IX = 0xf9ad; "< TRANSMITTING >"
*EF11: BD EC EA       '...'                   JSR     ZECEA                                     // Call 0xecea; Textausgabe LCD, Übergabe IX
*EF14: 7D 00 39       '}.9'                   TST     >M0039                                    // (0x0039) == 0; bit7 == 0
*EF17: 2A 05          '*.'                    BPL     ZEF1E  >---\                              // if (n) == 0
*EF19: 86 CC          '..'                    LDAA    #$CC       |                              // A = 0xcc
*EF1B: BD EE C5       '...'                   JSR     ZEEC5      |                              // Call 0xeec5; LCD Data mit Busy;
*EF1E: BD F3 BE       '...'           ZEF1E   JSR     ZF3BE  <---/                              // Call 0xf3be; Lese Port-1 Data; Rückgabe B mit Bit 4 = 1
*EF21: 8D 14          '..'                    BSR     ZEF37                                     // Call 0xef37; Send-Akustik-Koppler
```

```
*EF23: BD F3 D5       '...'                   JSR     ZF3D5                   // Call 0xf3d5; Init Port-2-Data
*EF26: 72 80 02       'r..'                   OIM     #$80,M0002              // (Port-1 Data) |= 0x80 Port17 LED ON
*EF29: DE 25          '.%'                    LDX     M0025                   // IX = (0x0025); Zeiger Text-Ende
*EF2B: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX; Zeiger Text-Start
*EF2D: BD E5 BC       '...'                   JSR     ZE5BC                   // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*EF30: 7F 00 30       '..0'                   CLR     >M0030                  // (0x0030) = 0
*EF33: 7F 00 39       '..9'                   CLR     >M0039                  // (0x0039) = 0
*EF36: 39             '9'                     RTS                             // RETURN

// Call Send-Akustik-Koppler
*EF37: 7F 00 08       '...'           ZEF37   CLR     >M0008                  // (0x0008) = 0; Port21 Output, TCSR1 Timer Control Status 1
*EF3A: 0E             '.'                     CLI                             // Clear Interrupt Mask = 0
*EF3B: DE 25          '.%'                    LDX     M0025                   // IX = (0x0025); Zeiger Text-Ende
*EF3D: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX; Zeiger Text-Start
*EF3F: C6 32          '.2'                    LDAB    #$32                    // B = 0x32
*EF41: BD EA CF       '...'                   JSR     ZEACF                   // Call 0xeacf Zeitschleife 10ms, 500ms
*EF44: D6 39          '.9'                    LDAB    M0039                   // B = (0x0039); (0x0039) = (0x00, 0x01, 0xff)
*EF46: 5C             '\'                     INCB                            // B++           (0x0039) = (0x01, 0x02, 0x00)
*EF47: 58             'X'                     ASLB                            // c <- B7 ... B0 <- 0 Shift left; B = 0x02, 0x04, 0x00 ... 0xfe Schrittweite 2
*EF48: CE F3 E1       '...'                   LDX     #MF3E1                  // IX = 0xf3e1 -> Baudrate
*EF4B: 3A             ':'                     ABX                             // IX += B; 600, 1200, 300 Bd
*EF4C: EE 00          '..'                    LDX     ,X                      // IX = (IX); 0x0683; 0x0341; 0x0d05; 600-1200-300 Bd
*EF4E: DF 5A          '.Z'                    STX     M005A                   // (0x005a) = IX
*EF50: DF 0B          '..'                    STX     M000B                   // (0x000b) = IX;  OutputCompareHigh Status TCSR1
*EF52: 86 08          '..'                    LDAA    #$08                    // A = 0x08; Initialisierung TCSR1
*EF54: 97 08          '..'                    STAA    M0008                   // (0x0008) = A; Enable Output compare Interrupt 1 Trigger IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*EF56: C6 10          '..'                    LDAB    #$10                    // B = 0x10; Schleife B
*EF58: 37             '7'             ZEF58   PSHB          <----\            // Push B
*EF59: 4F             'O'                     CLRA              |             // A = 0x00
*EF5A: BD EF B8       '...'                   JSR     ZEFB8     |             // Call 0xefb8; Übergabe A; Sende Header 16 * 0x00
*EF5D: 33             '3'                     PULB              |             // Pop B
*EF5E: 5A             'Z'                     DECB              |             // B--
*EF5F: 26 F7          '&.'                    BNE     ZEF58 >---/             // if (z) == 0; Schleife 16 Runden
*EF61: 0F             '.'                     SEI                             // Set Interrupt Mask / NMI
*EF62: C6 50          '.P'                    LDAB    #$50                    // B = 0x50
*EF64: BD EA CF       '...'                   JSR     ZEACF                   // Call 0xeacf Zeitschleife 10ms; 800 ms
*EF67: BD EA DF       '...'                   JSR     ZEADF                   // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*EF6A: 43             'C'                     COMA                            // !A
*EF6B: 97 8D          '..'                    STAA    M008D                   // (0x008d) = A
*EF6D: 0D             '.'                     SEC                             // carry = 1
*EF6E: 0E             '.'                     CLI                             // Clear Interrupt Mask = 0
*EF6F: C6 04          '..'                    LDAB    #$04                    // B = 0x04; Schleife B
*EF71: 37             '7'             ZEF71   PSHB          <---\             // Push B
*EF72: 96 8D          '..'                    LDAA    M008D     |             // A = (0x008d); 0x00 oder 0xff
*EF74: BD EF B8       '...'                   JSR     ZEFB8     |             // Call 0xefb8; Übergabe A; Sende Header-Start 4 * 0xff
*EF77: 33             '3'                     PULB              |             // Pop B
*EF78: 5A             'Z'                     DECB              |             // B--
*EF79: 26 F6          '&.'                    BNE     ZEF71 >---/             // if (z) == 0; Schleife 4 Runden
*EF7B: DE 2D          '.-'                    LDX     M002D                   // IX = (0x002d); Text-Startadresse
*EF7D: 09             '.'                     DEX                             // IX--; erstes Kompromat 0x28 "("
*EF7E: A6 00          '..'                    LDAA    ,X                      // A = (IX)
*EF80: BD EF B8       '...'                   JSR     ZEFB8                   // Call 0xefb8; Übergabe A; Sende erstes chiffriertes Zeichen "("
*EF83: DE 2D          '.-'                    LDX     M002D                   // IX = (0x002d); Zeiger Text-Startadresse
*EF85: 5F             '_'                     CLRB                            // B = 0
*EF86: 37             '7'             ZEF86   PSHB                <---\       // Push B
*EF87: A6 00          '..'                    LDAA    ,X              |       // A = (IX)
*EF89: BD EF B8       '...'                   JSR     ZEFB8           |       // Call 0xefb8; Übergabe A; Sende chiffrierten Klartext
*EF8C: DE 2D          '.-'                    LDX     M002D           |       // IX = (0x002d); Text-Startadresse
*EF8E: 9C 27          '.''                    CPX     M0027           |       // cmp IX, (0x0027); Text-Endadresse
*EF90: 24 0E          '$.'                    BCC     ZEFA0   >---\   |       // if (c) == 0; Sprung sende letztes chiffriertes Zeichen
*EF92: 08             '.'                     INX             |   |   |       // IX++
*EF93: DF 2D          '.-'                    STX     M002D   |   |   |       // (0x002d) = IX
*EF95: 33             '3'                     PULB            |   |   |       // Pop B
*EF96: 5A             'Z'                     DECB            |   |   |       // B--;  erste Runde 0xff, 0xfe ..., danach 8 Runden
*EF97: 2A ED          '*.'                    BPL     ZEF86   | >---+       // if (n) == 0; bit7 == 0; Schleife 8 Runden; Prüfung auf  B > 0
*EF99: 75 80 02       'u..'                   EIM     #$80,M0002  |   |       // (0x0002) XOR= 0x80; Port-17 LED OFF
*EF9C: C6 08          '..'                    LDAB    #$08        |   |       // B = 0x08
*EF9E: 20 E6          ' .'                    BRA     ZEF86       | >---/       // JR 0xef86
                                              |
*EFA0: 33             '3'             ZEFA0   PULB          <---/             // Pop B; Stack bereinigen
*EFA1: C6 20          '. '                    LDAB    #$20                    // B = 0x20; Schleife B
*EFA3: 37             '7'             ZEFA3   PSHB            <---\         // Push B
*EFA4: 86 04          '..'                    LDAA    #$04      |           // A = 0x04
*EFA6: BD EF B8       '...'                   JSR     ZEFB8     |           // Call 0xefb8; Übergabe A;Sende Fuß-Daten 0x04
*EFA9: 33             '3'                     PULB              |           // Pop B
*EFAA: 5A             'Z'                     DECB              |           // B--
*EFAB: 26 F6          '&.'                    BNE     ZEFA3 >---/           // if (z) == 0; Schleife 32 Runden
*EFAD: C6 32          '.2'                    LDAB    #$32                  // B = 0x32
*EFAF: BD EA CF       '...'                   JSR     ZEACF                 // Call 0xeacf Zeitschleife 10ms; 500ms
*EFB2: 86 04          '..'                    LDAA    #$04                  // A = 0x04
*EFB4: 97 08          '..'                    STAA    M0008                 // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*EFB6: 0F             '.'                     SEI                           // Set Interrupt Mask / NMI
*EFB7: 39             '9'                     RTS                           // RETURN

// Call Sende-Routine, Übergabe A
*EFB8: 36             '6'             ZEFB8   PSHA                          // Push A
*EFB9: C6 08          '..'                    LDAB    #$08                  // B = 0x08 Schleifenzähler
*EFBB: 0C             '.'                     CLC                           // Lösche carry
*EFBC: 79 00 44       'y.D'           ZEFBC   ROL     >M0044 <---\          // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c
*EFBF: 3E             '>'                     WAI               |           // Wait for Interrupt
*EFC0: 47             'G'                     ASRA              |           // A7 -> A7 ... A0 -> c Shift Rechts A7 konstant
*EFC1: 5A             'Z'                     DECB              |           // B--
*EFC2: 26 F8          '&.'                    BNE     ZEFBC  >---/          // if (z) == 0  Schleife 8 Runden
// A über carry nach (0x0044) geschoben
*EFC4: 33             '3'                     PULB                          // Pop B; B aus Push A
```

```
*EFC5: 7D 00 8D       '}..'                   TST     >M008D                          // (0x008d) == 0; bit7 == 1
*EFC8: 2B 0A          '+.'                    BNI     ZEFD4    >---\                  // if (n) == 1
*EFCA: 4F             'O'                     CLRA                |                  // A = 0
*EFCB: CE 00 07       '...'                   LDX     #M0007      |                  // IX = 0x0007; Schleifenzähler
*EFCE: 57             'W'             ZEFCE   ASRB           <----\ |                // B7 -> B7 ... B0 -> c Shift Rechts B7
*EFCF: 89 00          '..'                    ADCA    #$00       | |                // A += carry; Übertrag aus Schiebe rechts B
*EFD1: 09             '.'                     DEX                | |                // IX--
*EFD2: 26 FA          '&.'                    BNE     ZEFCE >----/ |                // if (z) == 0 Schleife 7 Runden
*EFD4: 47             'G'             ZEFD4   ASRA              <---/                // A7 -> A7 ... A0 -> c Shift Rechts A7
*EFD5: 79 00 44       'y.D'                   ROL     >M0044                          // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c; Übertrag aus A nach (0x0044)
*EFD8: 3E             '>'                     WAI                                     // Wait for Interrupt
*EFD9: 0D             '.'                     SEC                                     // carry = 1
*EFDA: 79 00 44       'y.D'                   ROL     >M0044                          // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c
*EFDD: 3E             '>'                     WAI                                     // Wait for Interrupt
*EFDE: DC 0B          '..'                    LDD     M000B                           // D = (0x000b); OutputCompareHigh Status TCSR1
*EFE0: D3 5A          '.Z'                    ADDD    M005A                           // D += (0x005a); Baudraten-Daten
*EFE2: DD 0B          '..'                    STD     M000B                           // (0x000b) = D; OutputCompareHigh Status TCSR1
*EFE4: 39             '9'                     RTS                                     // RETURN

// Einsprung Interrupt-Sprungverteiler; Ausgabe 0 oder 1 auf Port-14 und -24 TX und Printer
*EFE5: 96 44          '.D'          hdlr_SWI2 LDAA    M0044                         // A = (0x0044)
*EFE7: 44             'D'                     LSRA                                    // 0 -> A7 ... A0 -> c
*EFE8: 24 08          '$.'                    BCC     ZEFF2            >---\          // if (c) == 0
*EFEA: 72 10 02       'r..'                   OIM     #$10,M0002           |          // (Port-1 Data) |= 0x10; Port-14 TX "1" Senden
*EFED: 72 10 03       'r..'                   OIM     #$10,M0003           |          // (Port-2 Data) |= 0x10; Port-24 Printer "1" Senden
*EFF0: 20 06          ' .'                    BRA     ZEFF8   >---\        |          // JR 0xeff8
                                                                |        |
*EFF2: 71 EF 02       'q..'           ZEFF2   AIM     #$EF,M0002  |    <---/          // (Port-1 Data) &= 0xef; Port-14 TX "0" Senden
*EFF5: 71 EF 03       'q..'                   AIM     #$EF,M0003  |                   // (Port-2 Data) &= 0xef; Port-24 Printer "0" Senden
*EFF8: 96 08          '..'            ZEFF8   LDAA    M0008  <----/                   // A = (0x0008); Lese TCSR1 Timer Control Status 1
*EFFA: DC 0B          '..'                    LDD     M000B                           // D = (0x000b); OutputCompareHigh Status TCSR1
*EFFC: D3 5A          '.Z'                    ADDD    M005A                           // D += (0x005a); Baudraten-Daten
*EFFE: DD 0B          '..'                    STD     M000B                           // (0x000b) = D; OutputCompareHigh Status TCSR1
*F000: 3B             ';'                     RTI                                     // RETI (Inclusive Pop über alle Register)

// Call Sprungverteiler 33 Empfangen und Dechiffrieren via Sprungverteiler 35
*F001: 7F 00 CC       '...'                   CLR     >M00CC                          // (0x00cc) = 0
*F004: 20 04          ' .'                    BRA     ZF00A                           // JR 0xf00a -> weiter mit Sprungverteiler 35

// Call Sprungverteiler 35 Empfangen und Dechiffrieren
*F006: 86 FF          '..'                    LDAA    #$FF                            // A = 0xff
*F008: 97 CC          '..'                    STAA    M00CC                           // (0x00cc) = A
// Einsprung aus Sprungverteiler 33; Übergabe (0x00cc) = 0x00 oder 0xff
*F00A: BD F2 34       '..4'           ZF00A   JSR     ZF234                           // Call 0xf234; Textspeicher verfügbar
*F00D: BD F3 C8       '...'                   JSR     ZF3C8                           // Call 0xf3c8; Lese Port-1 Data; Rückgabe B mit Bit 2 = 1
*F010: BD F2 5E       '..^'           ZF010   JSR     ZF25E                           // Call 0xf25e; Bereitschaft Ready to Receive
*F013: 86 08          '..'                    LDAA    #$08                            // A = 0x08
*F015: 97 8A          '..'                    STAA    M008A                           // (0x008a) = A
*F017: BD F2 7E       '..~'           ZF017   JSR     ZF27E <----\                    // Call 0xF27E  Bearbeitung RAM bis 0x03b6, Rückgabe B (1 oder 0); Modem
*F01A: BD F2 CC       '...'                   JSR     ZF2CC      |                    // Call 0xf2cc; Übergabe B (1 oder 0); Rückgabe IX = 0x0008 oder 0x0001
*F01D: BD F2 FD       '...'                   JSR     ZF2FD      |                    // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F020: 4D             'M'                     TSTA               |                    // A == 0?; MSB == 1 n-Flag, A == 0 z-Flag
*F021: 26 F4          '&.'                    BNE     ZF017 >----+                    // if (z) == 0
*F023: 96 03          '..'                    LDAA    M0003      |                    // A = (0x0003); Port-2 Data
*F025: 88 08          '..'                    EORA    #$08       |                    // A XOR= 0x08 Port-23 Ready vom Drucker
*F027: 94 8A          '..'                    ANDA    M008A      |                    // A &= (0x008a) = 0x08
*F029: 27 EC          ''.'                    BEQ     ZF017 >----/                    // if (z) == 1 Port-23 Ready vom Drucker
*F02B: BD F2 E1       '...'                   JSR     ZF2E1                           // Call 0xf2e1; LCD Kommando 0x01, 0x80 Zeitschleife; Abfrage Port-1-Data
*F02E: BD F2 FD       '...'                   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F031: 97 8D          '..'                    STAA    M008D                           // (0x008d) = A
*F033: BD F2 FD       '...'                   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F036: 97 8E          '..'                    STAA    M008E                           // (0x008e) = A
*F038: BD F2 FD       '...'                   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F03B: 97 8F          '..'                    STAA    M008F                           // (0x008f) = A
*F03D: BD F2 FD       '...'                   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F040: 97 90          '..'                    STAA    M0090                           // (0x0090) = A
*F042: 4F             'O'                     CLRA                                    // A = 0
*F043: CE 00 8D       '...'                   LDX     #M008D                          // IX = 0x008d
*F046: C6 08          '..'            ZF046   LDAB    #$08  <----\                    // B = 0x08 Schleifenzähler
*F048: 64 00          'd.'            ZF048   LSR     ,X    <---\|                    // 0 -> IX7 ... IX0 -> c (0x008d ... 008f)
*F04A: 89 00          '..'                    ADCA    #$00      ||                    // A += C; Carry einsammeln, Anzahl der '1' in den 3 bytes
*F04C: 5A             'Z'                     DECB              ||                    // B--
*F04D: 26 F9          '&.'                    BNE     ZF048 >---/|                    // if (z) == 0 Schleife 8 Zeichen / schieben
*F04F: 08             '.'                     INX               |                    // IX++
*F050: 8C 00 90       '...'                   CPX     #M0090    |                    // cmp IX, 0x0090
*F053: 23 F1          '#.'                    BLS     ZF046  >---/                    // if (c) | (z) == 1; 3 byte * 8 bit geschoben
*F055: 8B F4          '..'                    ADDA    #$F4                            // A += 0xf4
*F057: 56             'V'                     RORB                                    // Rotiere B; c -> B7 ... B0 -> c; Einschieben carry nach B Bit 7
*F058: 81 0D          '..'                    CMPA    #$0D                            // CMP A, 0x0d
*F05A: 24 09          '$.'                    BCC     ZF065 >---\                     // if (c) == 0 Text i.o.
*F05C: CE F2 0F       '...'                   LDX     #MF20F    |                     // IX = 0xf20f; "** BAD TEXT **
*F05F: BD EC C5       '...'                   JSR     ZECC5     |                     // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*F062: 7E F1 3C       '~.<'                   JMP     ZF13C     |                     // JP 0xf13c
                                                                |
// Weiter, geladener Text OK                                    |
*F065: D7 8D          '..'            ZF065   STAB    M008D <---/                     // (0x008d) = B
*F067: BD F2 FD       '...'                   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B

// Call Empfangen (LOAD/Receiv) Übergabe A (0x003c); Rückgabe
*F06A: DE 2D          '.-'            ZF06A   LDX     M002D                           // IX = (0x002d); Text-Startadresse
*F06C: 09             '.'                     DEX                                     // IX--; erstes Kompromat Zeichen-Zeiger
*F06D: A7 00          '..'                    STAA    ,X                              // (IX) = A
*F06F: 7F 00 5D       '..]'           ZF06F   CLR     >M005D                          // (0x005d) = 0
*F072: BD F2 FD       '...'           ZF072   JSR     ZF2FD                           // Call 0xf2fd Empfangsroutine; Rückgabe A, B
*F075: 7D 00 8D       '}..'                   TST     >M008D                          // (0x008d) == 0; bit7 == 1
```

```
*F078: 2B 18        '+.'                  BNI     ZF092 >----\                          // if (n) == 1
*F07A: DA 8C        '..'                  ORAB    M008C      |                          // B |= (0x008c)
*F07C: D7 8C        '..'                  STAB    M008C      |                          // (0x008c) = B
*F07E: 8A 80        '..'                  ORAA    #$80       |                          // A |= 0x80; Setze Bit 7
*F080: 81 8D        '..'                  CMPA    #$8D       |                          // CMP A, 0x8d
*F082: 27 0E        ''.'                  BEQ     ZF092 >----\                          // if (z) == 1
*F084: 84 7F        '..'                  ANDA    #$7F       |                          // A &= 0x7f Maskiere
*F086: 81 04        '..'                  CMPA    #$04       |                          // CMP A, 0x04
*F088: 23 08        '#.'                  BLS     ZF092 >----+                          // if (c) | (z) == 1
*F08A: 81 20        '. '                  CMPA    #$20       |                          // CMP A, 0x20
*F08C: 24 04        '$.'                  BCC     ZF092 >----+                          // if (c) == 0
*F08E: 4F           'O'                   CLRA               |                          // A = 0
*F08F: 72 80 8C     'r..'                 OIM     #$80,M008C |                          // (0x008c) |= 0x80; Setze Bit 7
*F092: 97 8B        '..'            ZF092 STAA    M008B  <---/                          // (0x008b) = A; 0x00, 0x20, 0x04, 0x8d, ..
*F094: 81 04        '..'                  CMPA    #$04                                 // CMP A, 0x04
*F096: 27 05        ''.'                  BEQ     ZF09D  >---\                          // if (z) == 1
*F098: 7F 00 5E     '..^'                 CLR     >M005E     |                          // (0x005e) = 0
*F09B: 20 0C        ' .'                  BRA     ZF0A9      | >---\                    // JR 0xf0a9
// JP A == 0x04                                               |     |
*F09D: 7C 00 5E     '|.^'           ZF09D INC     >M005E <---/     |                    // (0x005e)++
*F0A0: D6 5E        '.^'                  LDAB    M005E            |                    // B = (0x005e)
*F0A2: C1 03        '..'                  CMPB    #$03             |                    // cmp B, 0x03
*F0A4: 25 03        '%.'                  BCS     ZF0A9        >---+                    // if (c) == 1
*F0A6: 7E F1 30     '~.0'                 JMP     ZF130            |       >---\        // JP 0xf130
                                                                   |           |
// JP                                                              |           |
*F0A9: D6 5C        '.\'            ZF0A9 LDAB    M005C        <---/           |        // B = (0x005c)
*F0AB: C1 28        '.('                  CMPB    #$28                         |        // cmp B, 0x28; 40 Elemente
*F0AD: 25 2B        '%+'                  BCS     ZF0DA    >---\               |        // if (c) == 1
*F0AF: D6 5F        '._'                  LDAB    M005F        |               |        // B = (0x005f)
*F0B1: 5A           'Z'                   DECB                 |               |        // B--
*F0B2: 2B 0F        '+.'                  BNI     ZF0C3        | >---\         |        // if (n) == 1; bit7 == 1
*F0B4: C1 03        '..'                  CMPB    #$03         |     |         |        // cmp B, 0x03
*F0B6: 26 20        '& '                  BNE     ZF0D8        |     | >---\   |        // if (z) == 0
*F0B8: D7 5F        '. '                  STAB    M005F        |     |     |   |        // (0x005f) = B
*F0BA: 86 C1        '..'                  LDAA    #$C1         |     |     |   |        // A = 0xc1; Sonderzeichen Kopfzeile
*F0BC: BD EC 3B     '..;'                 JSR     ZEC3B        |     |     |   |        // Call 0xec3b; LCD COMMAND; Übergabe A
*F0BF: 96 8B        '..'                  LDAA    M008B        |     |     |   |        // A = (0x008b)
*F0C1: 20 17        ' .'                  BRA     ZF0DA    >---+     |     |   |        // JR 0xf0da
// JP                                                              |     |   |
*F0C3: 75 80 02     'u..'           ZF0C3 EIM     #$80,M0002   | <---/     |   |        // (Port-1 Data) XOR= 0x80; Port-17 LED OFF
*F0C6: 7B 80 02     '{..'                 TIM     #$80,M0002   |           |   |        // (Port-1 Data) &= 0x80 Port-17 LED ON
*F0C9: 27 04        ''.'                  BEQ     ZF0CF        | >---\     |   |        // if (z) == 1
*F0CB: 86 05        '..'                  LDAA    #$05         |     |     |   |        // A = 0x05; Sonderzeichen Cursor
*F0CD: 20 02        ' .'                  BRA     ZF0D1        | >---+     |   |        // JR 0xf0d1
// JP                                                           |     |     |   |
*F0CF: 86 20        '. '            ZF0CF LDAA    #$20         |     |     |   |        // A = 0x20 " " auf LCD
*F0D1: BD EC 57     '..W'           ZF0D1 JSR     ZEC57        | <---/     |   |        // Call 0xec57; LCD Data; Übergabe A
*F0D4: 96 8B        '..'                  LDAA    M008B        |           |   |        // A = (0x008b)
*F0D6: C6 08        '..'                  LDAB    #$08         |           |   |        // B = 0x08
*F0D8: D7 5F        '._'            ZF0D8 STAB    M005F        |       <---/   |        // (0x005f) = B
*F0DA: DE 2D        '.-'            ZF0DA LDX     M002D    <---/               |        // IX = (0x002d)
*F0DC: A7 00        '..'                  STAA    ,X                           |        // (IX) = A
*F0DE: 08           '.'                   INX                                  |        // IX++
*F0DF: DF 2D        '.-'                  STX     M002D                        |        // (0x002d) = IX
*F0E1: D6 5C        '.\'                  LDAB    M005C                        |        // B = (0x005c)
*F0E3: C1 28        '.('                  CMPB    #$28                         |        // cmp B, 0x28
*F0E5: 24 0D        '$.'                  BCC     ZF0F4      >---\             |        // if (c) == 0
*F0E7: 5C           '\'                   INCB                   |             |        // B++
*F0E8: D7 5C        '.\'                  STAB    M005C          |             |        // (0x005c) = B
*F0EA: 7D 00 8D     '}..'                 TST     >M008D         |             |        // (0x008d) == 0?;  MSB == 1 n-Flag (0x008d) == 0 z-Flag
*F0ED: 27 02        ''.'                  BEQ     ZF0F1 >---\    |             |        // if (z) == 1 -> LCD Data; Übergabe A
*F0EF: 86 3F        '.?'                  LDAA    #$3F      |    |             |        // A = 0x3f; "?" auf LCD
*F0F1: BD EC 57     '..W'           ZF0F1 JSR     ZEC57 <---/    |             |        // Call 0xec57; LCD Data; Übergabe A
*F0F4: DE 2D        '.-'            ZF0F4 LDX     M002D      <---/             |        // IX = (0x002d)
*F0F6: 9C 27        '.''                  CPX     M0027                        |        // cmp IX, (0x0027)
*F0F8: 24 31        '$1'                  BCC     ZF12B       >---\            |        // if (c) == 0
*F0FA: 7D 00 CC     '}..'                 TST     >M00CC          |            |        // (0x00cc) == 0?;  MSB == 1 n-Flag (0x00cc) == 0 z-Flag
*F0FD: 27 13        ''.'                  BEQ     ZF112     >---\ |            |        // if (z) == 1
*F0FF: 71 BF 02     'q..'                 AIM     #$BF,M0002    | |            |        // (Port-1 Data) = (Port1-Data & 0xbf) Maskiere
*F102: B6 40 40     '.@@'                 LDAA    M4040         | |            |        // A = (0x4040) ??
*F105: 72 40 02     'r@.'                 OIM     #$40,M0002    | |            |        // M = 0x40 or (0x0002); Port-14 TX
*F108: 84 02        '..'                  ANDA    #$02          | |            |        // A &= 0x02
*F10A: 27 06        ''.'                  BEQ     ZF112    >----+ |            |        // if (z) == 1
*F10C: BD F2 52     '..R'                 JSR     ZF252          | |            |        // Call 0xf252; Kopiere (0x0025) > (0x002d), A = 0xff -> (0x003a), (0x008d) = 0x0
*F10F: 7E F0 10     '~..'                 JMP     ZF010          | |            |        // JP 0xf010
                                                                  | |            |
// JP                                                             | |            |
*F112: 7D 00 8C     '}..'           ZF112 TST     >M008C <------/ |            |        // (0x008c) == 0; bit7 == 1
*F115: 2B 03        '+.'                  BNI     ZF11A >---\     |            |        // if (n) == 1
*F117: 7E F0 6F     '~.o'                 JMP     ZF06F     |     |            |        // JP 0xf06f
                                                              |     |            |
// JP                                                         |     |            |
*F11A: DE CA        '..'            ZF11A LDX     M00CA <---/     |            |        // IX = (0x00ca)
*F11C: 08           '.'                   INX                     |            |        // IX++
*F11D: DF CA        '..'                  STX     M00CA           |            |        // (0x00ca) = IX
*F11F: 7C 00 5D     '|.]'                 INC     >M005D          |            |        // (0x005d)++
*F122: 96 5D        '.]'                  LDAA    M005D           |            |        // A = (0x005d)
*F124: 81 64        '.d'                  CMPA    #$64            |            |        // CMP A, 0x64
*F126: 2C 14        ',.'                  BGE     ZF13C           |  >---\     |        // if (n) == (v) == 1/0;
*F128: 7E F0 72     '~.r'                 JMP     ZF072           |      |     |        // JP 0xf072
                                                                  |      |     |
// JP                                                             |      |     |
*F12B: BD EA 99     '...'           ZF12B JSR     ZEA99      <----/      |     |        // Call 0xea99; Modem ON Übergabe A Port-1x, B Zeitschleife 60-100-60ms
*F12E: 20 3A        ' :'                  BRA     ZF16A                  |     |  >---\ // JR 0xf16a
```

```
                                                                  |     |      |
// JP                                                             |     |      |
*F130: DE 2D          '.-'            ZF130   LDX     M002D      | <---/      |       // IX = (0x002d)
*F132: 09             '.'                     DEX                |            |       // IX--
*F133: 09             '.'                     DEX                |            |       // IX--
*F134: 7D 00 8D       '}..'                   TST     >M008D     |            |       // (0x008d) == 0; bit7 == 1
*F137: 2B 01          '+.'                    BNI     ZF13A >---\|            |       // if (n) == 1
*F139: 09             '.'                     DEX               ||            |       // IX--
*F13A: DF 2D          '.-'            ZF13A   STX     M002D <---/|            |       // (0x002d) = IX
                                                                 |            |
// Call JP                                                       |            |
*F13C: 0F             '.'             ZF13C   SEI                <---/       |       // Set Interrupt Mask / NMI
*F13D: DC 27          '.''                    LDD     M0027               |       // D = (0x0027); Text-Endadresse
*F13F: 93 2D          '.-'                    SUBD    M002D               |       // D -= (0x002d); Test-Startadresse -> Länge
*F141: DE 2D          '.-'                    LDX     M002D               |       // IX = (0x002d); Zeiger Text-Start
*F143: 9C 25          '.%'                    CPX     M0025               |       // cmp IX, (0x0025); Zeiger Text-Ende
*F145: 22 15          '".'                    BHI     ZF15C >---\         |       // if (c) & (z) == 0
*F147: 09             '.'                     DEX               |         |       // IX--
*F148: DF 2D          '.-'                    STX     M002D     |         |       // (0x002d) = IX; Text-Start
*F14A: C3 00 02       '...'                   ADDD    #M0002    |         |       // D += 0x0002; Länge + 2
*F14D: BD EA 73       '..s'                   JSR     ZEA73     |         |       // Call 0xea73; Übergabe D
*F150: BD E2 12       '...'                   JSR     ZE212     |         |       // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*F153: 6F 00          'o.'                    CLR     ,X        |         |       // (IX) = 0x00
*F155: 6F 01          'o.'                    CLR     $01,X     |         |       // (IX + 0x01) = 0x00
*F157: BD E2 1A       '...'                   JSR     ZE21A     |         |       // Call 0xe21a  Init Textlänge Zeiger
*F15A: 20 42          ' B'                    BRA     ZF19E     | >-\     |       // JR 0xf19e
// JP                                                            |   |     |       //
*F15C: 7D 00 8D       '}..'           ZF15C   TST     >M008D <--/   |     |       // (0x008d) == 0; bit7 == 0
*F15F: 2A 04          '*.'                    BPL     ZF165 >---\   |     |       // if (n) == 0
*F161: C3 00 01       '...'                   ADDD    #M0001    |   |     |       // D += 0x0001
*F164: 09             '.'                     DEX               |   |     |       // IX--
*F165: DF 27          '.''            ZF165   STX     M0027 <---/   |     |       // (0x0027) = IX
*F167: BD EA 73       '..s'                   JSR     ZEA73         |     |       // Call 0xea73; Übergabe D
                                                                    |     |       //
// JP                                                               |     |       //
*F16A: BD E2 12       '...'           ZF16A   JSR     ZE212         |     <---/   // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*F16D: DC 27          '.''                    LDD     M0027         |           // D = (0x0027); Zeiger Text-aktual Position
*F16F: 93 25          '.%'                    SUBD    M0025         |           // D -= (0x0025); Zeiger Text-Ende
*F171: C3 00 02       '...'                   ADDD    #M0002        |           // D += 0x0002
*F174: 9A 8D          '..'                    ORAA    M008D         |           // A |= (0x008d)
*F176: ED 00          '..'                    STD     ,X            |           // (IX) = D
*F178: DE 25          '.%'                    LDX     M0025         |           // IX = (0x0025); Zeiger Text-Ende
*F17A: DF 2D          '.-'                    STX     M002D         |           // (0x002d) = IX; Zeiger Text-Start
*F17C: 7D 00 8D       '}..'                   TST     >M008D        |           // (0x008d) == 0; bit7 == 1
*F17F: 2B 1D          '+.'                    BNI     ZF19E      >--+           // if (n) == 1
*F181: 09             '.'                     DEX                |           // IX--
*F182: A6 00          '..'                    LDAA    ,X         |           // A = (IX)
*F184: 81 0A          '..'                    CMPA    #$0A       |           // CMP A, 0x0a
*F186: 2D 04          '-.'                    BLT     ZF18C >----\  |           // if (n) XOR (v) == 1
*F188: 81 50          '.P'                    CMPA    #$50       |  |           // CMP A, 0x50
*F18A: 2F 04          '/.'                    BLE     ZF190      |  | >--\      // if (z) & (n | v) == 1
*F18C: 86 28          '.('            ZF18C   LDAA    #$28  <----/  |    |      // A = 0x28
*F18E: A7 00          '..'                    STAA    ,X            |    |      // (IX) = A
*F190: 97 3C          '.<'            ZF190   STAA    M003C         | <--/      // (0x003c) = A
*F192: BD E5 BC       '...'                   JSR     ZE5BC         |           // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F195: BD EA 01       '...'                   JSR     ZEA01         |           // Call 0xea01; Textanfang-Ende GTX gefunden?
*F198: 7F 00 30       '..0'                   CLR     >M0030        |           // (0x0030) = 0x00
*F19B: BD E5 BC       '...'                   JSR     ZE5BC         |           // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F19E: 72 80 02       'r..'           ZF19E   OIM     #$80,M0002 <--/           // (0x0002) |= 0x80 Port-17 LED ON
*F1A1: 86 04          '..'                    LDAA    #$04                      // A = 0x04
*F1A3: 97 08          '..'                    STAA    M0008                     // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*F1A5: 86 C5          '..'                    LDAA    #$C5                      // A = 0xc5
*F1A7: DE CA          '..'                    LDX     M00CA                     // IX = (0x00ca)
*F1A9: 27 23          ''#'                    BEQ     ZF1CE >---\               // if (z) == 1
*F1AB: CE 00 00       '...'                   LDX     #M0000    |               // IX = 0x0000
*F1AE: DC 27          '.''                    LDD     M0027     |               // D = (0x0027); Zeiger Text-aktual Position
*F1B0: 93 25          '.%'                    SUBD    M0025     |               // D -= (0x0025); Zeiger Text-Ende
*F1B2: C3 00 02       '...'                   ADDD    #M0002    |               // D += 0x0002
*F1B5: 08             '.'             ZF1B5   INX               | <--\        // IX++
*F1B6: 93 CA          '..'                    SUBD    M00CA     |    |        // D -= (0x00CA)
*F1B8: 22 FB          '".'                    BHI     ZF1B5     | >--/        // if (c) & (z) == 0 Schleife bei D == 0x0000 und carry = 1
*F1BA: 86 C5          '..'                    LDAA    #$C5      |             // A = 0xc5
*F1BC: 8C 00 64       '..d'                   CPX     #M0064    |             // cmp IX, 0x0064
*F1BF: 24 0D          '$.'                    BCC     ZF1CE >---+             // if (c) == 0
*F1C1: 4A             'J'                     DECA              |             // A--
*F1C2: 8C 00 0A       '...'                   CPX     #M000A    |             // cmp IX, 0x000a
*F1C5: 24 07          '$.'                    BCC     ZF1CE >---+             // if (c) == 0
*F1C7: 4A             'J'                     DECA              |             // A--
*F1C8: 8C 00 03       '...'                   CPX     #M0003    |             // cmp IX, 0x0003
*F1CB: 24 01          '$.'                    BCC     ZF1CE >---+             // if (c) == 0
*F1CD: 4A             'J'                     DECA              |             // A--
*F1CE: 97 3A          '.:'            ZF1CE   STAA    M003A <---/             // (0x003a) = A
*F1D0: 16             '.'                     TAB                             // B = A
*F1D1: 86 C6          '..'                    LDAA    #$C6                    // A = 0xc6
*F1D3: 10             '.'                     SBA                             // A -= B; Berechneter Schleifenzähler
*F1D4: 36             '6'             ZF1D4   PSHA          <---\             // Push A
*F1D5: BD EA 94       '...'                   JSR     ZEA94     |             // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*F1D8: C6 0A          '..'                    LDAB    #$0A      |             // B = 0x0a
*F1DA: BD EA CF       '...'                   JSR     ZEACF     |             // Call 0xeacf Zeitschleife 10ms; 100ms
*F1DD: 32             '2'                     PULA              |             // Pop A
*F1DE: 4A             'J'                     DECA              |             // A--
*F1DF: 26 F3          '&.'                    BNE     ZF1D4 >---/             // if (z) == 0 Schleife >= 0x39 ?
*F1E1: BD EC A2       '...'                   JSR     ZECA2                   // Call 0xeca2
*F1E4: BD E3 E7       '...'                   JSR     ZE3E7                   // Call 0xe3e7
*F1E7: CE F2 1F       '...'                   LDX     #MF21F                  // IX = 0xF21f; "RECEICED : /:00 BYTE"
```

```
*F1EA: BD EC EA       '...'                   JSR     ZECEA                   // Call 0xecea; Textausgabe LCD, Übergabe IX
*F1ED: DC 27          '.''                    LDD     M0027                   // D = (0x0027); Zeiger Text-aktual Position
*F1EF: 93 25          '.%'                    SUBD    M0025                   // D -= (0x0025); Zeiger Text-Ende
*F1F1: 27 03          ''.'                    BEQ     ZF1F6 >---\             // if (z) == 1
*F1F3: C3 00 02       '...'                   ADDD    #M0002    |             // D += 0x0002
*F1F6: CE 00 D8       '...'           ZF1F6   LDX     #M00D8 <--/             // IX = 0x00d8
*F1F9: BD EC F6       '...'                   JSR     ZECF6                   // Call 0xecf6 -> Call 0xeeae; Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*F1FC: 7F 00 39       '..9'                   CLR     >M0039                  // (0x0039) = 0x00
*F1FF: 7D 00 CC       '}..'                   TST     >M00CC                  // (0x00cc) == 0?; MSB == 1 n-Flag (0x00cc) == 0 z-Flag
*F202: 26 03          '&.'                    BNE     ZF207 >---\             // if (z) == 0
*F204: 7E E1 6C       '~.l'                   JMP     ZE16C     |             // JP 0xe16c
                                                              |
                                                              |
*F207: C6 64          '.d'            ZF207   LDAB    #$64  <---/             // B = 0x64
*F209: BD EA CF       '...'                   JSR     ZEACF                   // Call 0xeacf Zeitschleife 10ms; 1 Sekunde
*F20C: 7E F0 0A       '~..'                   JMP     ZF00A                   // JP 0xf00a -> Sprungverteiler 35 "Receive"

// MF20F
*F20F: 2A 2A 2A 20 42 41 44 20 54 45 58 54 20 2A 2A AA                        // "*** BAD TEXT ***

// MF21F
*F21F: 52 45 43 45 49 56 45 44 20 3A 20 2F 3A 30 30 20                        // "RECEICED : /:00 BYTES"
*F22F: 42 59 54 45 D3

// Call Textspeicher verfügbar
*F234: BD             '..'              ZF234   JSR     ZE20D                   // Call 0xe20d; Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*F237: 27 06          ''.'                      BEQ     ZF23F >---\             // if (z) == 1
*F239: BD E2 F3       '...'                     JSR     ZE2F3     |             // Call 0xe2f3; Prüfung Textspeicher verfügbar
*F23C: BD E2 1A       '...'                     JSR     ZE21A     |             // Call  0xe21a  Init Textlänge Zeiger
*F23F: 7F 00 37       '..7'           ZF23F   CLR     >M0037 >--/             // (0x0037) = 0x00
*F242: 7F 00 08       '...'                     CLR     >M0008                  // (0x0008) = 0x00; Port21 Output, TCSR1 Timer Control Status 1
*F245: BD E2 C0       '...'                     JSR     ZE2C0                   // Call 0xe2c0; Übergabe (0x002f); Rückgabe B (0x0034), IX (0x002b)
*F248: CC 1F FF       '...'                     LDD     #M1FFF                  // D = 0x1fff
*F24B: 93 23          '.#'                      SUBD    M0023                   // D -= (0x0023)
*F24D: DD 5A          '.Z'                      STD     M005A                   // (0x005a) = D
*F24F: BD EA 26       '..&'                     JSR     ZEA26                   // Call 0xea26; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D

// Call Kopiere (0x0025) > (0x002d), A = 0xff -> (0x003a), (0x008d) = 0x0
*F252: DE 25          '.%'            ZF252   LDX     M0025                   // IX = (0x0025); Zeiger Text-Ende
*F254: DF 2D          '.-'                    STX     M002D                   // (0x002d) = IX; Zeiger Text-Start
*F256: 86 FF          '..'                    LDAA    #$FF                    // A = 0xff
*F258: 97 3A          '.:'                    STAA    M003A                   // (0x003a) = A
*F25A: 7F 00 8D       '...'           ZF25A   CLR     >M008D                  // (0x008d) = 0x00
*F25D: 39             '9'                     RTS                             // RETURN

// Call Bereitschaft Ready to Receive
*F25E: CE 00 00       '...'           ZF25E   LDX     #M0000                  // IX = 0x0000
*F261: DF CA          '..'                    STX     M00CA                   // (0x00ca) = IX
*F263: 86 29          '.)'            ZF263   LDAA    #$29                    // A = 0x29; 40 Elemente
*F265: 97 32          '.2'                    STAA    M0032                   // (0x0032) = A
*F267: CE F9 46       '..F'                   LDX     #MF946                  // IX = 0xf946; "READY TO RECEIVE (/:999"
*F26A: BD ED CD       '...'                   JSR     ZEDCD                   // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*F26D: DC 5A          '.Z'                    LDD     M005A                   // D = (0x005a)
*F26F: CE 00 F0       '...'                   LDX     #M00F0                  // IX = 0x00f0
*F272: BD EC F6       '...'           ZF272   JSR     ZECF6                   // Call 0xecf6 -> Call 0xeeae; Call Längenberechnung D, IX 0x00e4; Rückgabe Hunderter/Zehner (0x00e6, 0x00e7)
*F275: 71 7F 02       'q..'                   AIM     #$7F,M0002              // (Port1-Data &= 0x7f) Maskiere ASCII
*F278: 86 FF          '..'                    LDAA    #$FF                    // A = 0xff
*F27A: 97 3A          '.:'                    STAA    M003A                   // (0x003a) = A
*F27C: 0E             '.'                     CLI                             // Clear Interrupt Mask = 0
*F27D: 39             '9'                     RTS                             // RETURN

// Rückgabe B (1 oder 0); Modem; Modem; Modem
*F27E: 75 09 8A       'u..'           ZF27E   EIM     #$09,M008A <--\         // (0x008A) XOR= 0x09; Lösche/Setze Bit 0 und 3
*F281: CE 00 64       '..d'                   LDX     #M0064        |         // IX = 0x0064
*F284: 96 03          '..'            ZF284   LDAA    M0003         | <----\  // A = (0x0003); Port-2 Data
*F286: 88 08          '..'                    EORA    #$08          |      |  // A XOR= 0x08;  Port-23 Ready
*F288: 94 8A          '..'                    ANDA    M008A         |      |  // A &= (0x008a)
*F28A: 27 F2          ''.'                    BEQ     ZF27E    >----+      |  // if (z) == 1; falls A == 0 ist; Warte auf Daten von Port-2
*F28C: 09             '.'                     DEX                   |      |  // IX--
*F28D: 26 F5          '&.'                    BNE     ZF284         | >----/  // if (z) == 0 Schleife IX > 0; 100 Runden
*F28F: CE 01 2C       '..,'                   LDX     #M012C        |         // IX = 0x012c
*F292: 09             '.'             ZF292   DEX                   | <----\  // IX--
*F293: 27 E9          ''.'                    BEQ     ZF27E    >----+      |  // if (z) == 1; IX == 0
*F295: 96 03          '..'                    LDAA    M0003         |      |  // A = (0x0003); Port-2 Data
*F297: 88 08          '..'                    EORA    #$08          |      |  // A XOR= 0x08;  Port-23 Ready
*F299: 94 8A          '..'                    ANDA    M008A         |      |  // A &= (0x008a)
*F29B: 26 F5          '&.'                    BNE     ZF292         | >----/  // if (z) == 0
*F29D: CE 00 00       '...'                   LDX     #M0000        |         // IX = 0x0000
*F2A0: 08             '.'             ZF2A0   INX                   | <----\  // IX++
*F2A1: 8C 06 D6       '...'                   CPX     #M06D6        |      |  // cmp IX, 0x06d6
*F2A4: 24 D8          '$.'                    BCC     ZF27E    >----+      |  // if (c) == 0
*F2A6: 96 03          '..'                    LDAA    M0003         |      |  // A = (0x0003); Port-2 Data
*F2A8: 88 08          '..'                    EORA    #$08          |      |  // A XOR= 0x08;  Port-23 Ready
*F2AA: 94 8A          '..'                    ANDA    M008A         |      |  // A &= (0x008a)
*F2AC: 27 F2          ''.'                    BEQ     ZF2A0         | >----/  // if (z) == 1
*F2AE: 8C 01 8C       '...'                   CPX     #M018C        |         // cmp IX, 0x018c
*F2B1: 25 CB          '%.'                    BCS     ZF27E    >----+         // if (c) == 1
*F2B3: C6 01          '..'                    LDAB    #$01          |         // B = 0x01
*F2B5: 8C 01 B5       '...'                   CPX     #M01B5        |         // cmp IX, 0x01b5; 0x00xy ... 0x01b4 bearbeitet?
*F2B8: 23 11          '#.'                    BLS     ZF2CB         | >---\   // if (c) | (z) == 1 RETURN
*F2BA: 8C 03 18       '...'                   CPX     #M0318        |     |   // cmp IX, 0x0318; 0x01b4 ... 0x0318 bearbeitet?
*F2BD: 25 BF          '%.'                    BCS     ZF27E    >----+     |   // if (c) == 1
*F2BF: 5A             'Z'                     DECB                  |     |   // B--
*F2C0: 8C 03 6B       '..k'                   CPX     #M036B        |     |   // cmp IX, 0x036b; 0x0318 ... 0x036b bearbeitet?
*F2C3: 23 06          '#.'                    BLS     ZF2CB         | >---+   // if (c) | (z) == 1 RETURN
```

```
*F2C5: 8C 06 2F       '../'                   CPX     #M062F        |     |     // cmp IX, 0x062f; 0x036b ... 0x062f bearbeitet?
*F2C8: 25 B4          '%.'                    BCS     ZF27E    >----/     |     // if (c) == 1
*F2CA: 5A             'Z'                     DECB                        |     // B--
*F2CB: 39             '9'             ZF2CB   RTS                   <---/     // RETURN

// Call Übergabe B (1 oder 0); Rückgabe IX = 0x0008 oder 0x0001
*F2CC: D7 39          '.9'            ZF2CC   STAB    M0039                   // (0x0039) = B
*F2CE: 5C             '\'                     INCB                            // B++; 2 oder 1
*F2CF: 58             'X'                     ASLB                            // c <- B7 ... B0 <- 0 Shift left; 4 oder 2
*F2D0: CE F3 ED       '...'                   LDX     #MF3ED                  // IX = 0xf3ed Data
*F2D3: 3A             ':'                     ABX                             // IX += B (0xf3ef) = 0x0126 oder (0xf3f1) = 0x0087
*F2D4: EE 00          '..'                    LDX     ,X                      // IX = (IX); 0x0126 oder 0x0087
*F2D6: DF 60          '.`'                    STX     M0060                   // (0x0060) = IX
*F2D8: CE F3 F3       '...'                   LDX     #MF3F3                  // IX = 0xf3f3
*F2DB: 3A             ':'                     ABX                             // IX += B; (0xf3f5) = 0x0008 oder (0xf3f7) = 0x0001
*F2DC: EE 00          '..'                    LDX     ,X                      // IX = (IX); 0x0008 oder 0x0001
*F2DE: DF 62          '.b'                    STX     M0062                   // (0x0062) = IX
*F2E0: 39             '9'                     RTS                             // RETURN

// Call LCD Kommando 0x01, 0x80 Zeitschleife; Abfrage Port-1-Data
*F2E1: 86 01          '..'            ZF2E1   LDAA    #$01                    // A = 0x01; Clear Display
*F2E3: BD EC 37       '..7'                   JSR     ZEC37                   // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*F2E6: 86 80          '..'                    LDAA    #$80                    // A = 0x80; DD-RAM LCD
*F2E8: BD EC 37       '..7'                   JSR     ZEC37                   // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*F2EB: 7F 00 5C       '..\'                   CLR     >M005C                  // (0x005c) = 0x00
*F2EE: C6 3C          '.<'                    LDAB    #$3C                    // B = 0x3c
*F2F0: BD EA CF       '...'                   JSR     ZEACF                   // Call 0xeacf Zeitschleife 10ms; 600ms
*F2F3: 7F 00 5E       '..^'                   CLR     >M005E                  // (0x005e) = 0x00
*F2F6: 7F 00 5F       '._'                    CLR     >M005F                  // (0x005f) = 0x00
*F2F9: 71 7F 02       'q..'                   AIM     #$7F,M0002              // (Port-1 Data) &= 0x7f Maskiere
*F2FC: 39             '9'                     RTS                             // RETURN

// Call Empfangsroutine; Rückgabe A; Rückgabe A; Rückgabe A, B
*F2FD: 7F 00 8C       '...'           ZF2FD   CLR     >M008C                  // (0x008c) = 0x00
*F300: 96 03          '..'            ZF300   LDAA    M0003 <---\             // A = (0x0003); Port-2 Data
*F302: 88 08          '..'                    EORA    #$08      |             // A XOR= 0x08;  Port-23 Ready
*F304: 94 8A          '..'                    ANDA    M008A     |             // A &= (0x008a)
*F306: 27 F8          ''.'                    BEQ     ZF300 >---/             // if (z) == 1; warte auf Werteänderung
*F308: 96 03          '..'            ZF308   LDAA    M0003 <---\             // A = (0x0003); Port-2 Data
*F30A: 88 08          '..'                    EORA    #$08      |             // A XOR= 0x08;  Port-23 Ready
*F30C: 94 8A          '..'                    ANDA    M008A     |             // A &= (0x008a)
*F30E: 26 F8          '&.'                    BNE     ZF308 >---/             // if (z) == 0; Warte auf Werteänderung
*F310: 86 08          '..'                    LDAA    #$08                    // A = 0x08
*F312: 97 B3          '..'                    STAA    M00B3                   // (0x00b3) = A
*F314: D6 39          '.9'                    LDAB    M0039                   // B = (0x0039)
*F316: 5C             '\'                     INCB                            // B++
*F317: 58             'X'                     ASLB                            // c <- B7 ... B0 <- 0 Shift left
*F318: CE F3 E7       '...'                   LDX     #MF3E7                  // IX = 0xf3e7; Data Prozent-Werte
*F31B: 3A             ':'                     ABX                             // IX += B; 0xf3e7 += B
*F31C: EE 00          '..'                    LDX     ,X                      // IX = (IX)
*F31E: 09             '.'             ZF31E   DEX          <---\              // IX--
*F31F: 26 FD          '&.'                    BNE     ZF31E >---/             // if (z) == 0 Zeitschleife in Abhängigkeit von (IX + B)
*F321: 7F 00 8B       '...'                   CLR     >M008B                  // (0x008b) = 0x00
*F324: CE 00 07       '...'           ZF324   LDX     #M0007            <---\ // IX = 0x0007
*F327: 5F             '_'                     CLRB                        |   // B = 0x00
*F328: 96 03          '..'            ZF328   LDAA    M0003          <---\  |  // A = (Port-2 Data)
*F32A: 88 08          '..'                    EORA    #$08               |  |  // A XOR= 0x08; Port-23 Ready
*F32C: 94 8A          '..'                    ANDA    M008A              |  |  // A &= (0x008a)
*F32E: 27 03          ''.'                    BEQ     ZF333 >---\        |  |  // if (z) == 1 Prüfe bit3
*F330: 5C             '\'                     INCB             |         |  |  // B++
*F331: 20 03          ' .'                    BRA     ZF336    |  >--\  |  |  // JR 0xf336
                                                               |     |  |  |
*F333: 01             '.'             ZF333   NOP          <---/     |  |  |  // NOP
*F334: 20 00          ' .'                    BRA     ZF336        >--+  |  |  // JR 0xf336
                                                                     |  |  |
*F336: 3C             '<'             ZF336   PSHX                <---/  |  |  // Push IX
*F337: DE 62          '.b'                    LDX     M0062               |  |  // IX = (0x0062)
*F339: 09             '.'             ZF339   DEX          <----\         |  |  // IX--
*F33A: 26 FD          '&.'                    BNE     ZF339 >---/         |  |  // if (z) == 0 Zeitschleife
*F33C: 38             '8'                     PULX                        |  |  // Pop IX
*F33D: 09             '.'                     DEX                         |  |  // IX--
*F33E: 26 E8          '&.'                    BNE     ZF328          >---/  |  // if (z) == 0 Wiederhole abfrage Port-2
*F340: C1 02          '..'                    CMPB    #$02                   |  // cmp B, 0x02
*F342: 23 07          '#.'                    BLS     ZF34B     >---\        |  // if (c) | (z) == 1
*F344: C1 04          '..'                    CMPB    #$04        |        |  // cmp B, 0x04
*F346: 22 03          '".'                    BHI     ZF34B     >---+        |  // if (c) & (z) == 0
*F348: 72 80 8C       'r..'                   OIM     #$80,M008C    |        |  // (0x008c) |= 0x80; Setze bit 7
*F34B: 54             'T'             ZF34B   LSRB              <---/        |  // 0 -> B7 ... B0 -> c
*F34C: 54             'T'                     LSRB                           |  // 0 -> B7 ... B0 -> c
*F34D: 54             'T'                     LSRB                           |  // 0 -> B7 ... B0 -> c
*F34E: 76 00 8B       'v..'                   ROR     >M008B                 |  // Rotiere (0x008b); c -> (0x008b)7 ... (0x008b)0 -> c
*F351: DE 60          '.`'                    LDX     M0060                  |  // IX = (0x0060)
*F353: 09             '.'             ZF353   DEX          <---\          |  // IX--
*F354: 26 FD          '&.'                    BNE     ZF353 >---/          |  // if (z) == 0
*F356: 7A 00 B3       'z..'                   DEC     >M00B3               |  // (0x00B3)--
*F359: 2E C9          '..'                    BGT     ZF324             >---/  // if ((Z) & (v|n)) == 0
*F35B: 5F             '_'                     CLRB                           // B = 0x00
*F35C: 96 8B          '..'                    LDAA    M008B                  // A = (0x008b)
*F35E: 7D 00 8D       '}..'                   TST     >M008D                 // (0x008d) == 0; bit7 == 1
*F361: 2B 13          '+.'                    BNI     ZF376    >---\         // if (n) == 1 RETURN
*F363: CE 00 08       '...'                   LDX     #M0008       |         // IX = 0x0008 Schleifenzähler
*F366: 46             'F'             ZF366   RORA         <----\  |         // Rotiere A; c -> A7 ... A0 -> c
*F367: C9 00          '..'                    ADCB    #$00      |  |         // B += C + 0x00; Hole das Carry von RORA nach B
*F369: 09             '.'                     DEX               |  |         // IX--
*F36A: 26 FA          '&.'                    BNE     ZF366 >---/  |         // if (z) == 0 Schleife 8 Runden
```

```
*F36C: C4 01          '..'                    ANDB    #$01         |                    // B &= 0x01 Maskiere, nur Bit 0
*F36E: 56             'V'                     RORB                 |                    // Rotiere B; c -> B7 ... B0 -> c
*F36F: 56             'V'                     RORB                 |                    // Rotiere B; c -> B7 ... B0 -> c; Bit 0 nach Bit 7 geschoben
*F370: 96 8B          '..'                    LDAA    M008B        |                    // A = (0x008b)
*F372: 84 7F          '..'                    ANDA    #$7F         |                    // A &= 0x7f Maskiere
*F374: 97 8B          '..'                    STAA    M008B        |                    // (0x008b) = A
*F376: 39             '9'             ZF376   RTS                <---/                 // RETURN

// Call Sprungverteiler 36 Senden - Tonbandausgabe/DUMP Texte 1 bis max 62
*F377: CE F8 65       '..e'                   LDX     #MF865                             // IX = 0xf865; "START 'RECORD' ON TAPE "
*F37A: BD EC CA       '...'                   JSR     ZECCA                              // Call 0xecca; Textausgabe auf LCD, Warten auf Tastendruck; Übergabe IX
*F37D: 81 B3          '..'                    CMPA    #$B3                               // CMP A, 0xB3; "DUMP"
*F37F: 26 3C          '&<'                    BNE     ZF3BD         >---\                // if (z) == 0 RETURN
*F381: CE F9 AD       '...'                   LDX     #MF9AD             |                // IX = 0xf9ad; "< TRANSMITTING >"
*F384: BD EC EA       '...'                   JSR     ZECEA              |                // Call 0xecea; Textausgabe LCD, Übergabe IX
*F387: 86 01          '..'                    LDAA    #$01               |                // A = 0x01
*F389: 97 39          '.9'                    STAA    M0039              |                // (0x0039) = A
*F38B: BD F3 BE       '...'                   JSR     ZF3BE              |                // Call 0xf3be; Lese Port-1 Data; Rückgabe B mit Bit 4 = 1
*F38E: 96 2F          './'                    LDAA    M002F              |                // A = (0x002f); Zählervariable 0x1 ... 0x62
*F390: 97 72          '.r'                    STAA    M0072              |                // (0x0072) = A
*F392: 7F 00 2F       '../'                   CLR     >M002F             |                // (0x002f) = 0x00; Zählervariable 0x1 ... 0x62
*F395: 7C 00 2F       '|./'           ZF395   INC     >M002F      <---\  |                // (0x002f)++; Zählervariable 0x1 ... 0x62
*F398: 96 2F          './'                    LDAA    M002F          | |                // A = (0x002f); Zählervariable 0x1 ... 0x62
*F39A: 81 63          '.c'                    CMPA    #$63           | |                // CMP A, 0x63
*F39C: 22 12          '".'                    BHI     ZF3B0 >---\    | |                // if (c) & (z) == 0
*F39E: BD E2 0D       '...'                   JSR     ZE20D     |    | |                // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*F3A1: 27 F2          ''.'                    BEQ     ZF395     | >--+ |                // if (z) == 1
*F3A3: BD E2 1A       '...'                   JSR     ZE21A     |    | |                // Call  0xe21a  Init Textlänge Zeiger
*F3A6: BD EF 37       '..7'                   JSR     ZEF37     |    | |                // Call 0xef37 Send-Akustik-Koppler
*F3A9: C6 96          '..'                    LDAB    #$96      |    | |                // B = 0x96
*F3AB: BD EA CF       '...'                   JSR     ZEACF     |    | |                // Call 0xeacf Zeitschleife 10ms; 1,5 Sekunden
*F3AE: 20 E5          ' .'                    BRA     ZF395     | >--/ |                // JR 0xf395
// JP                                                           |      |
*F3B0: 96 72          '.r'            ZF3B0   LDAA    M0072 <---/      |                // A = (0x0072)
*F3B2: 97 2F          './'                    STAA    M002F            |                // (0x002f) = A; Zählervariable 0x1 ... 0x62
*F3B4: BD F3 D5       '...'                   JSR     ZF3D5            |                // Call 0xf3d5; Init Port-2-Data
*F3B7: 7F 00 39       '..9'                   CLR     >M0039           |                // (0x0039) = 0x00
*F3BA: BD E2 1A       '...'                   JSR     ZE21A            |                // Call  0xe21a  Init Textlänge Zeiger
*F3BD: 39             '9'             ZF3BD   RTS                  <---/                // RETURN

// Call Lese Port-1 Data; Rückgabe B mit Bit 4 = 1
*F3BE: D6 02          '..'            ZF3BE   LDAB    M0002                              // B = (Port-1 Data)
*F3C0: C4 F9          '..'                    ANDB    #$F9                               // B &= 0xf9 Maskiere
*F3C2: CA 10          '..'                    ORAB    #$10                               // B |= 0x10: setze Bit 4
*F3C4: D7 02          '..'                    STAB    M0002                              // (Port-1 Data) = B
*F3C6: 20 08          ' .'                    BRA     ZF3D0 >---\                        // JR 0xf3d0; Zeitschleife 0x32 Runden
                                                             |
// Call Lese Port-1 Data; Rückgabe B mit Bit 2 = 1           |
*F3C8: D6 02          '..'            ZF3C8   LDAB    M0002    |                         // B = (Port-1 Data)
*F3CA: C4 F9          '..'                    ANDB    #$F9     |                         // B &= 0xf9 Maskiere
*F3CC: CA 04          '..'                    ORAB    #$04     |                         // B |= 0x04; setze bit 2
*F3CE: D7 02          '..'                    STAB    M0002    |                         // (Port-1 Data) = B
*F3D0: C6 32          '.2'            ZF3D0   LDAB    #$32  <--/                         // B = 0x32
*F3D2: 7E EA CF       '~..'                   JMP     ZEACF                              // JP 0xeacf Zeitschleife 10ms; 500ms

// Call Init Port-2-Data
*F3D5: 71 FE 03       'q..'           ZF3D5   AIM     #$FE,M0003                         // (Port-2 Data) = (Port2-Data & 0xfe) bit0
*F3D8: 86 C8          '..'                    LDAA    #$C8                               // A = 0xc8
*F3DA: 97 02          '..'                    STAA    M0002                              // (Port-1 Data) = A
*F3DC: 86 CA          '..'                    LDAA    #$CA                               // A = 0xca
*F3DE: 97 02          '..'                    STAA    M0002                              // (Port-1 Data) = A
*F3E0: 39             '9'                     RTS                                        // RETURN

// Data MF3E1; Baud-Rate
*F3E1: 0D 05                                                                             //  300 Baud
*F3E3: 06 83                                                                             //  600 Baud
*F3E5: 03 41                                                                             // 1200 Baud

// Data MF3E7 Prozent-Anzeige
*F3E7: 04 60                                                                             // = 100%
*F3E9: 02 30                                                                             // = 50%
*F3EB: 01 04                                                                             // = 25%
*F3ED: 02 78                                                                             // = 60%
*F3EF: 01 26                                                                             // = 30%
*F3F1: 00 87                                                                             // = 10%
*F3F3: 00 14                                                                             // = 5%
*F3F5: 00 08                                                                             // = 2%
*F3F7: 00 01                                                                             // = 1%

// Call Tastendruck
*F3F9: BD F4 37       '..7'           ZF3F9   JSR     ZF437                <---\         // Call 0xf437; Tastaturabfrage Rückgabe A+B, IX = 0x009d, z-Flag
*F3FC: 27 04          ''.'                    BEQ     ZF402 >----\             |         // if (z) == 1
*F3FE: 81 FF          '..'                    CMPA    #$FF       |             |         // CMP A, 0xff
*F400: 26 08          '&.'                    BNE     ZF40A      | >----\      |         // if (z) == 0
*F402: 7F 00 C5       '...'           ZF402   CLR     >M00C5 <---/       |     |         // (0x00c5) = 0x00
*F405: 0E             '.'             ZF405   CLI                        |     | <---\   // Clear Interrupt Mask = 0
*F406: 1A             '.'                     SLP                        |     |     |   // Sleep; Schöner Warten
*F407: 0F             '.'                     SEI                        |     |     |   // Set Interrupt Mask / NMI
*F408: 20 EF          ' .'                    BRA     ZF3F9              |  >--+     |   // Funktion neu starten warten auf Werteänderung
                                                                         |     |     |
*F40A: D1 C5          '..'            ZF40A   CMPB    M00C5  <-----------/     |     |   // cmp B, (0x00c5)
*F40C: 26 1A          '&.'                    BNE     ZF428   >---\            |     |   // if (z) == 0
*F40E: 7D 00 38       '}.8'                   TST     >M0038      |            |     |   // (0x0038) == 0?;  MSB == 1 n-Flag (0x0038) == 0 z-Flag
*F411: 27 F2          ''.'                    BEQ     ZF405       |            | >---/   // if (z) == 1; Schleife
*F413: 7A 00 9B       'z..'                   DEC     >M009B      |            |         // (0x009b)--
```

```
*F416: 27 0A          ''.'                    BEQ     ZF422 >---\ |             |                 // if (z) == 1
*F418: 0E             '.'                     CLI               | |             |                 // Clear Interrupt Mask = 0
*F419: CE 04 E2       '...'                   LDX     #M04E2    | |             |                 // IX = 0x04e2
*F41C: 09             '.'             ZF41C   DEX         <--\  | |             |                 // IX--
*F41D: 26 FD          '&.'                    BNE     ZF41C >-/ | |             |                 // if (z) == 0; Zeitschleife
*F41F: 0F             '.'                     SEI               | |             |                 // Set Interrupt Mask / NMI
*F420: 20 D7          ' .'                    BRA     ZF3F9     | |         >---/                 // JR 0xf3f9; Tastendruck; Rückgabe A
                                                                | |
*F422: C6 0C          '..'            ZF422   LDAB    #$0C  <---/ |                               // B = 0x0c
*F424: D7 9B          '..'                    STAB    M009B       |                               // (0x009b) = B
*F426: 20 06          ' .'                    BRA     ZF42E       |  >---\                        // JR 0xf42e
                                                                  |      |
*F428: D7 C5          '..'            ZF428   STAB    M00C5   <---/      |                        // (0x00c5) = B
*F42A: C6 42          '.B'                    LDAB    #$42               |                        // B = 0x42; "B"
*F42C: D7 9B          '..'                    STAB    M009B              |                        // (0x009b) = B
*F42E: 36             '6'             ZF42E   PSHA                    >---/                        // Push A
*F42F: BD EA B0       '...'                   JSR     ZEAB0                                        // Call 0xeab0; Modem ON Übergabe A Port-14, B Zeitschleife 10ms
*F432: 32             '2'                     PULA                                                 // Pop A
*F433: 7F 00 38       '..8'                   CLR     >M0038                                       // (0x0038) = 0x00
*F436: 39             '9'                     RTS                                                  // RETURN

// Call Tastaturabfrage Rückgabe A (Tastencode oder 0xff), B (Shift), IX = 0x009d, z-Flag
*F437: CE 00 9D       '...'           ZF437   LDX     #M009D                                       // IX = 0x009d
*F43A: 3C             '<'                     PSHX                                                 // Push IX
*F43B: 86 09          '..'                    LDAA    #$09                                         // A = 0x09
*F43D: 97 A6          '..'                    STAA    M00A6                                        // (0x00a6) = A; Schleifenzähler für folgenden Sprung
*F43F: CE 40 01       '.@.'                   LDX     #M4001                                       // IX = 0x4001; Tastatur-Port-Data
*F442: DF A7          '..'                    STX     M00A7                                        // (0x00a7) = IX; Zeiger
*F444: 71 BF 02       'q..'                   AIM     #$BF,M0002                                   // (Port-1 Data) = (Port1-Data & 0xbf) Maskiere
*F447: B6 40 1F       '.@.'                   LDAA    M401F                                        // A = (0x401f) ????
*F44A: 26 07          '&.'                    BNE     ZF453                 >---\                  // if (z) == 0; Z == 1 Übergabe 0x009d, 0x00a6 Schleifenzähler; 0x00a7 IX(0x4001)
*F44C: 72 40 02       'r@.'                   OIM     #$40,M0002              |                    // (0x0002) |= 0x40 Port-16  Keyb = "1"
*F44F: 97 C4          '..'                    STAA    M00C4                   |                    // (0x00c4) = A; Tastatur ASCII
*F451: 38             '8'                     PULX                            |                    // Pop IX
*F452: 39             '9'                     RTS                             |                    // RETURN
                                                                              |
// JP Z == 1 Übergabe 0x009d, 0x00a6 Schleifenzähler; (0x00a7) IX = 0x4001     |
*F453: DE A7          '..'            ZF453   LDX     M00A7 <---*-------------/                    // IX = (0x00a7); Zeiger auf 0x4001
*F455: A6 00          '..'                    LDAA    ,X        |                                  // A = (IX); Tastatur Port
*F457: 9A A8          '..'                    ORAA    M00A8     |                                  // A |= (0x00a8)
*F459: 98 A8          '..'                    EORA    M00A8     |                                  // A XOR= (0x00a8); 0x01 Lösche Bit 0
*F45B: 38             '8'                     PULX              |                                  // Pop IX; 0x009d
*F45C: A7 00          '..'                    STAA    ,X        |                                  // (IX) = A
*F45E: 08             '.'                     INX               |                                  // IX++; 0x009e
*F45F: 3C             '<'                     PSHX              |                                  // Push IX
*F460: DC A7          '..'                    LDD     M00A7     |                                  // D = (0x00a7); Zeiger auf 0x4001
*F462: 05             '.'                     ASLD              |                                  // c <- A7 ... A0 <- B7 ... B0 <- 0 Shift left; 1000 0000 0000 0001; 0x8002
*F463: 88 C0          '..'                    EORA    #$C0      |                                  // A XOR= 0xc0; 0100 0000 0000 0010; 0x4002
*F465: DD A7          '..'                    STD     M00A7     |                                  // (0x00a7) = D
*F467: 7A 00 A6       'z..'                   DEC     >M00A6    |                                  // (0x00A6)--
*F46A: 26 E7          '&.'                    BNE     ZF453 >---/                                  // if (z) == 0 Schleife 8 Runden
*F46C: 72 40 02       'r@.'                   OIM     #$40,M0002                                   // (Port-1 Data) |= 0x40; Port-16 Keyboard = "1"
*F46F: 38             '8'                     PULX                                                 // Pop IX; 0x09e
*F470: 96 A0          '..'                    LDAA    M00A0                                        // A = (0x00a0)
*F472: 84 81          '..'                    ANDA    #$81                                         // A &= 0x81 Maskiere
*F474: 81 81          '..'                    CMPA    #$81                                         // CMP A, 0x81
*F476: 27 35          ''5'                    BEQ     ZF4AD                       >---\            // if (z) == 1
*F478: 97 35          '.5'                    STAA    M0035                           |            // (0x0035) = A; Tastatur z/s
*F47A: 98 A0          '..'                    EORA    M00A0                           |            // A XOR= (0x00a0); A -= 0x81
*F47C: 97 A0          '..'                    STAA    M00A0                           |            // (0x00a0) = A
*F47E: 86 CF          '..'                    LDAA    #$CF                            |            // A = 0xcf; LCD Kopfzeile SHIFT LOCK
*F480: BD EC 37       '..7'                   JSR     ZEC37                           |            // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*F483: 96 35          '.5'                    LDAA    M0035                           |            // A = (0x0035); Tastatur z/s
*F485: 27 04          ''.'                    BEQ     ZF48B     >----\                |            // if (z) == 1
*F487: 86 05          '..'                    LDAA    #$05           |                |            // A = 0x05; Ausgabe Sonderzeichen Cursor
*F489: 20 02          ' .'                    BRA     ZF48D >---\    |                |            // JR 0xf48d
// Init A = 0x20                                                |    |                |
*F48B: 86 20          '. '            ZF48B   LDAA    #$20       | <--/             |            // A = 0x20; Ausgabe " " auf LCD
// JP weiter                                                    |                  |
*F48D: BD EC 52       '..R'           ZF48D   JSR     ZEC52 <---/                  |            // Call 0xec52; LCD Data nach Busy; Übergabe A
*F490: 5F             ' '                     CLRB                                 |            // B = 0x00
// Schleife                                                                        |
*F491: 09             '.'             ZF491   DEX           <--------\             |            // IX--; 0x00a5
*F492: 6D 00          'm.'                    TST     ,X             |             |            // (IX) == 0?; MSB == 1 n-Flag (IX) == 0 z-Flag
*F494: 27 0B          ''.'                    BEQ     ZF4A1          | >---\       |            // if (z) == 1
*F496: 86 09          '..'                    LDAA    #$09           |     |       |            // A = 0x09
*F498: 0C             '.'                     CLC                    |     |       |            // carry = 0
*F499: 66 00          'f.'            ZF499   ROR     ,X     <---\   |     |       |            // Rotiere (IX); c -> (IX)7 ... (IX)0 -> c
*F49B: 24 01          '$.'                    BCC     ZF49E >-\ |   |     |       |            // if (c) == 0
*F49D: 5C             '\'                     INCB            | |   |     |       |            // B++; Zähler der "1" in IX 0 ... 8
*F49E: 4A             'J'             ZF49E   DECA          <--/ |   |     |       |            // A--
*F49F: 26 F8          '&.'                    BNE     ZF499 >----/   |     |       |            // if (z) == 0 Schleife 8 Runden
*F4A1: 8C 00 9D       '...'           ZF4A1   CPX     #M009D         | <---/       |            // cmp IX, (0x009d)
*F4A4: 26 EB          '&.'                    BNE     ZF491 >--------/             |            // if (z) == 0 Schleife
*F4A6: 4F             'O'                     CLRA                                 |            // A = 0x00
*F4A7: C1 01          '..'                    CMPB    #$01                         |            // cmp B, 0x01
*F4A9: 25 04          '%.'                    BCS     ZF4AF             >---\      |            // if (c) == 1; B == 0 dann Speicher A
*F4AB: 27 05          ''.'                    BEQ     ZF4B2         >---\    |     |            // if (z) == 1; wenn in IX nur eine "1" war
*F4AD: 86 FF          '..'            ZF4AD   LDAA    #$FF              |    |  <--/            // A = 0xff; Fehler, Rückgabe 0xFF
*F4AF: 97 C4          '..'            ZF4AF   STAA    M00C4   <---\     | <--/               // (0x00c4) = A; Rückgabe ASCII
*F4B1: 39             '9'                     RTS               |     |                        // RETURN
// JP weiter                                                    |     |
*F4B2: BD F7 59       '..Y'           ZF4B2   JSR     ZF759       | <---/                    // Call 0xf759; Adresse 0x0058 Zeiger auf 0x02fb
*F4B5: CE 00 9C       '...'                   LDX     #M009C      |                          // IX = 0x009c
*F4B8: C6 F7          '..'                    LDAB    #$F7        |                          // B = 0xf7
```

```
*F4BA: 08             '.'             ZF4BA   INX           <----\|                          // IX++; 0x009d ...
*F4BB: CB 08          '..'                    ADDB    #$08       ||                          // B += 0x08; 0xff, 0x07, 0x0f, 0x17, ...
*F4BD: 6D 00          'm.'                    TST     ,X         ||                          // (IX) == 0?; MSB == 1 n-Flag (IX) == 0 z-Flag
*F4BF: 27 F9          ''.'                    BEQ     ZF4BA >----/|                          // if (z) == 1; Schleife bis (IX) != 0x00 !!
*F4C1: 5C             '\'             ZF4C1   INCB          <---\ |                          // B++; 0x00, 0x08, 0x10, ...
*F4C2: 66 00          'f.'                    ROR     ,X        | |                          // Rotiere (IX); c -> (IX)7 ... (IX)0 -> c
*F4C4: 24 FB          '$.'                    BCC     ZF4C1 >---/ |                          // if (c) == 0;  bis c == 1
*F4C6: CE FF 02       '...'                   LDX     #MFF02      |                          // IX = 0xff02; Tastatur Tabelle ohne Shift l/r
*F4C9: 96 35          '.5'                    LDAA    M0035       |                          // A = (0x0035); Tastatur z/s
*F4CB: 27 0A          ''.'                    BEQ     ZF4D7 >---\ |                          // if (z) == 1; Übergabe IX 0xff02
*F4CD: CE FF 4A       '..J'                   LDX     #MFF4A    | |                          // IX = 0xff4a; Tastatur Tabelle mit Shift l/r
*F4D0: 81 01          '..'                    CMPA    #$01      | |                          // CMP A, 0x01
*F4D2: 27 03          ''.'                    BEQ     ZF4D7 >---+ |                          // if (z) == 1; Übergabe IX 0xff4a
*F4D4: CE FF 92       '...'                   LDX     #MFF92    | |                          // IX = 0xff92; Tastatur Tabelle mit Shift l/r
*F4D7: 3A             ':'             ZF4D7   ABX           <---/ |                          // IX += B
*F4D8: A6 00          '..'                    LDAA    ,X          |                          // A = (IX)
*F4DA: 7D 00 36       '}.6'                   TST     >M0036      |                          // (0x0036) == 0?;  MSB == 1 n-Flag (0x0036) == 0 z-Flag /* Tastatur z/s
*F4DD: 27 D0          ''.'                    BEQ     ZF4AF >-----+                          // if (z) == 1 (0x00c4) = A; ASCII
*F4DF: 81 61          '.a'                    CMPA    #$61        |                          // CMP A, 0x61; "a"
*F4E1: 25 CC          '%.'                    BCS     ZF4AF >-----+                          // if (c) == 1 (0x00c4) = A; ASCII Shift l/r
*F4E3: 81 7B          '.{'                    CMPA    #$7B        |                          // CMP A, 0x7b; "{"
*F4E5: 24 C8          '$.'                    BCC     ZF4AF >-----+                          // if (c) == 0 (0x00c4) = A; ASCII Shift l/r
*F4E7: 80 20          '. '                    SUBA    #$20        |                          // A-= 0x20; " "
*F4E9: 20 C4          ' .'                    BRA     ZF4AF >-----/                          // JR 0xf4af; (0x00c4) = A; ASCII Shift l/r

// Call JP Kopiere (0x00a9)->byte->(0x00ab); Adresse--; Schleifenzähler: (0x00ad)--
*F4EB: DE A9          '..'            ZF4EB   LDX     M00A9 <---\                          // IX = (0x00a9); Zeiger auf 0x004d, ..
*F4ED: A6 00          '..'                    LDAA    ,X        |                          // A = (IX)
*F4EF: 09             '.'                     DEX               |                          // IX--; 0x004c
*F4F0: DF A9          '..'                    STX     M00A9     |                          // (0x00a9) = IX; Lege Zeiger ab
*F4F2: DE AB          '..'                    LDX     M00AB     |                          // IX = (0x00ab); Zeiger auf 0x004e, 0x0055
*F4F4: A7 00          '..'                    STAA    ,X        |                          // (IX) = A
*F4F6: 09             '.'                     DEX               |                          // IX--; 0x004d, 0x0054
*F4F7: DF AB          '..'                    STX     M00AB     |                          // (0x00ab) = IX; Lege Zeiger ab
*F4F9: DE AD          '..'                    LDX     M00AD     |                          // IX = (0x00ad); Zeiger auf 0x0008
*F4FB: 09             '.'                     DEX               |                          // IX--; 0x0007
*F4FC: DF AD          '..'                    STX     M00AD     |                          // (0x00ad) = IX; lege Wert 0x0007 ab
*F4FE: 26 EB          '&.'                    BNE     ZF4EB >---/                          // if (z) == 0 Schleife
*F500: 39             '9'                     RTS                                          // RETURN

// Call Lösche 0x016a ... 0x018f mit 0x0, Fülle 0x016a und 0x17d + n aus 0xffda ... 0xffe8; n = 0 ... 7
*F501: 9F 6C          '.l'            ZF501   STS     M006C                                // (0x006c) = SP; Rette SP
// Lösche 0x016a ... 0x018f mit "0"
*F503: CE 01 6A       '..j'                   LDX     #M016A                               // IX = 0x016a
*F506: 6F 00          'o.'            ZF506   CLR     ,X    <---\                          // (IX) = 0x00
*F508: 08             '.'                     INX             |                          // IX++
*F509: 8C 01 90       '...'                   CPX     #M0190    |                          // cmp IX, 0x0190
*F50C: 25 F8          '%.'                    BCS     ZF506 >---/                          // if (c) == 1 Schleife
// Übertrage DB 0xffda + n, nach 0x016a + n ( n = 1 ... 7)
*F50E: 8E FF D9       '...'                   LDS     #MFFD9                               // SP = 0xffd9; Nutze SP als Register
*F511: CE 01 6A       '..j'                   LDX     #M016A                               // IX = 0x016a
*F514: 32             '2'             ZF514   PULA          <---\                          // Pop A         SP = 0xffda, A = 0x02, 03, ...
*F515: A7 00          '..'                    STAA    ,X        |                          // (IX) = A
*F517: 08             '.'                     INX               |                          // IX++
*F518: 8C 01 71       '..q'                   CPX     #M0171    |                          // cmp IX, 0x0171
*F51B: 25 F7          '%.'                    BCS     ZF514 >---/                          // if (c) == 1 Schleife 8 Runden
// Übetrage DB von 0xffE0 + n, nach 0x017d + n ( n = 1 ... 7)
*F51D: CE 01 7D       '..}'                   LDX     #M017D                               // IX = 0x017d
*F520: 32             '2'             ZF520   PULA          <---\                          // Pop A         SP = 0xffe1; A = 0x7d, 0x7c, ...
*F521: A7 00          '..'                    STAA    ,X        |                          // (IX) = A
*F523: 08             '.'                     INX               |                          // IX++
*F524: 8C 01 84       '...'                   CPX     #M0184    |                          // cmp IX, 0x0184
*F527: 25 F7          '%.'                    BCS     ZF520 >---/                          // if (c) == 1 Schleife 8 Runden
*F529: 9E 6C          '.l'                    LDS     M006C                                // SP = (0x006c);  SP wiederherstellen
*F52B: 39             '9'             ZF52B   RTS                           <---\          // RETURN
                                                                                    |
// Call Zeichenkonvertierung;                                                       |
*F52C: CE 01 70       '..p'           ZF52C   LDX     #M0170                      |          // IX = 0x0170
*F52F: C6 06          '..'                    LDAB    #$06                        |          // B = 0x06
*F531: 5C             '\'             ZF531   INCB          <---\                 |          // B++
*F532: 08             '.'                     INX               |                 |          // IX++
*F533: 6D 00          'm.'                    TST     ,X        |                 |          // (IX) == 0?;  MSB == 1 n-Flag (IX) == 0 z-Flag
*F535: 26 FA          '&.'                    BNE     ZF531 >---/                 |          // if (z) == 0 Schleife suche bis (IX) 0x0 ist
*F537: 20 04          ' .'                    BRA     ZF53D >---\                 |          // JR 0xf53d
                                                                |                 |
// Call Zeichenkonvertierung Übergabe A,                        |                 |
*F539: 8D C6          '..'            ZF539   BSR     ZF501     |                 |          // Call 0xf501; Lösche 0x016a ... 0x018f mit 0x0, Fülle 0x016a und 0x17d + n mit Werten aus 0xffda ... 0xffe8; n = 0 ... 7
*F53B: C6 07          '..'                    LDAB    #$07      |                 |          // B = 0x07
*F53D: D7 71          '.q'            ZF53D   STAB    M0071 <---/                 |          // (0x0071) = B
*F53F: CE F8 8C       '...'           ZF53F   LDX     #MF88C        <---\         |          // IX = 0xf88c; "CONVERSION CHARACTER . -> 00  + RETURN"
*F542: BD EC EA       '...'                   JSR     ZECEA             |         |          // Call 0xecea; Textausgabe LCD, Übergabe IX
*F545: 86 15          '..'                    LDAA    #$15              |         |          // A = 0x15
*F547: 97 32          '.2'                    STAA    M0032             |         |          // (0x0032) = A
*F549: BD F3 F9       '...'                   JSR     ZF3F9             |         |          // Call 0xf3f9; Tastendruck; Rückgabe A
*F54C: 4D             'M'                     TSTA                      |         |          // A == 0; bit7 == 1
*F54D: 2B DC          '+.'                    BNI     ZF52B             | >---+          // if (n) == 1 -> Return
*F54F: CE 00 E2       '...'                   LDX     #M00E2            |     |          // IX = 0x00e2
*F552: A7 00          '..'                    STAA    ,X                |     |          // (IX) = A
*F554: 36             '6'                     PSHA                      |     |          // Push A
*F555: 86 29          '.)'                    LDAA    #$29              |     |          // A = 0x29; 40 Elemente
*F557: 97 32          '.2'                    STAA    M0032             |     |          // (0x0032) = A
*F559: BD EC F9       '...'                   JSR     ZECF9             |     |          // Call 0xecf9; Übergabe (0x0032)
*F55C: 32             '2'                     PULA                      |     |          // Pop A
*F55D: CE 01 6A       '..j'                   LDX     #M016A            |     |          // IX = 0x016a; gefülltes Array (0xffda) Sonderzeichen
*F560: 5F             '_'                     CLRB                      |     |          // B = 0x00
```

```
*F561: A1 00          '..'              ZF561   CMPA    ,X   <----\       |     |                  // cmp A, (IX)
*F563: 27 0D          ''.'                      BEQ     ZF572     | >---\ |     |                  // if (z) == 1
*F565: 08             '.'                       INX               |     | |     |                  // IX++
*F566: 5C             '\'                       INCB              |     | |     |                  // B++
*F567: D1 71          '.q'                      CMPB    M0071     |     | |     |                  // cmp B, (0x0071); 0xffe0 0x7e
*F569: 25 F6          '%.'                      BCS     ZF561 >---/     | |     |                  // if (c) == 1 Rundenzähler B; Suche B in (0x017n)
*F56B: 5C             '\'                       INCB                    | |     |                  // B++
*F56C: C1 13          '..'                      CMPB    #$13            | |     |                  // cmp B, 0x13; gefunden? nein Abbruch
*F56E: 22 BB          '".'                      BHI     ZF52B           | | >---/                  // if (c) & (z) == 0 -> RETURN
*F570: D7 71          '.q'                      STAB    M0071           | |                        // (0x0071) = B; Speicher B Fundstelle
// Weiter mit der Conversion                                            | |
*F572: A7 00          '..'              ZF572   STAA    ,X          <---/ |                        // (IX) = A; 0x016a ...
*F574: A6 13          '..'                      LDAA    $13,X             |                        // A = (IX + 0x13) 0x017d ...
*F576: DF 5C          '.\'                      STX     M005C             |                        // (0x005c) = IX   0x016a ...  sichern
*F578: 8D 12          '..'                      BSR     ZF58C             |                        // Call 0xf58c  Wandelt Hex in ASCII-Hex-Code 0x0 -> "00" 0xff -> "FF"; Übergabe A; Rückgabe B, A
*F57A: D7 E7          '..'                      STAB    M00E7             |                        // (0x00e7) = B
*F57C: 97 E8          '..'                      STAA    M00E8             |                        // (0x00e8) = A
*F57E: BD EC F9       '...'                     JSR     ZECF9             |                        // Call 0xecf9; Übergabe (0x0032)
*F581: C6 1A          '..'                      LDAB    #$1A              |                        // B = 0x1a
*F583: BD EB 4B       '..K'                     JSR     ZEB4B             |                        // Call 0xeb4b; Hex-Eingabe, Übergabe B
*F586: DE 5C          '.\'                      LDX     M005C             |                        // IX = (0x005c) hole 0x016a ... zurück
*F588: A7 13          '..'                      STAA    $13,X             |                        // (IX + 0x13) = A
*F58A: 20 B3          ' .'                      BRA     ZF53F          >---/                        // JR 0xf53f; nächstes Zeichen Convertieren

// Call Wandelt Hex in ASCII-Hex-Code 0x0 -> "00" 0xff -> "FF"; Übergabe A; Rückgabe B, A
*F58C: 36             '6'               ZF58C   PSHA                                               // Push A
*F58D: 44             'D'                       LSRA                                               // 0 -> A7 ... A0 -> c
*F58E: 44             'D'                       LSRA                                               // 0 -> A7 ... A0 -> c
*F58F: 44             'D'                       LSRA                                               // 0 -> A7 ... A0 -> c
*F590: 44             'D'                       LSRA                                               // 0 -> A7 ... A0 -> c; 0x0n
*F591: 8D 04          '..'                      BSR     ZF597                                      // Call 0xf597 DAA Übergabe A; Rückgabe A
*F593: 16             '.'                       TAB                                                // B = A
*F594: 32             '2'                       PULA                                               // Pop A
*F595: 84 0F          '..'                      ANDA    #$0F                                       // A &= 0x0f Maskiere; 0000 nnnn
// Call Übergabe A
*F597: 8B 90          '..'              ZF597   ADDA    #$90                                       // A += 0x90           1001 nnnn
*F599: 19             '.'                       DAA                                                // Decimal Adjust
*F59A: 89 40          '.@'                      ADCA    #$40                                       // A += carry + 0x40
*F59C: 19             '.'                       DAA                                                // Decimal Adjust
*F59D: 39             '9'                       RTS                                                // RETURN

// Call Sprungverteiler 15
*F59E: 7F 00 37       '..7'                     CLR     >M0037                                     // (0x0037) = 0x00
*F5A1: BD E2 0D       '...'                     JSR     ZE20D                                      // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*F5A4: 27 2E          ''.'                      BEQ     ZF5D4 >---\                                // if (z) == 1
*F5A6: BD EA DF       '...'                     JSR     ZEADF     |                                // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*F5A9: 27 29          ')'                       BEQ     ZF5D4 >---+                                // if (z) == 1
*F5AB: CE F7 83       '...'                     LDX     #MF783    |                                // IX = 0xf783; "  - PLEASE WAIT - "
*F5AE: BD EC EA       '...'                     JSR     ZECEA     |                                // Call 0xecea; Textausgabe LCD, Übergabe IX
*F5B1: BD F6 A5       '...'                     JSR     ZF6A5     |                                // Call 0xf6a5; Init Serielle und Timer
*F5B4: DE 25          '.%'                      LDX     M0025     |                                // IX = (0x0025); Zeiger Text-Ende
*F5B6: DF 2D          '.-'              ZF5B6   STX     M002D     | <---\                          // (0x002d) = IX; Umkopieren Zeiger; Zeiger Text-Start
*F5B8: BD E5 BC       '...'                     JSR     ZE5BC     |     |                          // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F5BB: 8D 5F          '._'                      BSR     ZF61C     |     |                          // Call 0xf61c
*F5BD: DE 2B          '.+'                      LDX     M002B     |     |                          // IX = (0x002b); Zeiger Text-aktual Position
*F5BF: 08             '.'                       INX               |     |                          // IX++
*F5C0: 9C 27          '.''                      CPX     M0027     |     |                          // cmp IX, (0x0027); Zeiger Text-aktual Position
*F5C2: 25 F2          '%.'                      BCS     ZF5B6     | >---/                          // if (c) == 1
*F5C4: 0F             '.'                       SEI               |                                // Set Interrupt Mask / NMI
*F5C5: DE 25          '.%'              ZF5C5   LDX     M0025     | <---\                          // IX = (0x0025)
*F5C7: DF 2D          '.-'                      STX     M002D     |     |                          // (0x002d) = IX; Umkopieren; Zeiger Text-Start-Ende-Länge
*F5C9: BD E5 BC       '...'                     JSR     ZE5BC     |     |                          // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F5CC: 7F 00 30       '..0'                     CLR     >M0030    |     |                          // (0x0030) = 0x00
*F5CF: 86 04          '..'                      LDAA    #$04      |     |                          // A = 0x04
*F5D1: 97 08          '..'                      STAA    M0008     |     |                          // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*F5D3: 39             '9'                       RTS               |     |                          // RETURN
                                                                  |     |
                                                                  |     |
*F5D4: BD EA 94       '...'             ZF5D4   JSR     ZEA94 <---/     |                          // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*F5D7: 20 EC          ' .'                      BRA     ZF5C5       >---/                          // JR 0xf5c5

// Call Prüfe (0x0031) == (0x0070)
*F5D9: 7F 00 31       '..1'             ZF5D9   CLR     >M0031                                     // (0x0031) = 0x00
*F5DC: BD E5 BC       '...'                     JSR     ZE5BC                                      // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F5DF: 86 7F          '..'                      LDAA    #$7F                                       // A = 0x7f
*F5E1: 97 30          '.0'                      STAA    M0030                                      // (0x0030) = A
*F5E3: 7F 00 70       '..p'                     CLR     >M0070                                     // (0x0070) = 0x00
*F5E6: 96 34          '.4'                      LDAA    M0034                                      // A = (0x0034)
*F5E8: 80 28          '.('                      SUBA    #$28                                       // A-= 0x28
*F5EA: 25 02          '%.'                      BCS     ZF5EE >---\                                // if (c) == 1
*F5EC: 97 70          '.p'                      STAA    M0070     |                                // (0x0070) = A
*F5EE: BD ED EC       '...'             ZF5EE   JSR     ZEDEC <---/                                // Call 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe
*F5F1: C6 78          '.x'                      LDAB    #$78                                       // B = 0x78
*F5F3: BD EA CF       '...'                     JSR     ZEACF                                      // Call 0xeacf Zeitschleife 10ms; 1,2 Sekunden
*F5F6: 96 31          '.1'                      LDAA    M0031                                      // A = (0x0031)
*F5F8: 91 70          '.p'                      CMPA    M0070                                      // CMP A, (0x0070)
*F5FA: 24 15          '$.'                      BCC     ZF611    >---\                             // if (c) == 0
*F5FC: 8B 0A          '..'              ZF5FC   ADDA    #$0A  <---\  |                             // A += 0x0a
*F5FE: 97 31          '.1'                      STAA    M0031     |  |                             // (0x0031) = A
*F600: BD ED EC       '...'                     JSR     ZEDEC     |  |                             // Call 0xedec; Ausgabe Freier Speicherplatz, Übergabe, Rückgabe
*F603: C6 3C          '.<'                      LDAB    #$3C      |  |                             // B = 0x3c
*F605: BD EA CF       '...'                     JSR     ZEACF     |  |                             // Call 0xeacf Zeitschleife 10ms; 600ms
*F608: 96 31          '.1'                      LDAA    M0031     |  |                             // A = (0x0031)
*F60A: 91 70          '.p'                      CMPA    M0070     |  |                             // CMP A, (0x0070)
*F60C: 25 EE          '%.'                      BCS     ZF5FC >---/  |                             // if (c) == 1
```

```
*F60E: BD EA AA       '...'                   JSR     ZEAAA        |                  // Call 0xeaaa; Modem ON Übergabe A Port-14, B Zeitschleife  30ms
                                                                   |
// CALL JP                                                         |
*F611: BD F6 A5       '...'           ZF611   JSR     ZF6A5    <---/                  // Call 0xf6a5; Init Serielle und Timer
*F614: 8D 06          '..'                    BSR     ZF61C >---\                     // Call 0xf61c
*F616: 86 04          '..'                    LDAA    #$04      |                     // A = 0x04
*F618: 97 08          '..'                    STAA    M0008     |                     // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*F61A: 0F             '.'                     SEI               |                     // Set Interrupt Mask / NMI
*F61B: 39             '9'                     RTS               |                     // RETURN
                                                                |
// Call                                                         |
*F61C: DE 2B          '.+'            ZF61C   LDX     M002B <---/                     // IX = (0x002b); Zeiger Text-aktual Position
*F61E: A6 00          '..'            ZF61E   LDAA    ,X    <---\                     // A = (IX)
*F620: 09             '.'                     DEX               |                     // IX--
*F621: 84 7F          '..'                    ANDA    #$7F      |                     // A &= 0x7f Maskiere ASCII
*F623: 81 20          '. '                    CMPA    #$20      |                     // CMP A, 0x20 " "
*F625: 27 F7          ''.'                    BEQ     ZF61E >---/                     // if (z) == 1
*F627: 08             '.'                     INX                                     // IX++
*F628: DF 5C          '.\'                    STX     M005C                           // (0x005c) = IX
*F62A: DE 29          '.)'                    LDX     M0029                           // IX = (0x0029)
*F62C: DF B1          '..'            ZF62C   STX     M00B1 <---\                     // (0x00ba) = IX
*F62E: A6 00          '..'                    LDAA    ,X        |                     // A = (IX)
*F630: BD F6 60       '..`'                   JSR     ZF660     |                     // Call 0xf660; Serielle Daten lesen Übergabe A
*F633: DE B1          '..'                    LDX     M00B1     |                     // IX = (0x00b1)
*F635: 08             '.'                     INX               |                     // IX++
*F636: 9C 5C          '.\'                    CPX     M005C     |                     // cmp IX, (0x005c)
*F638: 23 F2          '#.'                    BLS     ZF62C >---/                     // if (c) | (z) == 1
*F63A: 09             '.'                     DEX                                     // IX--
*F63B: A6 00          '..'                    LDAA    ,X                              // A = (IX)
*F63D: 81 8D          '..'                    CMPA    #$8D                            // CMP A, 0x8d
*F63F: 27 04          ''.'                    BEQ     ZF645 >---\                     // if (z) == 1 RETURN
*F641: 86 0D          '..'                    LDAA    #$0D      |                     // A = 0x0d; WR/ZV Enter
*F643: 20 1B          ' .'                    BRA     ZF660     |                     // JR 0xf660 Serielle Daten lesen Übergabe A
*F645: 39             '9'             ZF645   RTS          <----/                     // RETURN

// Call Serielle Daten lesen
*F646: BD F6 A5       '...'           ZF646   JSR     ZF6A5                           // Call 0xf6a5;  Init Serielle und Timer
*F649: CE 00 CD       '...'                   LDX     #M00CD                          // IX = 0x00cd
*F64C: DF B1          '..'            ZF64C   STX     M00B1 <---\                     // (0x00b1) = IX
*F64E: A6 00          '..'                    LDAA    ,X        |                     // A = (IX)
*F650: BD F6 60       '..`'                   JSR     ZF660     |                     // Call 0xf660; Serielle Daten lesen Übergabe A
*F653: DE B1          '..'                    LDX     M00B1     |                     // IX = (0x00b1)
*F655: 08             '.'                     INX               |                     // IX++
*F656: 8C 00 F5       '...'                   CPX     #M00F5    |                     // cmp IX, 0x00f5
*F659: 25 F1          '%.'                    BCS     ZF64C >---/                     // if (c) == 1; Schleife 0x00cd ... 0x00f5
*F65B: 86 0D          '..'                    LDAA    #$0D                            // A = 0x0d; WR/ZV Enter
*F65D: 8D 01          '..'                    BSR     ZF660                           // Call 0xf660; Übergabe A = 0x0d
*F65F: 39             '9'                     RTS                                     // RETURN
```

```
// Call  Serielle Daten lesen; Übergabe A; Rückgabe
*F660: 84 7F          '..'            ZF660   ANDA    #$7F                            // A &= 0x7f; Übergabe z. B. 0x0d; ASCII
*F662: C6 13          '..'                    LDAB    #$13                            // B = 0x13
*F664: CE 01 6A       '..j'                   LDX     #M016A                          // IX = 0x016a
*F667: A1 00          '..'            ZF667   CMPA    ,X    <---\                     // cmp A, (IX)
*F669: 27 06          ''.'                    BEQ     ZF671     |  >----\               // if (z) == 1; gefunden
*F66B: 08             '.'                     INX               |       |               // IX++
*F66C: 5A             'Z'                     DECB              |       |               // B--
*F66D: 26 F8          '&.'                    BNE     ZF667 >---/       |               // if (z) == 0 Schleife 18 Runden; Suche A in (IX)
*F66F: 20 02          ' .'                    BRA     ZF673             |  >----\       // JR 0xf673; nicht gefunden
                                                                        |       |
*F671: A6 13          '..'            ZF671   LDAA    $13,X       <---/       |       // A = (IX + 0x13) -> 0x017d ...; bei gefunden
*F673: 36             '6'             ZF673   PSHA                        <----/       // Push A
*F674: C6 08          '..'                    LDAB    #$08                            // B = 0x08
*F676: 3E             '>'             ZF676   WAI           <---\                     // Wait for Interrupt
*F677: D5 03          '..'                    BITB    M0003     |                     // B &= (Port-2 Data)
*F679: 27 FB          ''.'                    BEQ     ZF676 >---/                     // if (z) == 1 Prüfe bit 3 Port-23 Ready
*F67B: 71 EF 03       'q..'                   AIM     #$EF,M0003                      // (Port-2 Data &= 0xef) Port-2x
*F67E: C6 07          '..'                    LDAB    #$07                            // B = 0x07 Schleifenzähler
*F680: 47             'G'             ZF680   ASRA          <---\                     // A7 -> A7 ... A0 -> c
*F681: 79 00 44       'y.D'                   ROL     >M0044    |                     // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c; 7bit von A nach (0x0044) gespiegelt
*F684: 3E             '>'                     WAI               |                     // Wait for Interrupt
*F685: 5A             'Z'                     DECB              |                     // B--
*F686: 26 F8          '&.'                    BNE     ZF680 >---/                     // if (z) == 0 Schleife 7 Runden
*F688: 32             '2'                     PULA                                    // Pop A aus (IX + 0x13) 0x017d ...
*F689: 5F             ' '                     CLRB                                    // B = 0x00
*F68A: CE 00 07       '...'                   LDX     #M0007                          // IX = 0x0007
*F68D: 47             'G'             ZF68D   ASRA          <---\                     // A7 -> A7 ... A0 -> c Shift Rechts A7
*F68E: C9 00          '..'                    ADCB    #$00      |                     // B += c hole das Carry von ASRA nach B; Anzahl der Bits
*F690: 09             '.'                     DEX               |                     // IX--
*F691: 26 FA          '&.'                    BNE     ZF68D >---/                     // if (z) == 0 Schleife 7 Runden
*F693: 54             'T'                     LSRB                                    // 0 -> B7 ... B0 -> c
*F694: 79 00 44       'y.D'                   ROL     >M0044                          // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c; 1bit von B nach (0x0044) gespiegelt
*F697: 3E             '>'                     WAI                                     // Wait for Interrupt
*F698: 0D             '.'                     SEC                                     // Set carry
*F699: 79 00 44       'y.D'                   ROL     >M0044                          // Rotiere (0x0044); c <- (0x0044)7 ... (0x0044)0 <- c; das zweite Bit = 1
*F69C: 3E             '>'                     WAI                                     // Wait for Interrupt
*F69D: DC 0B          '..'                    LDD     M000B                           // D = (0x000b); OutputCompareHigh Status TCSR1
*F69F: C3 03 41       '..A'                   ADDD    #M0341                          // D += 0x0341
*F6A2: DD 0B          '..'                    STD     M000B                           // (0x000b) = D; OutputCompareHigh Status TCSR1
*F6A4: 39             '9'                     RTS                                     // RETURN

// Call Init Serielle und Timer
*F6A5: CE 03 41       '..A'           ZF6A5   LDX     #M0341                          // IX = 0x0341
*F6A8: DF 5A          '.Z'                    STX     M005A                           // (0x005a) = IX
*F6AA: DF 0B          '..'                    STX     M000B                           // (0x000b) = IX;  OutputCompareHigh Status TCSR1
```

```
*F6AC: 86 08          '..'                    LDAA    #$08                            // A = 0x08; Enable Output Interrupt 1 (OCI1) to trigger internal IRQ3
*F6AE: 97 08          '..'                    STAA    M0008                           // (0x0008) = A; Port21 Output, TCSR1 Timer Control Status 1
*F6B0: 0E             '.'                     CLI                                     // Clear Interrupt Mask = 0
*F6B1: 39             '9'                     RTS                                     // RETURN

// Call Sprungverteiler 16
*F6B2: 7F 00 08       '...'                   CLR     >M0008                          // (0x0008) = 0x00; Port21 Output, TCSR1 Timer Control Status 1
*F6B5: 0E             '.'                     CLI                                     // Clear Interrupt Mask = 0
*F6B6: 96 2F          './'                    LDAA    M002F                           // A = (0x002f); Text-Nr. 0x1 ... 0x62
*F6B8: 36             '6'                     PSHA                                    // Push A
*F6B9: 7F 00 2F       '../'                   CLR     >M002F                          // (0x002f) = 0x00; Text-Nr. 0x1 ... 0x62
*F6BC: 7C 00 2F       '|./'           ZF6BC   INC     >M002F <----\                   // (0x002f)++; Text-Nr. 0x1 ... 0x62
*F6BF: 96 2F          './'                    LDAA    M002F       |                   // A = (0x002f); Text-Nr. 0x1 ... 0x62
*F6C1: 81 63          '.c'                    CMPA    #$63        |                   // CMP A, 0x63
*F6C3: 22 26          '"&'                    BHI     ZF6EB       | >--\              // if (c) & (z) == 0
*F6C5: BD E2 0D       '...'                   JSR     ZE20D       |    |              // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*F6C8: 27 F2          ''.'                    BEQ     ZF6BC >-----+    |              // if (z) == 1
*F6CA: BD E2 1A       '...'                   JSR     ZE21A       |    |              // Call  0xe21a  Init Textlänge Zeiger
*F6CD: BD EA DF       '...'                   JSR     ZEADF       |    |              // Call 0xeadf;  IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*F6D0: 27 06          ''.'                    BEQ     ZF6D8 >----\|    |              // if (z) == 1
*F6D2: BD EC 74       '..t'                   JSR     ZEC74      ||    |              // Call 0xec74; Rückgabe  A = (0x0030) bzw. (0x0031)
*F6D5: BD E5 BC       '...'                   JSR     ZE5BC      ||    |              // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*F6D8: BD ED 82       '...'           ZF6D8   JSR     ZED82 <----/|    |              // Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
*F6DB: BD ED F6       '...'                   JSR     ZEDF6       |    |              // Call 0xedf6; Textnr. Chiffriert
*F6DE: BD F6 46       '..F'                   JSR     ZF646       |    |              // Call 0xf646; Serielle Daten lesen
*F6E1: BD EA 94       '...'                   JSR     ZEA94       |    |              // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*F6E4: C6 32          '.2'                    LDAB    #$32        |    |              // B = 0x32
*F6E6: BD EA CF       '...'                   JSR     ZEACF       |    |              // Call 0xeacf Zeitschleife 10ms; 500ms
*F6E9: 20 D1          ' .'                    BRA     ZF6BC   >---/    |              // JR 0xf6bc
                                                                       |
*F6EB: 32             '2'             ZF6EB   PULA                <---/              // Pop A
*F6EC: 97 2F          './'                    STAA    M002F                          // (0x002f) = A; Zählervariable 0x1 ... 0x62
*F6EE: 0F             '.'                     SEI                                    // Set Interrupt Mask / NMI
*F6EF: 86 04          '..'                    LDAA    #$04                           // A = 0x04
*F6F1: 97 08          '..'                    STAA    M0008                          // (0x0008) = A; Enable Timer Interrupt IRQ3, Port21 Output, TCSR1 Timer Control Status 1
*F6F3: 7E E2 1A       '~..'                   JMP     ZE21A                          // JP 0xe21a mit Return Init Textlänge Zeiger

// CPU TOF Timer Overflow; Tastatur & LCD Timer
*F6F6: DC 08          '..'          hdlr_SWI3 LDD     M0008                          // D = (0x0008); Lese TCSR1 Timer Control Status 1
*F6F8: 7A 00 56       'z.V'                   DEC     >M0056                         // (0x0056)--
*F6FB: 2E 45          '.E'                    BGT     ZF742         >---\            // if ((Z) & (v|n)) == 0 Interrupt-Routine dritter Teil
*F6FD: 86 05          '..'                    LDAA    #$05              |            // A = 0x05
*F6FF: 97 56          '.V'                    STAA    M0056             |            // (0x0056) = A
*F701: 86 D5          '..'                    LDAA    #$D5              |            // A = 0xd5; LCD Kopfzeile BATT
*F703: BD EC 37       '..7'                   JSR     ZEC37             |            // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*F706: 86 20          '. '                    LDAA    #$20              |            // A = 0x20; Ausgabe " " auf LCD
*F708: BD EC 52       '..R'                   JSR     ZEC52             |            // Call 0xec52; LCD Data nach Busy; Übergabe A
*F70B: 7D 00 32       '}.2'                   TST     >M0032            |            // (0x0032) == 0; bit7 == 1
*F70E: 2B 32          '+2'                    BNI     ZF742         >---+            // if (n) == 1 Interrupt-Routine dritter Teil;
*F710: 7D 00 57       '}.W'                   TST     >M0057            |            // (0x0057) == 0?;  MSB == 1 n-Flag (0x0057) == 0 z-Flag
*F713: 26 0C          '&.'                    BNE     ZF721     >---\   |            // if (z) == 0 Interrupt-Routine zweiter Teil
*F715: CE 00 CD       '...'                   LDX     #M00CD        |   |            // IX = 0x00cd
*F718: D6 32          '.2'                    LDAB    M0032         |   |            // B = (0x0032)
*F71A: C4 7F          '..'                    ANDB    #$7F          |   |            // B &= 0x7f Maskiere
*F71C: 3A             ':'                     ABX                   |   |            // IX += B
*F71D: E6 00          '..'                    LDAB    ,X            |   |            // B = (IX); IX = 0x00cd ... 0x01fc
*F71F: 20 13          ' .'                    BRA     ZF734         |   | >---\      // JR Interrupt-routine zweiter Teil
                                                                    |   |     |
// Interrupt-Routine zweiter Teil                                   |   |     |
*F721: 7B 01 02       '{..'           ZF721   TIM     #$01,M0002 <--/   |     |      // (Port-1 Data) &= 0x01 Check U min?
*F724: 26 05          '&.'                    BNE     ZF72B >----\      |     |      // if (z) == 0 Test Bit0 Port-10 U.min
*F726: 86 D5          '..'                    LDAA    #$D5       |      |     |      // A = 0xd5
*F728: BD EE C5       '...'                   JSR     ZEEC5      |      |     |      // Call 0xeec5; LCD Data mit Busy; LCD Kopfzeile BATT
*F72B: C6 01          '..'            ZF72B   LDAB    #$01  <----/      |     |      // B = 0x01
*F72D: 7D 00 37       '}.7'                   TST     >M0037            |     |      // (0x0037) == 0?;  MSB == 1 n-Flag (0x0037) == 0 z-Flag
*F730: 26 02          '&.'            ZF730   BNE     ZF734 >----\      |     |      // if (z) == 0
*F732: C6 00          '..'                    LDAB    #$00       |      |     |      // B = 0x00
*F734: 96 32          '.2'            ZF734   LDAA    M0032 <----/      | <---/      // A = (0x0032)
*F736: 8A 80          '..'                    ORAA    #$80              |            // A |= 0x80; DD-RAM LCD
*F738: BD EC 37       '..7'                   JSR     ZEC37             |            // Call 0xec37; Warte auf Busy LCD-Controller, Übergabe A -> LCD Port COMMAND
*F73B: 17             '.'                     TBA                       |            // A = B; 0 oder 1; Sonderzeichen ▮ oder []
*F73C: BD EC 52       '..R'                   JSR     ZEC52             |            // Call 0xec52; LCD Data nach Busy; Übergabe A
*F73F: 73 00 57       's.W'                   COM     >M0057            |            // Complement (0x0057)
*F742: DE 58          '.X'            ZF742   LDX     M0058         <---/            // IX = (0x0058); sollte auf 0x02fb zeigen
*F744: 09             '.'                     DEX                                    // IX--; 0x02fa ...
*F745: DF 58          '.X'                    STX     M0058                          // (0x0058) = IX; Sichere Zeiger auf ...
*F747: 27 01          ''.'                    BEQ     ZF74A     >---\                // if (z) == 1; TimerCSR auf 0, Warte auf Interrupt
*F749: 3B             ';'           hdlr NMI  RTI                   |              // RETI (Inclusive Pop über alle Register)
                                                                    |
// TimerCSR auf 0, Warte auf Interrupt                              |
*F74A: 7F 00 08       '...'           ZF74A   CLR     >M0008   <---/               // (0x0008) = 0x00;Port21 Output, TCSR1 Timer Control Status 1
*F74D: 71 F7 02       'q..'                   AIM     #$F7,M0002                   // (Port-1 Data) &= 0xf7; außer bit3 Port-13 Keyboard
*F750: 71 FB 03       'q..'                   AIM     #$FB,M0003                   // (Port-2 Data) &= 0xfb; außer bit2 Port-22 unbekannt
*F753: 8E 02 87       '...'           ZF753   LDS     #M0287 <---\                 // SP = 0x0287
*F756: 3E             '>'                     WAI                |                 // Wait for Interrupt; über TimerCSR
*F757: 20 FA          ' .'                    BRA     ZF753  >---/                 // JR 0xf753; Schleife Warte auf Interrupt

// Call Adresse 0x0058 ist Zeiger auf 0x02fb
*F759: 3C             '<'             ZF759   PSHX                                 // Push IX
*F75A: CE 02 FB       '...'                   LDX     #M02FB                       // IX = 0x02fb
*F75D: DF 58          '.X'                    STX     M0058                        // (0x0058) = IX; Zeiger auf 0x02fb
*F75F: 38             '8'                     PULX                                 // Pop IX
*F760: 39             '9'                     RTS                                  // RETURN

// Definiere (0x0056, 0x0057) = 0x01ff RAM-Ende;  RUFT WER AUF??
```

```
*F761: 86 01          '..'                    LDAA    #$01                    // A = 0x01
*F763: 97 56          '.V'                    STAA    M0056                   // (0x0056) = A
*F765: 86 FF          '..'                    LDAA    #$FF                    // A = 0xff
*F767: 97 57          '.W'                    STAA    M0057                   // (0x0057) = A
*F769: 39             '9'                     RTS                             // RETURN

// MF76A
*F76A: 3C 20 46 52 45 45 20 BE                                                // "< FREE >"
// MF772
*F772: 2A 2A 2A 20 57 52 4F 4E 47 20 4B 45 59 20 2A 2A                        // "*** WRONG KEY ***  - PLEASE WAIT -"
*F782: 2A 20 20 2D 20 50 4C 45 41 53 45 20 57 41 49 54
*F792: 20 AD
// MF794
*F794: 45 4E 2F 44 45 43 52 59 50 54 20 54 45 58 54 20                        // "EN/DECRYPT TEXT .. ?"
*F7A4: 2E 2E 20 3F A0
// MF7A9
*F7A9: 45 4E 43 52 59 50 54 45 44 20 54 45 58 54 2C 20                        // "ENCRYPTED TEXT, LENGTH /:99 BYTE"
*F7B9: 4C 45 4E 47 54 48 20 2F 3A 39 39 20 42 59 54 45
*F7C9: D3
// MF7CA
*F7CA: 4E 45 57 20 4B 45 59 3A 20 2E 2E 2E 2E 2E 2E 2E                        // "NEW" KEY: ..............."
*F7DA: 2E 2E 2E 2E 2E 2E 2E 2E 2E A0
// MF7E4
*F7E4: 3C 20 4E 45 57 20 4B 45 59 20 41 43 43 45 50 54                        // "< NEW KEY ACCEPTED >"
*F7F4: 45 44 20 BE
// MF7F8
*F7F8: 4E 45 57 20 4B 45 59 3A 20 54 45 58 54 20 30 30                        // "NEW KEY: TEXT 00 "
*F808: A0
//MF809
*F809: 4D 45 4D 4F 52 59 20 4F CB                                             // "MEMORY OK"
// MF812
*F812: 4D 45 4D 4F 52 59 20 45 52 52 4F D2                                    // "MEMORY ERROR"
//MF81E
*F81E: 3C 20 2E 2E 2E 2E 2E 2E 2E 2E 20 4E 4F 54 20 46                        // "< ........ NOT FOUND >
*F82E: 4F 55 4E 44 20 BE
MF834
*F834: 43 4F 50 59 52 49 47 48 54 20 31 39 38 34 20 57                        // "COPYRIGHT 1984 WEST-TEC PX V2"
*F844: 45 53 54 2D 54 45 43 20 50 58 20 56 B2
// MF851
*F851: 53 45 41 52 43 48 20 46 4F 52 20 2E 2E 2E 2E 2E                        // "SEARCH FOR ........ "
*F861: 2E 2E 2E A0
//MF865
*F865: 53 54 41 52 54 20 27 52 45 43 4F 52 44 27 20 4F                        // "START 'RECORD' ON TAPE "
*F875: 4E 20 54 41 50 45 A0
// MF87B
*F87C: 2A 2A 2A 20 42 41 44 20 54 45 58 54 20 2A 2A AA                        // "*** BAD TEXT ***"
// MF88C
*F88C: 43 4F 4E 56 45 52 53 49 4F 4E 20 43 48 41 52 41                        // "CONVERSION CHARACTER . -> 00  + RETURN"
*F89C: 43 54 45 52 20 2E 20 2D 3E 20 30 30 20 20 2B 20
*F8AC: 52 45 54 55 52 CE
// MF8B2
*F8B2: 2A 2A 2A 20 4D 45 4D 4F 52 59 20 46 55 4C 4C 20                        // "*** MEMORY FULL ***"
*F8C2: 2A 2A AA
// MF8C5
*F8C5: 2A 2A 2A 20 45 52 52 4F 52 20 49 4E 20 28 28 28                        // "*** ERROR IN ((())) ***"
*F8D5: 29 29 29 20 2A 2A AA
MF8DC
*F8DC: 43 4C 45 41 52 20 41 4C 4C 20 54 45 58 54 20 3F                        // "CLEAR ALL TEXT ? "
*F8EC: A0
//MF8ED
*F8ED: 2A 2A 2A 20 4E 4F 20 46 52 45 45 20 54 45 58 54                        // "*** NO FREE Text AVAIABLE ***"
*F8FD: 20 41 56 41 49 4C 41 42 4C 45 20 2A 2A AA
// MF80B
*F90B: 2A 2A 2A 20 43 41 4E 20 4E 4F 54 20 43 41 4C 43                        // "*** CAN NOT CALCULATE ***
*F91B: 55 4C 41 54 45 20 2A 2A AA
// MF924
*F924: 49 4E 53 45 52 54 20 54 45 58 54 20 30 30 20 3F                        // "INSERT TEXT 00 ? "
*F934: A0
// MF935
*F935: 44 45 4C 45 54 45 20 54 45 58 54 20 2E 2E 20 3F                        // "DELETE TEXT  .. ? "
*F945: A0
// MF946
*F946: 52 45 41 44 59 20 54 4F 20 52 45 43 45 49 56 45                        // "READY TO RECEIVE      (/:999"
*F956: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
*F966: 20 20 28 2F 3A 39 39 A9
// MF96E
*F96E: 53 45 54 20 52 49 47 48 54 20 4D 41 52 47 49 4E                        // "SET RIGHT MARGIN AT .. "
*F97E: 20 41 54 20 2E 2E A0
// MF985
*F985: 52 45 41 44 59 20 54 4F 20 53 45 4E 44 20 5B 2E                        // "READY TO SEND [.] - PRESS SENDKEY (/:999"
*F995: 5D 20 2D 20 50 52 45 53 53 20 53 45 4E 44 4B 45
*F9A5: 59 20 28 2F 3A 39 39 A9
// MF9AD
*F9AD: 3C 20 54 52 41 4E 53 4D 49 54 54 49 4E 47 20 BE                        // "< TRANSMITTING >"
// MF9BD
*F9BD: 2B 20 50 52 45 53 53 20 41 47 41 49 CE                                 // "+ PRESS AGAIN"

// Data LCD Initialisierung
*F9CA: 30 30 30 20 20 80 00 C0 00 10 00 60            MF9CA                   // Spezifikation LCD Datenblatt 4bit: 0x30,0x30,0x30,0x20,0x20,0x80,

// LCD 8 Sonderzeichen 5x7; Adresse 0 ... 7
*F9D6: 1F 1F 1F 1F 1F 1F 1F 1F                        MF9D6                    // ▮
*F9DE: 00 1B 11 00 00 00 11 1B                                                // [] als ein Symbol
*F9E6: 00 0E 00 0E 11 1F 11 11                                                // Ā
*F9EE: 00 0E 00 0E 01 0F 11 0F                                                // ā
```

```
*F9F6: 00 00 00 00 00 00 00 1F                                                  //   (kein Cursor, der liegt darunter)
*F9FE: 00 02 00 00 00 00 00 00                                                  // '                 // '
*FA06: 00 00 00 00 00 00 04 1F                                                  // _._  als ein Symbol
*FA0E: 00 04 04 04 04 04 04 04                                                  // |

// Call A = (!A-low << 4) & A-low, Übergabe A Rückgabe A
*FA16: 97 5A          '.Z'              ZFA16   STAA    M005A                   // (0x005a) = A
*FA18: 43             'C'                       COMA                          // Complement A
*FA19: 48             'H'                       ASLA                          // c <- A7 ... A0 <- 0 Shift left
*FA1A: 48             'H'                       ASLA                          // c <- A7 ... A0 <- 0 Shift left
*FA1B: 48             'H'                       ASLA                          // c <- A7 ... A0 <- 0 Shift left
*FA1C: 48             'H'                       ASLA                          // c <- A7 ... A0 <- 0 Shift left
*FA1D: 9A 5A          '.Z'                      ORAA    M005A                 // A |= (0x005a)
*FA1F: 39             '9'                       RTS                           // RETURN

// Call Sprungverteiler 28 EN/DECRYPTION
*FA20: BD E2 0D       '...'                     JSR     ZE20D                 // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*FA23: 26 03          '&.'                      BNE     ZFA28 >---\           // if (z) == 0
*FA25: 7E EA 94       '~..'                     JMP     ZEA94     |           // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
// JP                                                               |
*FA28: CE F7 94       '...'             ZFA28   LDX     #MF794 <--/           // IX = 0xf794; "EN/DECRYPTE TEXT ..?
*FA2B: C6 10          '..'                      LDAB    #$10                  // B = 0x10
*FA2D: 96 2F          './'                      LDAA    M002F                 // A = (0x002f); Text Adresse 0x03 ... 0x62; D = 0x0310, ... 0x6210
*FA2F: BD EC CF       '...'                     JSR     ZECCF                 // Call 0xeccf; Übergabe D = 0x0310 ... 0x6310, IX AusgabeText; Rückgabe D (0x0066)
*FA32: BD F3 F9       '...'                     JSR     ZF3F9                 // Call 0xf3f9; Tastendruck; Rückgabe A
*FA35: 81 AB          '..'                      CMPA    #$AB                  // CMP A, 0xab; Taste "CODE"
*FA37: 27 01          '.'                       BEQ     ZFA3A >---\           // if (z) == 1 Ausgabe: Please Wait und De- Chiffrieren
*FA39: 39             '9'                       RTS               |           // RETURN
                                                                   |
// Ausgabe Please Wait - De- Chiffrieren                           |
*FA3A: CE F7 83       '...'             ZFA3A   LDX     #MF783 <--/           // IX = 0xf783; "  - PLEASE WAIT - "
*FA3D: BD EC EA       '...'                     JSR     ZECEA                 // Call 0xecea; Textausgabe LCD, Übergabe IX
// JP Wrong-KEY
*FA40: BD E2 12       '...'             ZFA40   JSR     ZE212                 // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*FA43: A6 00          '..'                      LDAA    ,X                    // A = (IX)
*FA45: 2B 16          '+.'                      BNI     ZFA5D  >----\         // if (n) == 1; bit7 == 0 Chiffrieren, Übergabe IX
*FA47: 3C             '<'                       PSHX                |         // Push IX
*FA48: BD FE 21       '..!'                     JSR     ZFE21       |         // Call 0xfe21 Abfrage IX (0x0025) <= (0x0027); while (0x0025 <= 0x0027) ix++ 0x0025 erstes ASCII im Speicher TestIfStringIsHex
*FA4B: 38             '8'                       PULX                |         // Pop IX
*FA4C: 25 0D          '%.'                      BCS     ZFA5B >---\ |         // if (c) == 1; Dechiffrieren Übergabe IX
*FA4E: CE FC 6C       '..l'                     LDX     #MFC6C    | |         // IX = 0xfc6c; "** PX WILL NOT ENCRYPT (HEX) NUMBERS **"      /*
*FA51: BD EC C5       '...'                     JSR     ZECC5     | |         // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*FA54: BD EA CD       '...'                     JSR     ZEACD     | |         // Call 0xeacd; Zeitschleife * 0x64;
*FA57: BD EA CD       '...'                     JSR     ZEACD     | |         // Call 0xeacd; Zeitschleife * 0x64;
*FA5A: 39             '9'                       RTS               | |         // RETURN
                                                                  | |
//Übergabe IX Passwort-Verarbeitung                               | |
*FA5B: A6 00                            ZFA5B   LDAA    ,X   <----/ |         // A = (IX)
*FA5D: 88 80                            ZFA5D   EORA    #$80  <-----/         // A XOR= 0x80; Bit 7 löschen/setzen
*FA5F: A7 00                                    STAA    ,X                    // (IX) = A
*FA61: 97 BE                                    STAA    M00BE                 // (0x00be) = A; Spätere Abfrage Dechiffrieren/Chiffrieren
*FA63: C6 04                                    LDAB    #$04                  // B = 0x04; Schleifenzähler
*FA65: CE 01 50                         ZFA65   LDX     #M0150 <--\           // IX = 0x0150; Bearbeiter Schlüssel in 0xFC93
*FA68: 3A                                       ABX               |           // IX += B; B = 4, 3, 2, 1
*FA69: A6 00                                    LDAA    ,X        |           // A = (IX); 0x0154, 0x0153, 0x0152, 0x0151
*FA6B: A8 04                                    EORA    $04,X     |           // A XOR= (IX + 0x04); 0x0158, 0x0157, 0x0156, 0x0155
*FA6D: 8D A7                                    BSR     ZFA16     |           // Call 0xfa16; A = (!A-low << 4) & A-low, Übergabe A Rückgabe A
*FA6F: CE 00 74                                 LDX     #M0074    |           // IX = 0x0074
*FA72: 3A                                       ABX               |           // IX += B; 0x0078, 0x0077, 0x0076, 0x0075
*FA73: A7 00                                    STAA    ,X        |           // (IX) = A; XOR werte in 0x0075 bis 0x0078 ablegen
*FA75: 5A                                       DECB              |           // B--
*FA76: 26 ED                                    BNE     ZFA65 >---/           // if (z) == 0 Schleife 4 Runden
*FA78: C6 04                                    LDAB    #$04                  // B = 0x04; Schleifenzähler
*FA7A: CE 01 58                         ZFA7A   LDX     #M0158 <--\           // IX = 0x0158
*FA7D: 3A                                       ABX               |           // IX += B; 0x015c, 0x015b, 0x015a, 0x0159
*FA7E: A6 00                                    LDAA    ,X        |           // A = (IX)
*FA80: A8 04                                    EORA    $04,X     |           // A XOR= (IX + 0x04); 0x0160, 0x015f, 0x015e, 0x015d
*FA82: 8D 92                                    BSR     ZFA16     |           // Call 0xfa16; A = (!A-low << 4) & A-low, Übergabe A Rückgabe A
*FA84: CE 00 78                                 LDX     #M0078    |           // IX = 0x0078
*FA87: 3A                                       ABX               |           // IX += B
*FA88: A7 00                                    STAA    ,X        |           // (IX) = A; 0x007d, ... 0x0079
*FA8A: 5A                                       DECB              |           // B--
*FA8B: 26 ED                                    BNE     ZFA7A >---/           // if (z) == 0 Schleife 4 Runden
*FA8D: C6 04                                    LDAB    #$04                  // B = 0x04; Schleifenzähler
*FA8F: CE 00 75                                 LDX     #M0075                // IX = 0x0075
*FA92: A6 00                            ZFA92   LDAA    ,X     <--\           // A = (IX); 0x0075 ... 0x0078
*FA94: A8 04                                    EORA    $04,X     |           // A XOR= (IX + 0x04); 0x0079 ... 0x007c
*FA96: 88 F0                                    EORA    #$F0      |           // A XOR= 0xf0; invertiere High-Teil
*FA98: A7 08                                    STAA    $08,X     |           // (IX + 0x8) = A; 0x007d ... 0x0080
*FA9A: 08                                       INX               |           // IX++
*FA9B: 5A                                       DECB              |           // B--
*FA9C: 26 F4                                    BNE     ZFA92 >---/           // if (z) == 0 Schleife 4 Runden
*FA9E: C6 0F                                    LDAB    #$0F                  // B = 0x0f; Schleifenzähler 16 Runden
*FAA0: CE 01 50                         ZFAA0   LDX     #M0150 <--\           // IX = 0x0150; Klartext-Passwort
*FAA3: 3A                                       ABX               |           // IX += B; 0x015f .... 0x0151
*FAA4: A6 00                                    LDAA    ,X        |           // A = (IX)
*FAA6: BD FA 16                                 JSR     ZFA16     |           // Call 0xfa16; A = (!A-low << 4) & A-low, Übergabe A Rückgabe A
*FAA9: CE 00 89                                 LDX     #M0089    |           // IX = 0x0089; KeySpeicher
*FAAC: 3A                                       ABX               |           // IX += B; 0x0098 ... 0x008A
*FAAD: A7 00                                    STAA    ,X        |           // (IX) = A
*FAAF: 5A                                       DECB              |           // B--
*FAB0: 26 EE                                    BNE     ZFAA0 >---/           // if (z) == 0 Schleife 16 byte
*FAB2: 86 FF                                    LDAA    #$FF                  // A = 0xff
*FAB4: 97 99                                    STAA    M0099                 // (0x0099) = A
*FAB6: DE 25                                    LDX     M0025                 // IX = (0x0025); Zeiger Text-Start
```

```
*FAB8: 09                                DEX                                    // IX--
*FAB9: DF 2D                             STX     M002D                          // (0x002d) = IX; Klartext-Startadresse - Kompromat die Erste
// JP PRNG-Funktion
*FABB: C6 1F                    ZFABB    LDAB    #$1F              <---\        // B = 0x1f; Schleifenzähler 32 Runden
*FABD: D7 5C                             STAB    M005C                  |        // (0x005c) = 0x1f = B  Schleifenzähler
*FABF: CE FE DB                 ZFABF    LDX     #MFEDB <---\           |        // IX = 0xFEDB + 0x0, ... 0x1f; Index der Masken-Tabelle
*FAC2: D6 5C                             LDAB    M005C      |           |        // B = (0x005c)  Schleifenzähler
*FAC4: C4 0F                             ANDB    #$0F       |           |        // B &= 0x0f  ... 0x0
*FAC6: 3A                                ABX                |           |        // IX += B; B Schrittweite 0x0f ... 0x00
*FAC7: 96 8A                             LDAA    M008A      |           |        // A = (0x008a); gebildeter Schlüssel (aus 0x0150-0x015c)
*FAC9: A4 00                             ANDA    ,X         |           |        // A &= (IX); gebildeter Schlüssel mit Permutation AND Verknüpft
*FACB: 97 5A                             STAA    M005A      |           |        // (0x005a) = A; Wert1
*FACD: 96 8B                             LDAA    M008B      |           |        // A = (0x008b)
*FACF: A4 01                             ANDA    $01,X      |           |        // A &= (IX + 0x01); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAD1: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a); Wert2 XOR= Wert1
*FAD3: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FAD5: 96 8C                             LDAA    M008C      |           |        // A = (0x008c)
*FAD7: A4 02                             ANDA    $02,X      |           |        // A &= (IX + 0x02); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAD9: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert3 XOR= Wert2
*FADB: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FADD: 96 8D                             LDAA    M008D      |           |        // A = (0x008d)
*FADF: A4 03                             ANDA    $03,X      |           |        // A &= (IX + 0x03); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAE1: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert4 XOR= Wert3
*FAE3: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FAE5: 96 8E                             LDAA    M008E      |           |        // A = (0x008e)
*FAE7: A4 04                             ANDA    $04,X      |           |        // A &= (IX + 0x04); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAE9: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert5 XOR= Wert4
*FAEB: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FAED: 96 8F                             LDAA    M008F      |           |        // A = (0x008f)
*FAEF: A4 05                             ANDA    $05,X      |           |        // A &= (IX + 0x05); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAF1: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert6 XOR= Wert5
*FAF3: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FAF5: 96 90                             LDAA    M0090      |           |        // A = (0x0090)
*FAF7: A4 06                             ANDA    $06,X      |           |        // A &= (IX + 0x06); gebildeter Schlüssel mit Permutation AND Verknüpft
*FAF9: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert7 XOR= Wert6
*FAFB: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FAFD: 96 91                             LDAA    M0091      |           |        // A = (0x0091)
*FAFF: A4 07                             ANDA    $07,X      |           |        // A &= (IX + 0x07); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB01: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert8 XOR= Wert7
*FB03: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB05: 96 92                             LDAA    M0092      |           |        // A = (0x0092)
*FB07: A4 08                             ANDA    $08,X      |           |        // A &= (IX + 0x08); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB09: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert9 XOR= Wert8
*FB0B: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB0D: 96 93                             LDAA    M0093      |           |        // A = (0x0093)
*FB0F: A4 09                             ANDA    $09,X      |           |        // A &= (IX + 0x09); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB11: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert10 XOR= Wert9
*FB13: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB15: 96 94                             LDAA    M0094      |           |        // A = (0x0094)
*FB17: A4 0A                             ANDA    $0A,X      |           |        // A &= (IX + 0x0a); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB19: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert11 XOR= Wert10
*FB1B: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB1D: 96 95                             LDAA    M0095      |           |        // A = (0x0095)
*FB1F: A4 0B                             ANDA    $0B,X      |           |        // A &= (IX + 0x0b); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB21: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert12 XOR= Wert11
*FB23: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB25: 96 96                             LDAA    M0096      |           |        // A = (0x0096)
*FB27: A4 0C                             ANDA    $0C,X      |           |        // A &= (IX + 0x0c); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB29: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert13 XOR= Wert12
*FB2B: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB2D: 96 97                             LDAA    M0097      |           |        // A = (0x0097)
*FB2F: A4 0D                             ANDA    $0D,X      |           |        // A &= (IX + 0x0d); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB31: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert14 XOR= Wert13
*FB33: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB35: 96 98                             LDAA    M0098      |           |        // A = (0x0098)
*FB37: A4 0E                             ANDA    $0E,X      |           |        // A &= (IX + 0x0e); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB39: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert15 XOR= Wert14
*FB3B: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB3D: 96 99                             LDAA    M0099      |           |        // A = (0x0099); 0xff
*FB3F: A4 0F                             ANDA    $0F,X      |           |        // A &= (IX + 0x0f); gebildeter Schlüssel mit Permutation AND Verknüpft
*FB41: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert16 XOR= Wert15
*FB43: 97 5A                             STAA    M005A      |           |        // (0x005a) = A
*FB45: 44                                LSRA               |           |        // 0 -> A7 ... A0 -> c
*FB46: 98 5A                             EORA    M005A      |           |        // A XOR= (0x005a) Wert17 = Wert16 XOR (Wert16 >> 1)
*FB48: 84 55                             ANDA    #$55       |           |        // A &= 0x55 Maskiere
*FB4A: 97 5A                             STAA    M005A      |           |        // (0x005a) = A; Wert18 = Wert17 & 0x55
*FB4C: CE 00 8A                          LDX     #M008A     |           |        // IX = 0x008a; gebildeter Schlüssel
*FB4F: D6 5C                             LDAB    M005C      |           |        // B = (0x005c) Schleifenzähler 0x1f ... 0x0
*FB51: 53                                COMB               |           |        // Complement B; 0xe0 . 0xe1 .. 0xf0 . 0xf1 .. 0xff
*FB52: C4 0F                             ANDB    #$0F       |           |        // B &= 0x0f; 0x00 ... 0x0f
*FB54: 3A                                ABX                |           |        // IX += B; (0x008a) = Schleifenzähler 0 ... 15
*FB55: A6 00                             LDAA    ,X         |           |        // A = (IX); = gebildeter Schlüssel (0x008a ...)
*FB57: 48                                ASLA               |           |        // c <- A7 ... A0 <- 0
*FB58: 84 AA                             ANDA    #$AA       |           |        // A &= 0xAA Maskiere
*FB5A: 9A 5A                             ORAA    M005A      |           |        // A |= (0x005a); Wert18 |= ((0x008a) << 1) & 0xaa
*FB5C: A7 00                             STAA    ,X         |           |        // (IX) = A; gebildeter Schlüssel (0x008a + n)
*FB5E: 7A 00 5C                          DEC     >M005C     |           |        // (0x005c)-- Schleifenzähler 32 Runden
*FB61: 2B 03                             BNI     ZFB66      | >----\    |        // if (n) == 1; bit7 == 0; Chiffrx; 0x005c == -1
*FB63: 7E FA BF                          JMP     ZFABF >----/      |    |        // JP 0xfabf; Schleife bis Abbruch
                                                                   |    |
// Weiter mit Substitution                                         |    |
*FB66: C6 04                    ZFB66    LDAB    #$04   <----------/    |        // B = 0x04; Schleifenzähler
*FB68: CE 00 89                 ZFB68    LDX     #M0089 <--\            |        // IX = 0x0089; KeySpeicher
*FB6B: 3A                                ABX                |           |        // IX += B; 0x0089 + B; Schritte 4, 3, 2, 1
*FB6C: A6 07                             LDAA    $07,X      |           |        // A = (IX + 0x07) (0x0089 + B + 7); 0x0094 ... 0x0091 gebildeter Schlüssel
```

```
*FB6E: 48                                ASLA              |           |                    // c <- A7 ... A0 <- 0 Shift left
*FB6F: 89 00                             ADCA    #$00      |           |                    // A += carry; Add Übertrag
*FB71: 48                                ASLA              |           |                    // c <- A7 ... A0 <- 0 Shift left
*FB72: 89 00                             ADCA    #$00      |           |                    // A += carry; Add Übertrag
*FB74: A8 00                             EORA    ,X        |           |                    // A XOR= (IX) (0x008d ... 0x008a) gebildeter Schlüssel
*FB76: CE 00 81                          LDX     #M0081    |           |                    // IX = 0x0081; Ablage Ergebnis
*FB79: 3A                                ABX               |           |                    // IX += B; 0x0081 + B; 0x0085 ... 0x0082
*FB7A: A7 00                             STAA    ,X        |           |                    // (IX) = A; (0x0085 ... 0x0082) = A
*FB7C: CE 00 74                          LDX     #M0074    |           |                    // IX = 0x0074
*FB7F: 3A                                ABX               |           |                    // IX += B; 0x0074 + B; 0x0078 ... 0x0075
*FB80: A6 00                             LDAA    ,X        |           |                    // A = (IX) A = (0x0078 ... 0x0075); Ablage Werte aus 0xfa6f
*FB82: A8 08                             EORA    $08,X     |           |                    // A XOR= (0x0078 ... 0x0075 + 0x08); 0x0080 ... 0x007d; Werte aus 0xfa92
*FB84: 37                                PSHB              |           |                    // Push B; Rette Schleifenzähler
*FB85: CE FE 4B                          LDX     #MFE4B    |           |                    // IX = 0xfe4b; noch nicht der Data Bereich
*FB88: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB89: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB8A: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB8B: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left; 40, 30, 20, 10
*FB8C: 3A                                ABX               |           |                    // IX += B; fe4b + (40, 30, 20, 10)
*FB8D: 3C                                PSHX              |           |                    // Push IX; FE8B, FE7B, FE6B, FE5B: Permutationsreihen
*FB8E: 16                                TAB               |           |                    // B = A
*FB8F: 54                                LSRB              |           |                    // 0 -> B7 ... B0 -> c
*FB90: 54                                LSRB              |           |                    // 0 -> B7 ... B0 -> c
*FB91: 54                                LSRB              |           |                    // 0 -> B7 ... B0 -> c
*FB92: 54                                LSRB              |           |                    // 0 -> B7 ... B0 -> c; B-High->Low
*FB93: 3A                                ABX               |           |                    // IX += B; FE8B ... FE9A; ...
*FB94: E6 00                             LDAB    ,X        |           |                    // B = (IX); Inhalt der Permutationsreihe: FE8B, FE7B, FE6B, FE5B
*FB96: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB97: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB98: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left
*FB99: 58                                ASLB              |           |                    // c <- B7 ... B0 <- 0 Shift left; 4bit Low nach High
*FB9A: D7 5A                             STAB    M005A     |           |                    // (0x005a) = B; Sichere
*FB9C: 16                                TAB               |           |                    // B = A
*FB9D: C4 0F                             ANDB    #$0F      |           |                    // B &= 0x0f; Low-Wert aus 0x0080(+n) XOR 0x0078(+n)
*FB9F: 38                                PULX              |           |                    // Pop IX; FE8B, FE7B ...
*FBA0: 3A                                ABX               |           |                    // IX += B; FE8B + n (0x0 ...0xf)
*FBA1: A6 00                             LDAA    ,X        |           |                    // A = (IX); hole Permutationswert
*FBA3: 9A 5A                             ORAA    M005A     |           |                    // A |= (0x005a);  Permutationswert 1 OR 2
*FBA5: 33                                PULB              |           |                    // Pop B; Wiederherstellen Schleifenzähler
*FBA6: CE 00 78                          LDX     #M0078    |           |                    // IX = 0x0078
*FBA9: 3A                                ABX               |           |                    // IX += B; 0x0078 + n (4 ... 1)
*FBAA: A8 00                             EORA    ,X        |           |                    // A XOR= (IX); Perm 3 xor (1 or 2)
*FBAC: CE 00 85                          LDX     #M0085    |           |                    // IX = 0x0085
*FBAF: 3A                                ABX               |           |                    // IX += B; 0x0085 + n (4 ... 1)
*FBB0: A7 00                             STAA    ,X        |           |                    // (IX) = A; Ablegen 4bit Perm-3
*FBB2: 5A                                DECB              |           |                    // B--
*FBB3: 26 B3                             BNE     ZFB68 >---/           |                    // if (z) == 0; Schleife 4 Runden, Rechnen mit Permutationsarray
*FBB5: C6 08                             LDAB    #$08                  |                    // B = 0x08
*FBB7: 36                        ZFBB7   PSHA          <--------\       |                    // Push A; Rette Ergebnis
*FBB8: 37                                PSHB                   |       |                    // Push B; Rette Schleifenzähler
*FBB9: 86 03                             LDAA    #$03           |       |                    // A = 0x03; Schleifenzähler
*FBBB: CE 00 83                          LDX     #M0083         |       |                    // IX = 0x0083
*FBBE: 5F                                CLRB                   |       |                    // B = 0x00
*FBBF: 68 00                     ZFBBF   ASL     ,X    <----\   |       |                    // c <- IX7 ... IX0 <- 0 Shift left
*FBC1: 59                                ROLB              |   |       |                    // Rotiere B; c <- B7 ... B0 <- c ; einsammel High von IX über carry
*FBC2: 68 04                             ASL     $04,X     |   |       |                    // c <- IX7 ... IX0 <- 0 Shift left; IX + 4 = 0x0087 ... 0x0089
*FBC4: 59                                ROLB              |   |       |                    // Rotiere B; c <- B7 ... B0 <- c; Schiebe carry nach B, aus (0x0087 ...)
*FBC5: 08                                INX               |   |       |                    // IX++; 0x0083 ... 0x0085
*FBC6: 4A                                DECA              |   |       |                    // A--; Schleife A
*FBC7: 26 F6                             BNE     ZFBBF >---/   |       |                    // if (z) == 0; 3 Runden
                                                               |       |
// Substitution                                                |       |
*FBC9: CE FE 9B       '...'              LDX     #MFE9B        |       |                    // IX = 0xfe9b; Index-Tabelle
*FBCC: 3A             ':'                ABX                   |       |                    // IX += B; 0xfe9b + 6 bit aus B; bit 0 ... 5
*FBCD: A6 00          '..'               LDAA    ,X            |       |                    // A = (IX); Substitution
*FBCF: 33             '3'                PULB                  |       |                    // Pop B; wiederherstellen Schleifenzähler
*FBD0: 37             '7'                PSHB                  |       |                    // Push B; Rette Schleifenzähler
*FBD1: 44             'D'        ZFBD1   LSRA          <----\   |       |                    // 0 -> A7 ... A0 -> c Lösche A, bit 7 ... 0 nach carry
*FBD2: 5A             'Z'                DECB                |   |       |                    // B--; Schleife B
*FBD3: 26 FC          '&.'               BNE     ZFBD1 >---/   |       |                    // if (z) == 0 Schleife 8 ... 1-Runden
                                                               |       |                    // extrahiert carry aus der Index-Tabelle die Bits 7, 6, ... 0
// Ausgangspermutation                                         |       |
*FBD5: 33             '3'                PULB                  |       |                    // Pop B; wiederherstellen Schleifenzähler 8 ... 1
*FBD6: 32             '2'                PULA                  |       |                    // Pop A; wiederherstellen Ergebnis == 0x0086
*FBD7: 49             'I'                ROLA                  |       |                    // Rotiere A; c <- A7 ... A0 <- c; Bittausch via carry Schleife
*FBD8: 5A             'Z'                DECB                  |       |                    // B--
*FBD9: 26 DC          '&.'               BNE     ZFBB7 >-------/       |                    // if (z) == 0 Schleife 8 Runden
*FBDB: 98 82          '..'               EORA    M0082                 |                    // A XOR= (0x0082); XOR die erste
*FBDD: 98 86          '..'               EORA    M0086                 |                    // A XOR= (0x0086); (0x79 xor 0x5a); XOR die zweite
*FBDF: 36             '6'                PSHA                          |                    // Push A
*FBE0: DC 2D          '.-'               LDD     M002D                 |                    // D = (0x002d); Klartext-Start-neu; 0xfab6
*FBE2: 93 25          '.%'               SUBD    M0025                 |                    // D -= (0x0025); Ergebnis -1
*FBE4: 32             '2'                PULA                          |                    // Pop A; überschreibe High in D mit A;
*FBE5: 5C             '\'                INCB                          |                    // B++; aus D; Startadresse Klartext geradegebogen
*FBE6: C4 07          '..'               ANDB    #$07                  |                    // B &= 0x07, Schleifenzähler zwischen 0 und 7
/* Die Schleife wird betreten ab B > 0, und entsprechend Zeichenzahl 1 ... 7                  |                    // durchlaufen
*FBE8: 27 06          ''.'               BEQ     ZFBF0    >----\       |                    // if (z) == 1; mod(8)
*FBEA: 48             'H'        ZFBEA   ASLA          <---\   |       |                    // c <- A7 ... A0 <- 0 Shift left; wenn nicht -1 dann wird ab 0x0200 chiffriert
*FBEB: 89 00          '..'               ADCA    #$00      |   |       |                    // A += carry, Lade Übertrag nach bit 0
*FBED: 5A             'Z'                DECB              |   |       |                    // B--
*FBEE: 26 FA          '&.'               BNE     ZFBEA >---/   |       |                    // if (z) == 0 rotiere A max 7 Runden
// De- Chiffrierung via CFB                                    |       |
*FBF0: DE 2D          '.-'       ZFBF0   LDX     M002D    <----/       |                    // IX = (0x002d) Klartext; Zeiger Text-Start
*FBF2: E6 00          '..'               LDAB    ,X                    |                    // B = (IX)
*FBF4: A8 00          '..'               EORA    ,X                    |                    // A XOR= (IX); A key; A XOR= KTXT
```

```
*FBF6: A7 00          '..'                    STAA    ,X                      |                       // (IX) = A; Ablage chiffriert
*FBF8: 08             '.'                     INX                             |                       // IX++
*FBF9: DF 2D          '.-'                    STX     M002D                   |                       // (0x002d) = IX Nächstes Zeichen
*FBFB: 9C 27          '.''                    CPX     M0027                   |                       // cmp IX, (0x0027) Textende?
*FBFD: 22 1B          '".'                    BHI     ZFC1A     >----\        |                       // if (c) & (z) == 0 Textende
*FBFF: 7D 00 BE       '}..'                   TST     >M00BE          |       |                       // (0x00BE) == 0; bit7 == 1
*FC02: 2B 01          '+.'                    BNI     ZFC05  >----\   |       |                       // if (n) == 1 Lade A mit B; A <- Klartext
*FC04: 17             '.'                     TBA                  |  |       |                       // A = B; Klartext/GTX in CFB einspeisen
*FC05: 97 81          '..'            ZFC05   STAA    M0081  <----/  |       |                       // (0x0081) = A
*FC07: C6 04          '..'                    LDAB    #$04           |       |                       // B = 0x04
*FC09: CE 00 7D       '..}'                   LDX     #M007D         |       |                       // IX = 0x007d
*FC0C: A6 01          '..'            ZFC0C   LDAA    $01,X <---\    |       |                       // A = (IX + 0x01); 0x007e ... 0x0081
*FC0E: 48             'H'                     ASLA                |    |       |                       // c <- A7 ... A0 <- 0 Shift left
*FC0F: 89 00          '..'                    ADCA    #$00        |    |       |                       // A += carry; rotiere A
*FC11: A7 00          '..'                    STAA    ,X          |    |       |                       // (IX) = A; 0x007d ... 0x0080
*FC13: 08             '.'                     INX                 |    |       |                       // IX++; 0x007E ...0x0081
*FC14: 5A             'Z'                     DECB                |    |       |                       // B--
*FC15: 26 F5          '&.'                    BNE     ZFC0C >---/    |       |                       // if (z) == 0 Schleife 4 Runden
*FC17: 7E FA BB       '~..'                   JMP     ZFABB          |   >---/                       // JP 0xfabb, weiterer Aufruf PRNG
// Weiter                                                            |
*FC1A: 7D 00 BE       '}..'           ZFC1A   TST     >M00BE    <----/                              // (0x00BE) == 0; bit7 == 1, 0xff = Chiffrieren
*FC1D: 2B 42          '+B'                    BNI     ZFC61            >----\                       // if (n) == 1
*FC1F: DE 25          '.%'                    LDX     M0025                 |                       // IX = (0x0025); Zeiger Text-Ende
*FC21: 4F             'O'                     CLRA                          |                       // A = 0x00
*FC22: 5F             '_'                     CLRB                          |                       // B = 0x00; D = 0x0000
*FC23: 6D 00          'm.'            ZFC23   TST     ,X    <---\           |                       // (IX) == 0; bit7 == 0
*FC25: 2A 03          '*.'                    BPL     ZFC2A     | >--\      |                       // if (n) == 0
*FC27: C3 00 01       '...'                   ADDD    #M0001    |     |     |                       // D += 0x0001; Zähler Spruchnr.
*FC2A: 08             '.'             ZFC2A   INX               | <--/      |                       // IX++
*FC2B: 9C 27          '.''                    CPX     M0027     |           |                       // cmp IX; (0x0027); Zeiger Text-aktual Position
*FC2D: 23 F4          '#.'                    BLS     ZFC23 >---/           |                       // if (c) | (z) == 1; Schleife 0x0025 ... 0x0026
*FC2F: 05             '.'                     ASLD                          |                       // c <- A7 ... A0 <- B7 ... B0 <- 0
*FC30: 05             '.'                     ASLD                          |                       // c <- A7 ... A0 <- B7 ... B0 <- 0; D *= 4
*FC31: DD 5A          '.Z'                    STD     M005A                 |                       // (0x005a) = D
*FC33: DC 27          '.''                    LDD     M0027                 |                       // D = (0x0027); Zeiger Text-aktual Position
*FC35: 93 25          '.%'                    SUBD    M0025                 |                       // D -= (0x0025); Zeiger Text-Ende
*FC37: C3 00 01       '...'                   ADDD    #M0001                |                       // D += 0x0001
*FC3A: 18             '.'                     XGDX                          |                       // IX = D tausche Registerinhalt
*FC3B: 9C 5A          '.Z'                    CPX     M005A                 |                       // cmp IX, (0x005a)
*FC3D: 24 09          '$.'                    BCC     ZFC48  >---\          |                       // if (c) == 0; Text-Länge zu kurz?
*FC3F: CE F7 72       '..r'                   LDX     #MF772     |          |                       // IX = 0xf772; "*** WRONG KEY ***"
*FC42: BD EC C5       '...'                   JSR     ZECC5      |          |                       // Call 0xecc5; Ausgabe Text auf LCD; Übergabe IX
*FC45: 7E FA 40       '~.@'                   JMP     ZFA40      |          |                       // JP 0xfa40
// Weiter                                                       |          |
*FC48: DE 27          '.''            ZFC48   LDX     M0027  <---/          |                       // IX = (0x0027); Zeiger Text-aktual Position
*FC4A: 86 8D          '..'                    LDAA    #$8D                  |                       // A = 0x8d
*FC4C: A7 00          '..'                    STAA    ,X                    |                       // (IX) = A
*FC4E: DE 25          '.%'                    LDX     M0025                 |                       // IX = (0x0025); Zeiger Text-Ende
*FC50: 09             '.'                     DEX                           |                       // IX--
*FC51: A6 00          '..'                    LDAA    ,X                    |                       // A = (IX)
*FC53: 81 0A          '..'                    CMPA    #$0A                  |                       // CMP A, 0x0a
*FC55: 25 04          '%.'                    BCS     ZFC5B >-\             |                       // if (c) == 1
*FC57: 81 50          '.P'                    CMPA    #$50    |             |                       // CMP A, 0x50
*FC59: 23 04          '#.'                    BLS     ZFC5F   | >-\         |                       // if (c) | (z) == 1
*FC5B: 86 28          '.('            ZFC5B   LDAA    #$28 <--/   |         |                       // A = 0x28; Wr/Zv wird durch "(" ersetzt
*FC5D: A7 00          '..'                    STAA    ,X          |         |                       // (IX) = A
*FC5F: 97 3C          '.<'            ZFC5F   STAA    M003C <-----/         |                       // (0x003c) = A
*FC61: DE 25          '.%'            ZFC61   LDX     M0025           <---/                         // IX = (0x0025); Zeiger Text-aktual Position
*FC63: DF 2D          '.-'                    STX     M002D                                         // (0x002d) = IX; Zeiger Text-Ende
*FC65: BD E5 BC       '...'                   JSR     ZE5BC                                         // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*FC68: 7F 00 30       '..0'                   CLR     >M0030                                        // (0x0030) = 0x00
*FC6B: 39             '9'                     RTS                                                   // RETURN

// MFC6C
*FC6C: 2A 2A 20 50 58 20 57 49 4C 4C 20 4E 4F 54 20 45                                              // "** PX WILL NOT ENCRYPT (HEX) NUMBERS **"
*FC7C: 4E 43 52 59 50 54 20 28 48 45 58 9 20 4E 55 4D
*FC8C: 42 45 52 53 20 2A AA

// Call Sprungverteiler 29: Eingabe Schlüssel
*FC93: CE F7 CA                               LDX     MF7CA
*FC96: BD ED 82       '...'                   JSR     ZED82                                         // Call 0xed82; Fülle 0x00cd ... 0x00f5 mit 0x20
*FC99: BD ED CD       '...'                   JSR     ZEDCD                                         // Call 0xedcd; Übergabe IX (Textadresse, Textlänge)
*FC9C: 86 09          '..'                    LDAA    #$09                                          // A = 0x09
*FC9E: 97 32          '.2'                    STAA    M0032                                         // (0x0032) = A; Initialwert
*FCA0: 96 32          '.2'            ZFCA0   LDAA    M0032                       <---\             // A = (0x0032)
*FCA2: 36             '6'                     PSHA                                    |             // Push A; Rette (0x0032)
*FCA3: 86 29          '.)'                    LDAA    #$29                            |             // A = 0x29; 40 Elemente
*FCA5: 97 32          '.2'                    STAA    M0032                           |             // (0x0032) = A
*FCA7: BD EC F9       '...'                   JSR     ZECF9                           |             // Call 0xecf9 Übergabe (0x0032)
*FCAA: 32             '2'                     PULA                                    |             // Pop A;  aus Push A
*FCAB: 97 32          '.2'                    STAA    M0032                           |             // (0x0032) = A
*FCAD: BD F3 F9       '...'           ZFCAD   JSR     ZF3F9                   <---\   |             // Call 0xf3f9; Tastendruck; Rückgabe A
*FCB0: 81 AB          '..'                    CMPA    #$AB                        |   |             // CMP A, 0xab; Taste "CODE"
*FCB2: 27 3B          ''';'                   BEQ     ZFCEF >---\                 |   |             // if (z) == 1
*FCB4: 81 AC          '..'                    CMPA    #$AC      |                 |   |             // CMP A, 0xac; Taste "KEY"
*FCB6: 27 37          ''7'                    BEQ     ZFCEF >---+                 |   |             // if (z) == 1
*FCB8: 81 90          '..'                    CMPA    #$90      |                 |   |             // CMP A, 0x90; Taste "<-"
*FCBA: 24 17          '$.'                    BCC     ZFCD3     | >---\           |   |             // if (c) == 0
*FCBC: CE 00 CD       '...'           ZFCBC   LDX     #M00CD    |     |           |   |             // IX = 0x00cd
*FCBF: D6 32          '.2'                    LDAB    M0032     |     |           |   |             // B = (0x0032)
*FCC1: 3A             ':'             ZFCC1   ABX               |     |           |   |             // IX += B
*FCC2: 8C 00 E6       '...'                   CPX     #M00E6    |     |           |   |             // cmp IX, 0x00e6
*FCC5: 26 05          '&.'                    BNE     ZFCCC     |     |  >---\    |   |             // if (z) == 0
*FCC7: BD EA 94       '...'                   JSR     ZEA94     |     |      |    |   |             // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*FCCA: 20 E1          ' .'            ZFCCA   BRA     ZFCAD     |     |      | >--/   |             // JR 0xfca0; Tastendruck abrufen
```

```
                                                    |     |     |        |
*FCCC: A7 00          '..'            ZFCCC   STAA    ,X        |     |  <---/        |           // (IX) = A
*FCCE: 7C 00 32       '|.2'                   INC     >M0032    |     |               |           // (0x0032)++
*FCD1: 20 CD          ' .'                    BRA     ZFCA0     |     |           >---+           // JR 0xfca0; Tastendruck abrufen
                                                    |     |     |                   |
*FCD3: 81 97          '..'            ZFCD3   CMPA    #$97      | <---/               |           // CMP A, 0x97; Taste DEL
*FCD5: 27 05          ''.'                    BEQ     ZFCDC     | >---\               |           // if (z) == 1
*FCD7: BD EA 94       '...'           ZFCD7   JSR     ZEA94     |     | <---\         |           // Call 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*FCDA: 20 C4          ' .'                    BRA     ZFCA0     |     |     |     >---+           // JR 0xfca0; Tastendruck abrufen
                                                    |     |     |     |         |
*FCDC: D6 32          '.2'            ZFCDC   LDAB    M0032     | <---/     |         |           // B = (0x0032)
*FCDE: 5A             'Z'                     DECB              |           |         |           // B--
*FCDF: C1 09          '..'                    CMPB    #$09      |           |         |           // cmp B, 0x09
*FCE1: 25 F4          '%.'                    BCS     ZFCD7     |       >---+         |           // if (c) == 1
*FCE3: D7 32          '.2'                    STAB    M0032     |           |         |           // (0x0032) = B
*FCE5: CE 00 CD       '...'                   LDX     #M00CD    |           |         |           // IX = 0x00cd
*FCE8: 3A             ':'                     ABX               |           |         |           // IX += B
*FCE9: 86 2E          '..'                    LDAA    #$2E      |           |         |           // A = 0x2e
*FCEB: A7 00          '..'                    STAA    ,X        |           |         |           // (IX) = A
*FCED: 20 B1          ' .'                    BRA     ZFCA0     |           |     >---/           // JR 0xfca0; Tastendruck abrufen
// Taste KEY gedrückt                                           |           |
*FCEF: C6 10          '..'            ZFCEF   LDAB    #$10  <---/           |                     // B = 0x10; 16 Zeichen
// Schleife                                                     |
*FCF1: CE 00 D5       '...'           ZFCF1   LDX     #M00D5 <--\           |                     // IX = 0x00d5
*FCF4: 3A             ':'                     ABX               |           |                     // IX += B; 0xe5 ... 0x00d6
*FCF5: A6 00          '..'                    LDAA    ,X        |           |                     // A = (IX)
*FCF7: 84 0F          '..'                    ANDA    #$0F      |           |                     // A &= 0x0f Maskiere
*FCF9: CE 01 50       '..P'                   LDX     #M0150    |           |                     // IX = 0x0150
*FCFC: 3A             ':'                     ABX               |           |                     // IX += B; 0x0160 ... 0x0151
*FCFD: A7 00          '..'                    STAA    ,X        |           |                     // (IX) = A; Kopiere (0x00d5 + n) untere 4 Bits nach (0x0150 + n)
*FCFF: 5A             'Z'                     DECB              |           |                     // B--
*FD00: 26 EF          '&.'                    BNE     ZFCF1 >---/           |                     // if (z) == 0; Schleife 16 Runden
*FD02: CE F7 E4       '...'                   LDX     #MF7E4                |                     // IX = 0xf7e4; "< NEW KEY ACCEPTED >"
*FD05: 7E EC C5       '~..'                   JMP     ZECC5             >---/                     // JP 0xecc5; Ausgabe Text auf LCD; Übergabe IX

// Call Sprungverteiler 30: Ein-/Ausgabe Schlüssel
*FD08: CE F7 F8       '...'                   LDX     #MF7F8                                      // IX = 0xf7f8; "NEW KEY: TEXT 00 "
*FD0B: BD EC EA       '...'                   JSR     ZECEA                                       // Call 0xecea; Textausgabe LCD, Übergabe IX
*FD0E: C6 0E          '..'                    LDAB    #$0E                                        // B = 0x0e
*FD10: CE AB AD       '...'                   LDX     #MABAD                                      // IX = 0xabad
*FD13: DF 5A          '.Z'                    STX     M005A                                       // (0x005a) = IX
*FD15: BD EA F5       '...'                   JSR     ZEAF5                                       // Call 0xeaf5; LCD Ausgaben, Übergabe B
*FD18: 4D             'M'                     TSTA                                                // A == 0?; MSB == 1 n-Flag, A == 0 z-Flag
*FD19: 26 03          '&.'                    BNE     ZFD1E >----\                                // if (z) == 0
*FD1B: 7E EA 94       '~..'                   JMP     ZEA94      |                                // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                 |
// Call Übergabe: A Rückgabe:                                    |
*FD1E: DE 2D          '.-'            ZFD1E   LDX     M002D <----/                                // IX = (0x002d); Zeiger Text-Start-Ende-Länge
*FD20: 3C             '<'                     PSHX                                                // Push IX; Rette Zeiger
*FD21: D6 2F          './'                    LDAB    M002F                                       // B = (0x002f); Rette Zählervariable 0x1 ... 0x62
*FD23: 37             '7'                     PSHB                                                // Push B
*FD24: 97 2F          './'                    STAA    M002F                                       // (0x002f) = A; A aus Call 0xeaf5 + Zählervariable 0x1 ... 0x62
*FD26: BD E2 0D       '...'                   JSR     ZE20D                                       // Call 0xe20d;Inhalt aus IX = (0x018e + 2*(0x002f)) Rückgabe Flag
*FD29: 27 05          ''.'                    BEQ     ZFD30 >----\                                // if (z) == 1
*FD2B: BD EA DF       '...'                   JSR     ZEADF      |                                // Call 0xeadf; IF (IX)(0x018e + 2 * (0x002f)) == 0; A = 0x00 else A = 0xff
*FD2E: 26 0F          '&.'                    BNE     ZFD3F      | >--\                          // if (z) == 0
*FD30: 33             '3'             ZFD30   PULB          <----/    |                          // Pop B; Stelle 0x002f wieder her
*FD31: D7 2F          './'                    STAB    M002F           |                          // (0x002f) = B; Zählervariable 0x1 ... 0x62
*FD33: BD E2 1A       '...'                   JSR     ZE21A           |                          // Call  0xe21a  Init Textlänge Zeiger
*FD36: 38             '8'                     PULX                    |                          // Pop IX; Wiederherstellen Zeiger
*FD37: DF 2D          '.-'                    STX     M002D           |                          // (0x002d) = IX
*FD39: BD E5 BC       '...'                   JSR     ZE5BC           |                          // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*FD3C: 7E EA 94       '~..'                   JMP     ZEA94           |                          // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                      |
Passwort-Verarbeitung                                               |
*FD3F: BD E2 1A       '...'           ZFD3F   JSR     ZE21A       <---/                           // Call  0xe21a Init Textlänge Zeiger
*FD42: C6 0F          '..'                    LDAB    #$0F                                        // B = 0x0f Schleifenzähler
*FD44: DE 25          '.%'            ZFD44   LDX     M0025 <----\                                // IX = (0x0025); Zeiger Text-Start-Ende-Länge
*FD46: 3A             ':'                     ABX               |                                 // IX += B; Zeiger + Schleife
*FD47: A6 00          '..'                    LDAA    ,X        |                                 // A = (IX); Hole Wert
*FD49: 84 0F          '..'                    ANDA    #$0F      |                                 // A &= 0x0f Maskiere
*FD4B: CE 01 51       '..Q'                   LDX     #M0151    |                                 // IX = 0x0151
*FD4E: 3A             ':'                     ABX               |                                 // IX += B; 0x0151 ... 0x0160
*FD4F: A7 00          '..'                    STAA    ,X        |                                 // (IX) = A
*FD51: 5A             'Z'                     DECB              |                                 // B--
*FD52: 2A F0          '*.'                    BPL     ZFD44 >---+                                 // if (n) == 0; bit7 == 0; Schleife
*FD54: 27 EE          ''.'                    BEQ     ZFD44 >---/                                 // if (z) == 1;  >= 0; Schleife
// Kopiere von (0x0025) + n nach (0x0151) + n; die unteren 4 Bits des Inhaltes der Adressen; Aus Schlüsselwort die unteren 4bits
*FD56: 32             '2'                     PULA                                                // Pop A; entspricht B aus Push B
*FD57: 97 2F          './'                    STAA    M002F                                       // (0x002f) = A; Zählervariable 0x1 ... 0x62
*FD59: BD E2 1A       '...'                   JSR     ZE21A                                       // Call  0xe21a  Init Textlänge Zeiger
*FD5C: 38             '8'                     PULX                                                // Pop IX
*FD5D: DF 2D          '.-'                    STX     M002D                                       // (0x002d) = IX
*FD5F: BD E5 BC       '...'                   JSR     ZE5BC                                       // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*FD62: CE F7 E4       '...'                   LDX     #MF7E4                                      // IX = 0xf7e4 "< NEW KEY ACCEPTED >"
*FD65: 7E EC C5       '~..'                   JMP     ZECC5                                       // JP 0xecc5; Ausgabe Text auf LCD; Übergabe IX

// Call
*FD68: BD E2 12       '...'           ZFD68   JSR     ZE212                                       // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*FD6B: EC 00          '..'                    LDD     ,X                                          // D = (IX)
*FD6D: 27 53          ''S'                    BEQ     ZFDC2          >---\                        // if (z) == 1; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*FD6F: 2A 54          '*T'                    BPL     ZFDC5              |  >---\                   // if (n) == 0; bit7 == 0 springe
*FD71: 84 7F          '..'                    ANDA    #$7F               |      |                   // A &= 0x7f Maskiere
*FD73: 05             '.'                     ASLD                       |      |                   // c <- A7 ... A0 <- B7 ... B0 <- 0 Shift left
*FD74: C3 00 01       '...'                   ADDD    #M0001             |      |                   // D += 0x0001; Setze Bit 0 in B
```

```
*FD77: DE 27          '.''                    LDX     M0027              |      |                // IX = (0x0027)
*FD79: DF 2D          '.-'                    STX     M002D              |      |                // (0x002d) = IX; Zeiger Text-Start-Ende-Länge
*FD7B: BD EA 26       '..&'                   JSR     ZEA26              |      |                // Call 0xea26; Prüfung freie Länge Textspeicher, Übergabe D Rückgabe D
*FD7E: DE 27          '.''                    LDX     M0027              |      |                // IX = (0x0027)
*FD80: 86 8D          '..'                    LDAA    #$8D               |      |                // A = 0x8d
*FD82: A7 00          '..'                    STAA    ,X                 |      |                // (IX) = A
*FD84: DF 5A          '.Z'                    STX     M005A              |      |                // (0x005a) = IX
*FD86: DE 2D          '.-'                    LDX     M002D              |      |                // IX = (0x02d)
*FD88: A6 00          '..'            ZFD88   LDAA    ,X           <---\ |      |                // A = (IX)
*FD8A: BD F5 8C       '...'                   JSR     ZF58C              | |      |                // Call 0xf58c  Wandelt Hex in ASCII-Hex-Code 0x0 -> "00" 0xff -> "FF"; Übergabe A; Rückgabe B, A
*FD8D: DE 5A          '.Z'                    LDX     M005A              | |      |                // IX = (0x005a)
*FD8F: 09             '.'                     DEX                        | |      |                // IX--
*FD90: 09             '.'                     DEX                        | |      |                // IX--
*FD91: 09             '.'                     DEX                        | |      |                // IX--
*FD92: A7 02          '..'                    STAA    $02,X              | |      |                // (IX + 0x02) = A
*FD94: E7 01          '..'                    STAB    $01,X              | |      |                // (IX + 0x01) = B
*FD96: 86 20          '. '                    LDAA    #$20               | |      |                // A = 0x20
*FD98: A7 00          '..'                    STAA    ,X                 | |      |                // (IX) = A
*FD9A: DF 5A          '.Z'                    STX     M005A              | |      |                // (0x005a) = IX
*FD9C: DE 2D          '.-'                    LDX     M002D              | |      |                // IX = (0x002d); Zeiger Text-Start
*FD9E: 9C 25          '.%'                    CPX     M0025              | |      |                // cmp IX, (0x0025); Zeiger Text-Ende
*FDA0: 25 05          '%.'                    BCS     ZFDA7 >---\        | |      |                // if (c) == 1; Abbruch Schleife
*FDA2: 09             '.'                     DEX                  |     | |      |                // IX--
*FDA3: DF 2D          '.-'                    STX     M002D        |     | |      |                // (0x002d) = IX
*FDA5: 20 E1          ' .'                    BRA     ZFD88        | >---/ |      |                // JR 0xfd88; Schleife
// JP                                                              |       |      |
*FDA7: 86 18          '..'            ZFDA7   LDAA    #$18  <---/        |      |                // A = 0x18
*FDA9: 97 3C          '.<'                    STAA    M003C              |      |                // (0x003c) = A
*FDAB: A7 00          '..'                    STAA    ,X                 |      |                // (IX) = A
*FDAD: 08             '.'                     INX                        |      |                // IX++
*FDAE: DF 2D          '.-'                    STX     M002D              |      |                // (0x002d) = IX
*FDB0: 7F 00 30       '..0'                   CLR     >M0030             |      |                // (0x0030) = 0x00
*FDB3: BD E2 12       '...'                   JSR     ZE212              |      |                // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*FDB6: A6 00          '..'                    LDAA    ,X                 |      |                // A = (IX)
*FDB8: 84 7F          '..'                    ANDA    #$7F               |      |                // A &= 0x7f Maskiere
*FDBA: A7 00          '..'                    STAA    ,X                 |      |                // (IX) = A
*FDBC: BD E5 BC       '...'                   JSR     ZE5BC              |      |                // Call 0xe5bc; Vergleiche/Suche (0x002d) (0x0027) (0x0025) Rückgabe B (0x0034), IX (0x002b)
*FDBF: 7E EA 01       '~..'                   JMP     ZEA01              |      |                // JP 0xea01; Textanfang-Ende GTX gefunden?
                                                                         |      |
// JP                                                                    |      |
*FDC2: 7E EA 94       '~..'           ZFDC2   JMP     ZEA94 <------------+      |                // JP 0xea94; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
                                                                                |
// JP                                                                           |
*FDC5: BD FE 21       '..!'           ZFDC5   JSR     ZFE21              |  <---/                // Call 0xfe21; Abfrage IX (0x0025) <= (0x0027); while (0x0025 <= 0x0027) ix++ 0x0025 erstes ASCII im Speicher TestIfStringIsHex
*FDC8: 25 F8          '%.'                    BCS     ZFDC2 >------------/                      // if (c) == 1; Sprungverteiler 17, 18; Modem ON Übergabe A Port-1x, B Zeitschleife 100ms
*FDCA: BD E2 12       '...'                   JSR     ZE212                                     // Call 0xe212; Übergabe (0x002f); Rückgabe IX = 0x018e + 2*(0x002f) Z-Flag
*FDCD: A6 00          '..'                    LDAA    ,X                                        // A = (IX)
*FDCF: 8A 80          '..'                    ORAA    #$80                                      // A |= 0x80 Setze bit7
```

```
*FDD1: A7 00          '..'                    STAA    ,X                                        // (IX) = A
*FDD3: 5F             '_'                     CLRB                                              // B = 0x00
*FDD4: DE 25          '.%'                    LDX     M0025                                     // IX = (0x0025); Zeiger Text-Start
*FDD6: 09             '.'                     DEX                                               // IX--
*FDD7: DF 5A          '.Z'                    STX     M005A                                     // (0x005a) = IX; Zeiger Text-Start-neu -1
*FDD9: 08             '.'                     INX                                               // IX++
*FDDA: DF 2D          '.-'            ZFDDA   STX     M002D              <---\                  // (0x002d) = IX; Zeiger Text-Start
*FDDC: A6 00          '..'                    LDAA    ,X                     |                  // A = (IX); erstes Zeichen
*FDDE: 84 7F          '..'                    ANDA    #$7F                   |                  // A &= 0x7f Maskiere
*FDE0: 81 20          '. '                    CMPA    #$20                   |                  // CMP A, 0x20: " "
*FDE2: 27 26          ''&'                    BEQ     ZFE0A >---\            |                  // if (z) == 1
*FDE4: 81 2C          '.,'                    CMPA    #$2C      |            |                  // CMP A, 0x2c "-"
*FDE6: 27 22          ''"'                    BEQ     ZFE0A >---+            |                  // if (z) == 1
*FDE8: 81 0D          '..'                    CMPA    #$0D      |            |                  // CMP A, 0x0d; WR/ZV Enter
*FDEA: 27 1E          ''.'                    BEQ     ZFE0A >---+            |                  // if (z) == 1
*FDEC: 80 30          '.0'                    SUBA    #$30      |            |                  // A-= 0x30; "0"
*FDEE: 81 09          '..'                    CMPA    #$09      |            |                  // CMP A, 0x09; ehemals "9"
*FDF0: 23 04          '#.'                    BLS     ZFDF6     | >---\      |                  // if (c) | (z) == 1
*FDF2: 84 1F          '..'                    ANDA    #$1F      |     |      |                  // A &= 0x1f Maskiere; größer ASCII "9"
*FDF4: 80 07          '..'                    SUBA    #$07      |     |      |                  // A-= 0x07; Sonderzeichen zwischen Ziffern und Buchstaben
*FDF6: DE 5A          '.Z'            ZFDF6   LDX     M005A     | <---/      |                  // IX = (0x005a); Zeiger Text-Start-neu -1
*FDF8: 68 00          'h.'                    ASL     ,X        |            |                  // c <- IX7 ... IX0 <- 0 Shift left
*FDFA: 68 00          'h.'                    ASL     ,X        |            |                  // c <- IX7 ... IX0 <- 0 Shift left
*FDFC: 68 00          'h.'                    ASL     ,X        |            |                  // c <- IX7 ... IX0 <- 0 Shift left
*FDFE: 68 00          'h.'                    ASL     ,X        |            |                  // c <- IX7 ... IX0 <- 0 Shift left
*FE00: AA 00          '..'                    ORAA    ,X        |            |                  // A |= (IX); Lowteil aus A, High-Teil Ah |= IXh; tausch High-low 0xf0 -> 0x0f
*FE02: A7 00          '..'                    STAA    ,X        |            |                  // (IX) = A
*FE04: 53             'S'                     COMB              |            |                  // Complement B; 0x0 -> 0xff; je nach Runde 0x0 oder 0xff
*FE05: 2B 03          '+.'                    BNI     ZFE0A >---+            |                  // if (n) == 1; bit7 == 1; wenn B == 0
*FE07: 08             '.'                     INX               |            |                  // IX++; Zähler
*FE08: DF 5A          '.Z'                    STX     M005A     |            |                  // (0x005a) = IX
*FE0A: DE 2D          '.-'            ZFE0A   LDX     M002D <---/            |                  // IX = (0x002d); Zeiger Text-Start
*FE0C: 08             '.'                     INX                            |                  // IX++
*FE0D: 9C 27          '.''                    CPX     M0027                  |                  // cmp IX, (0x0027); Zeiger Text-aktual Position
*FE0F: 23 C9          '#.'                    BLS     ZFDDA           >------/                  // if (c) | (z) == 1, noch nicht am Ende
*FE11: 18             '.'                     XGDX                                              // IX = D; Tausche Registerinhalt
*FE12: 93 5A          '.Z'                    SUBD    M005A                                     // D = (0x002d) - (0x005a); Länge
*FE14: DE 5A          '.Z'                    LDX     M005A                                     // IX = (0x005a)
*FE16: DF 2D          '.-'                    STX     M002D                                     // (0x002d) = IX; neuer Zeiger Text-Start
*FE18: BD EA 73       '..s'                   JSR     ZEA73                                     // Call 0xea73; Übergabe D
*FE1B: DE 2D          '.-'                    LDX     M002D                                     // IX = (0x002d)
*FE1D: 09             '.'                     DEX                                               // IX--
*FE1E: DF 27          '.''                    STX     M0027                                     // (0x0027) = IX
*FE20: 39             '9'                     RTS                                               // RETURN


// Call Abfrage IX (0x0025) <= (0x0027); while (0x0025 <= 0x0027) ix++ 0x0025 erstes ASCII im Speicher
```

```
// Test String == HEX?
*FE21: DE 25          '.%'            ZFE21   LDX     M0025                                   // IX = (0x0025); Zeiger Text-Start
*FE23: 5F             '_'                     CLRB                                            // B = 0x00
*FE24: A6 00          '..'            ZFE24   LDAA    ,X                      <---\           // while (0x0025 <= 0x0027)
*FE26: 84 7F          '..'                    ANDA    #$7F                        |           // A &= 0x7f Maskiere ASCII
*FE28: 81 0D          '..'                    CMPA    #$0D                        |           // CMP A, 0x0d; WR/ZV Enter
*FE2A: 27 27          ''''                    BEQ     ZFE53   >---\               |           // if (z) == 1; Textende? ohne carray
*FE2C: 81 20          '. '                    CMPA    #$20        |               |           // CMP A, 0x20 " "
*FE2E: 27 20          '' '                    BEQ     ZFE50 >---\ |               |           // if (z) == 1;
*FE30: 81 2C          '.,'                    CMPA    #$2C      | |               |           // CMP A, 0x2c ","
*FE32: 27 1C          ''.'                    BEQ     ZFE50 >---+ |               |           // if (z) == 1;
*FE34: 81 30          '.0'                    CMPA    #$30      | |               |           // CMP A, 0x30 "0"
*FE36: 25 21          '%!'                    BCS     ZFE59     | |   >-----\     |           // if (c) == 1 RETURN c = 1
*FE38: 81 39          '.9'                    CMPA    #$39      | |         |     |           // CMP A, 0x39 "9"
*FE3A: 23 10          '#.'                    BLS     ZFE4C     | | >---\   |     |           // if (c) | (z) == 1
*FE3C: 81 41          '.A'                    CMPA    #$41      | |     |   |     |           // CMP A, 0x41 "A"
*FE3E: 25 19          '%.'                    BCS     ZFE59     | |     | >-+     |           // if (c) == 1 RETURN c = 1
*FE40: 81 46          '.F'                    CMPA    #$46      | |     |   |     |           // CMP A, 0x46 "F"
*FE42: 23 08          '#.'                    BLS     ZFE4C     | | >---+   |     |           // if (c) | (z) == 1
*FE44: 81 61          '.a'                    CMPA    #$61      | |     |   |     |           // CMP A, 0x61 "a"
*FE46: 25 11          '%.'                    BCS     ZFE59     | |     | >-+     |           // if (c) == 1 RETURN c = 1
*FE48: 81 66          '.f'                    CMPA    #$66      | |     |   |     |           // CMP A, 0x66 "f"
*FE4A: 22 0D          '".'                    BHI     ZFE59     | |     | >-+     |           // if (c) & (z) == 0 RETURN c = 1
// set b true                                                   | |     |   |     |
*FE4C: C8 01          '..'            ZFE4C   EORB    #$01      | | <---/   |     |           // B XOR= 0x01; setze B = 1
*FE4E: 20 03          ' .'                    BRA     ZFE53 >---|-+         |     |           // Textende? ohne carray
                                                               | |           |     |
// B == 1                                                      | |           |     |
*FE50: 5D             ']'             ZFE50   TSTB          <---/ |         |     |           // B == 0?; MSB == 1 n-Flag, B == 0 z-Flag  break b = not/false (0)
*FE51: 26 06          '&.'                    BNE     ZFE59       |     >---+     |           // if (z) == 0 RETURN carray = 1
// Ende Text? ohne carray                                        |             |     |
*FE53: 08             '.'             ZFE53   INX             <---/           |     |           // IX++
*FE54: 9C 27          '.''                    CPX     M0027                   |     |           // cmp IX; (0x0027)
*FE56: 23 CC          '#.'                    BLS     ZFE24                   | >---/           // if (c) | (z) == 1
*FE58: 39             '9'                     RTS                             |                 // RETURN
                                                                              |
// Return carry = 1                                                           |
*FE59: 0D             '.'             ZFE59   SEC                     <-----/                 // c = 1
*FE5A: 39             '9'                     RTS                                             // RETURN c = 1

Permutationstabelle
// No.:00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
++++++++++++++++++++++++++++++++++++++++++++++++++++++
*FE5B: 06 0A 01 02 0E 07 08 09 0C 04 03 0F 0B 00 05 0D
*FE6B: 02 07 00 05 0F 03 01 09 0E 0D 0B 08 0C 0A 04 06
*FE7B: 09 01 03 0C 02 00 0A 0E 0D 06 0B 07 05 08 0F 04
*FE8B: 0B 06 09 00 05 0E 0F 08 04 0C 01 0D 02 07 03 0A

//  Substitutions-Tabelle MFE9B
*FE9B: 96 4B 65 3A AC 6C 53 74 78 A5 47 B2 4D A6 59 5A
*FEAB: 8D 56 2B C3 71 D2 66 3C 1D C9 93 2E A9 72 17 B1
*FEBB: B4 E4 A3 4E 27 5C 8B C5 E8 95 E1 D1 87 B8 1E CA
*FECB: 1B 63 D8 2D D4 9A 99 36 8E C6 69 E2 39 35 6A 9C

// Data MFEDB
// Index+
//     0f 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e                                      // Index der Masken-Tabelle
++++++++++++++++++++++++++++++++++++++++++++++++++++++
*FEDB:    0B 0A 78 0C E0 29 7B CF C3 4B 2B CC 82 60 80
*FEEA: 06 0B 0A 78 0C E0 29 7B CF C3 4B 2B CC 82 60 80

// Data
*FEFA: AA 55 55 AA 55

*FEFF: 7E E0 03       '~..'           hdlr_RST JMP     ZE003                                // Einsprung aus der Tabelle Interrupt-Routinen: 0xffee, ....

// Zeichenwandlung Tastatur Spalte:'01234567'       Zeile    MFF02 ohne Shift
*FF02: 00 2D 38 36 35 33 31 B1     '.-86531.'        A0; B1 -> Transmit
*FF0A: 30 00 00 37 79 34 32 9E     '0..7y42.'        A1; 9E -> Print
*FF12: 3A A8 00 68 67 72 73 71     ':..hgrsq'        A2; A8 -> Tab
*FF1A: A0 0D 9A 00 92 93 91 A1     ' .......'        A3; Shift L, Enter, Clear All, ^ hoch, v runter, > rechts, Shift R
*FF22: 97 A2 2C 6E 00 63 90 B0     '.",n.c..'        A4; B0 -> RCVE, 90 -> "< links"; A2 -> Search; 97 -> DEL
*FF2A: 70 9C 69 75 74 00 77 AB     'p.iut.w.'        A5; AB -> Code; 9C -> Insert
*FF32: 3B B3 6B 6A 66 64 00 9b     ';.kjfd..'        A6; 9B -> MARG; B3 -> DUMP
*FF4A: 2E 96 6C 62 76 78 7A 00     '..lbvxz.'        A7; 96 -> TEXT
*FF42: 39 00 6F 20 6D 65 61 AF     '9.o mea.'        A8; AF -> Caps Lock

// Zeichenwandlung Tastatur Spalte:'01234567'       Zeile   MFF4A Shift left
*FF4A: 00 2B 7E 7C 02 23 21 00     '.+~|Ä#!.'        A0
*FF52: 3D 00 00 5F 59 24 22 9F     '=..ÖY$".'        A1; 9F -> List
*FF5A: 3C A9 00 48 47 52 53 51     '<..HGRSQ'        A2; A9 -> SET
*FF62: A0 0D 98 00 A4 A6 A5 A1     ' .......'        A3; Shift L, Enter, Clear Line, ^ hoch Anfang, v runter Ende, > rechts Ende, Shift R
*FF6A: 99 A2 5B 4E 00 43 A7 B0     '."[N.C..'        A4; A7 -> "< links Anfang"; 99 -> DEL TXT
*FF72: 50 9D 49 55 54 00 57 AC     'P.IUT.W,'        A5; AC -> Key; 9D -> INS TXT
*FF7A: 3F B2 4B 4A 46 44 00 AE     '?.KJFD..'        A6; AE -> MARG LOW; B2 -> LOAD
*FF82: 2F 94 4C 42 56 58 5A 00     '/.LBVXZ.'        A7; 94 -> TEXT runter;
*FF8A: 28 00 4F 20 4D 45 41 AF     '(.O MEA.'        A8; AF -> Caps Lock

// Zeichenwandlung Tastatur Spalte:'01234567'       Zeile   MFF92 Shift right
*FF92: 00 25 7D 7B 03 60 40 00     '.%{.`äL@'        A0
*FF9A: 7F 00 00 5E 59 2A 26 9F     '0..öY*&.'        A1; 9F -> List
*FFA2: 3E AA 00 48 47 52 53 51     '>..HGRSQ'        A2; AA -> CLR
*FFAA: A0 0D 98 00 A4 A6 A5 A1     ' .......'        A3; Shift L, Enter, Clear Line, ^ hoch Anfang, v runter Ende, > rechts Ende, Shift R
*FFB2: 99 A2 5D 4E 00 43 A7 B0     '."]N.C..'        A4; A7 -> "< links Anfang"; 99 -> DEL TXT
*FFBA: 50 9D 49 55 54 00 57 AD     'P.IUT.W-'        A5; AD -> Key TXT; 9D -> INS TXT
```

```
*FFC2: 27 B2 4B 4A 46 44 00 A3     ''W'.KJFd'        A6; A3 -> MARG HIGH; B2 -> LOAD
*FFCA: 5C 95 4C 42 56 58 5A 00     '\.LBVXZ.'        A7; 95 -> TEXT hoch;
*FFD2: 29 00 4F 20 4D 45 41 AF     ').O MEA.'        A8; AF -> Caps Lock


// SP- Adressierung in 0xf50e und 0xf520 -Werte nach (IX) 0x016a + n ablegen; für Conversions-Char
*FFDA: 02 03 5E 5F 7C 7D 7E        '..^_|}~'           MFFD9 + 1 wg. Pop A; erste Schleife  -> 0x016a ... 0x0170
*FFE1: 5D 7D 7C 5C 5B 7E 5E        ']}|\[~^'                                zweite Schleife -> 0x017D ... 0x0183
*FFE8: 4F 44 55 4C 45 00           'ODULE'                                                   //

/******************/
/* VECTOR ADRESSES */
/******************/

// Interrupt-Sprungtabelle
*FFEE: FE FF                                       CPU Interrupt TRAP                      // 0xfeff -> JMP 0xE003 Restart

// Interrupt Vectoring memory map
// /IRQ2
*FFF0: F7 49                       sci_vector sci_entry   CPU SCI (RDRF + ORFE + TDRE)       // 0xf749 -> RTI
*FFF2: F6 F6                       tof_vector tof_entry   CPU TOF Timer Overflow             // Funktion 0xf6f6
*FFF4: EF E5                       ocf_vector ocf_entry   CPU OCF Timer Output Compare       // Funktion 0xefe5; Ausgabe 0 oder 1 auf Port-1 und -2
*FFF6: F7 49                       icf_vector icf_entry   CPU ICF Timer Input Capture        // 0xf749 -> RETI
// /IRQ1
*FFF8: FE FF                       irq_vector int_entry   CPU /IRQ oder /IS3                 // 0xfeff -> JMP 0xE003 Restart
*FFFA: F7 49                       swi_vector swi_entry   CPU Software Interrupt             // 0xf749 -> RTI
*FFFC: F7 49                       nmi_vector nmi_entry   CPU /NMI                           // 0xf749 -> RTI
*FFFE: FE FF                       res_vector reset       CPU Interrupt /RES                 // 0xfeff -> JMP 0xE003 Restart
```