DSC analyzer, version SX64

For at være i stand til at teste på DSC radioudstyr[*1] blev programmet ***DSC-analyzer*** udviklet i 80erne.

Først kodet til en Commodore SX64 (Fig.1), der som C64 bl.a. udmærker sig ved at have en yderst versatil CIA-chip[*2] der f.eks. kan sættes til at opsamle datastrømme, timet af en præcis ekstern reference.

Programmet er dels kodet i Assembler[*3] for timing-kritisk opsamling og afsendelse af DSC-beskeder på bitniveau, samt i højniveausproget Comal80 som brugerinterface (Fig.2), og præsentation af data (Fig.3).

Koden blev efterhånden så omfattende, at den måtte deles op i moduler, der loades inden programstart (Fig.4).

Fig.1

[*1] *Digital selective calling or DSC is a standard for transmitting pre-defined digital messages via the maritime radio systems. It is a core part of the Global Maritime Distress Safety System (GMDSS).* Kilde

[*2] *Complex Interface Adapter MOS6526*

[*3] *CPU MOS6510* Kilde

[*4] *FEC (Forward Error Correction) The sender encodes the message in a redundant way by using an error-correcting code (ECC).* Kilde

[*5] *ECC (Error Correction Character) er en modulo2 sum af de modtagne primære databit (DX) samt de redundante databit (RX).* Kilde Se evt. også Fig.5

[*6] *UniComal er en object orienteret version af Comal80*

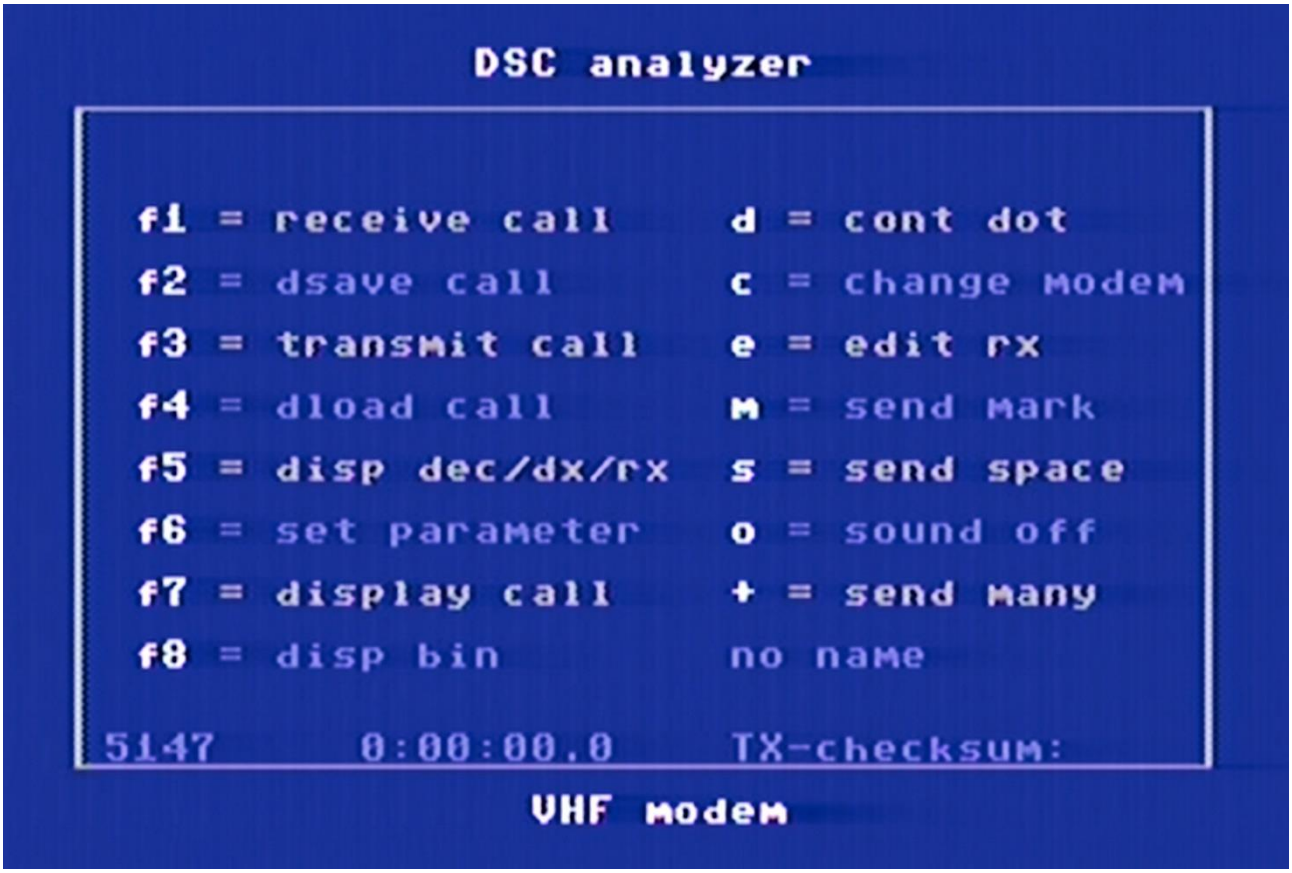


Fig.2 Brugerinterface

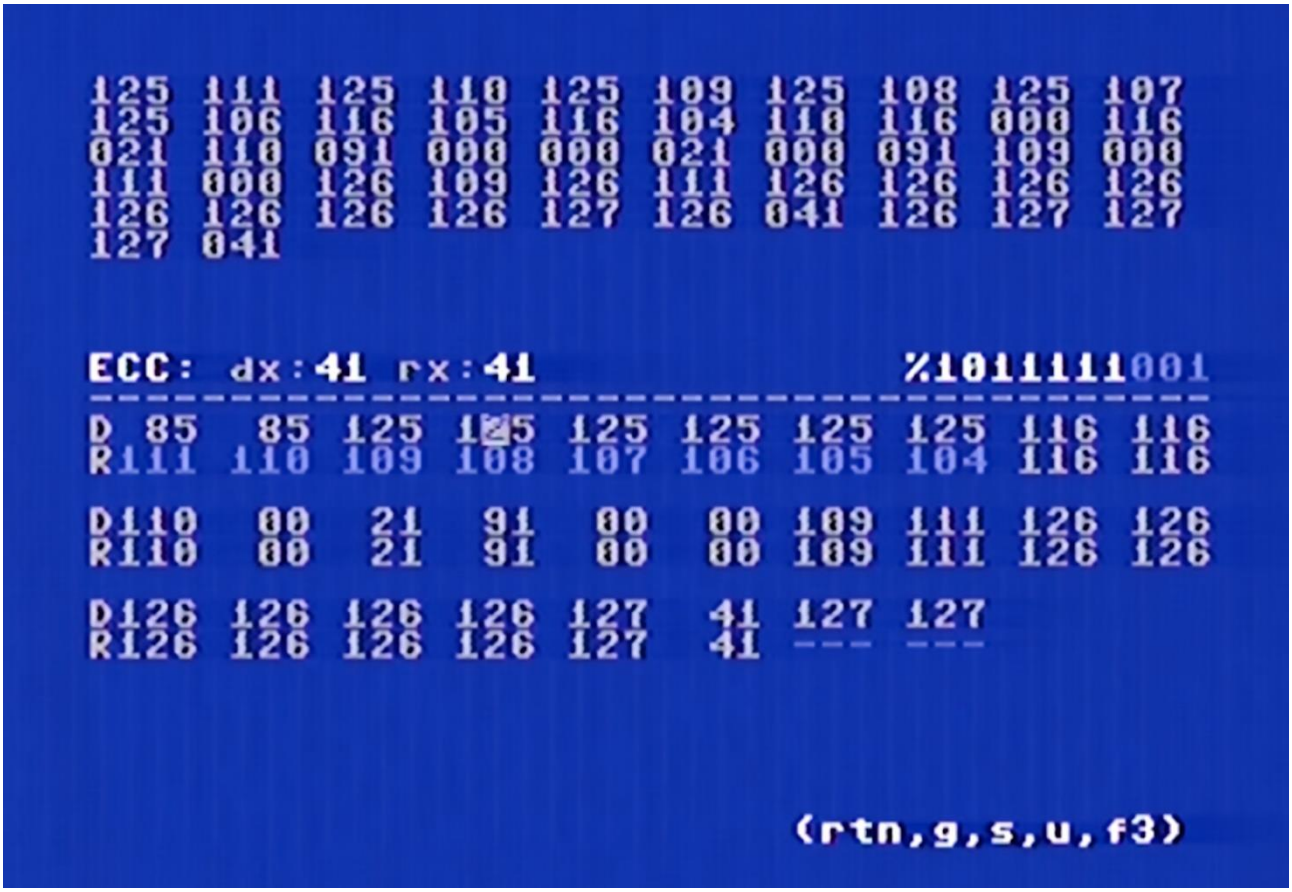


Fig.3 Modtaget DSC-besked. Øverst de modtagne bit (7 databit + 3 paritetsbit) på decimal form, og nederst korrigeret for FEC[*4] -forskydning samt beregnet ECC[*5].

Fig.4 Programmet kunne efterhånden ikke være editorhukommelsen, og måtte deles op i moduler.

```
FUNC ecc#(dxrx$)
  n:=phas_start+24 // 1.format spc
  IF dxrx$="dx" THEN
    ec#:=call#(n)
    n:+8
  ELSE
    n:+10
    ec#:=call#(n)
    n:+8
  ENDIF
  ec#:=ec# BITXOR call#(n)
  REPEAT
    n:+4
    ec#:=ec# BITXOR call#(n)
    eos_test#:=eos#(n,dxrx$)
  UNTIL eos_test# OR eos_test#=-1
  RETURN ec#
ENDFUNC ecc#
```

Fig.5 Eksempel på Comal80-kode. Her funktionen til beregning af ECC[*5] modulo2 sum i ***DSC-analyzer***.

```
8292 ae 07 8a   colcol     ldx nul_et
8295 e0 08      otte       cpx #8
8297 d0 08                 bne ni
8299 a9 9a                 lda #farve3
829b 20 d2 ff              jsr chrout
829e 4c b9 82              jmp skriv
82a1 e0 09      ni         cpx #9
82a3 d0 08                 bne ti
82a5 a9 9a                 lda #farve3
82a7 20 d2 ff              jsr chrout
82aa 4c b9 82              jmp skriv
82ad e0 0a      ti         cpx #10
82af d0 08                 bne skriv
82b1 a9 9a                 lda #farve3
82b3 20 d2 ff              jsr chrout
82b6 4c b9 82              jmp skriv


82b9 a5 a6      skriv      lda $a6
82bb 20 d2 ff              jsr chrout      ;jsr udskriv
82be ae 07 8a              ldx nul_et
82c1 e0 0a                 cpx #10         ;10 bit?
82c3 f0 06                 beq gem_bytes

82c5 20 3c 83              jsr timeout
82c8 4c 43 82              jmp timstart    ;one bit more


82cb ae 06 8a   gem_bytes  ldx husk_x
82ce 18                    clc
82cf 6e 00 8a              ror rul_1_7     ;rul endnu et 0

82d2 a5 c5      chktast    lda $c5
82d4 c9 40                 cmp #64
82d6 f0 07                 beq er_rxslut
82d8 a9 00                 lda #0
82da 85 c6                 sta $c6
82dc 4c f1 82              jmp til_comal

82df ad 01 8a   er_rxslut  lda rul_8_a
82e2 25 07                 and %00000111
82e4 cd 18 8a              cmp rxslutbits
82e7 d0 13                 bne not_retur

82e9 ac 00 8a              ldy rul_1_7
82ec cc 18 8a              cpy rxslutbits
82ef d0 0b                 bne not_retur

82f1 a9 ff      til_comal  lda #%11111111
82f3 9d 00 90              sta rxdataram,x
82f6 e8                    inx
82f7 9d 00 90              sta rxdataram,x
82fa 58                    cli
82fb 60                    rts


82fc ad 00 8a   not_retur  lda rul_1_7
82ff 9d 00 90              sta rxdataram,x
8302 e8                    inx
8303 8e 06 8a              stx husk_x
8306 ad 01 8a              lda rul_8_a
8309 9d 00 90              sta rxdataram,x
830c e8                    inx
830d 8e 06 8a              stx husk_x
```

Fig.6 Som eksempel på assemblerkoden, ses her side 6 af i alt 9 A4-sider.

DSC analyzer, version PC

I starten af 90'erne blev programmet migreret til PC (Fig.7 til 9), nu med UniComal[*6] som højniveau-sprog, og timing-kritisk data-opsamling og afsendelse skrevet i 8086-maskinkode. I mangel af en dedikeret I/O-port, benyttes i stedet signalbenene på PC'ernes RS232-stik.

På linje med alt andet testudstyr, var programmet underlagt laboratoriets akkrediteringskrav, og afslørede gennem tiden utallige afvigelser fra gældende standarder på det udstyr det testede, hvorfor firmaerne viste interesse for at købe det, hvilket dog var mod laboratoriets formål, nemlig at sælge målinger, og ikke SW.

Da laboratoriet var med i internationalt standardiseringsarbejde, blev der på et tidspunkt behov for at afdække sammenhængen mellem BIT-error (BER) og MESSAGE-error (MER) på forskellige beskedtyper, og forskellige DSC-modtagere, visende hvor godt fabrikanterne formåede at udnytte dataformatets indbyggede fejlkorrektioner (FEC[*4], ECC[*5] samt paritetsbit).

Der blev derfor udviklet en udgave at programmet, hvor indsættelse af et stigende antal tilfældige bitfejl i afsendte beskeder simulerede dårlig radiodækning, og for at vide hvordan det testede udstyr (EUT) tolkede de afsendte beskeder, skulle programmet også simulere EUT-printer, dvs. modtage og tolke det EUT'en udskrev, hvilket der kom en 75-siders tyk rapport ud af, hvoraf et par sider kan ses fra side 9.

Alt i alt en udfordrende men yderst interessant programmeringsopgave.

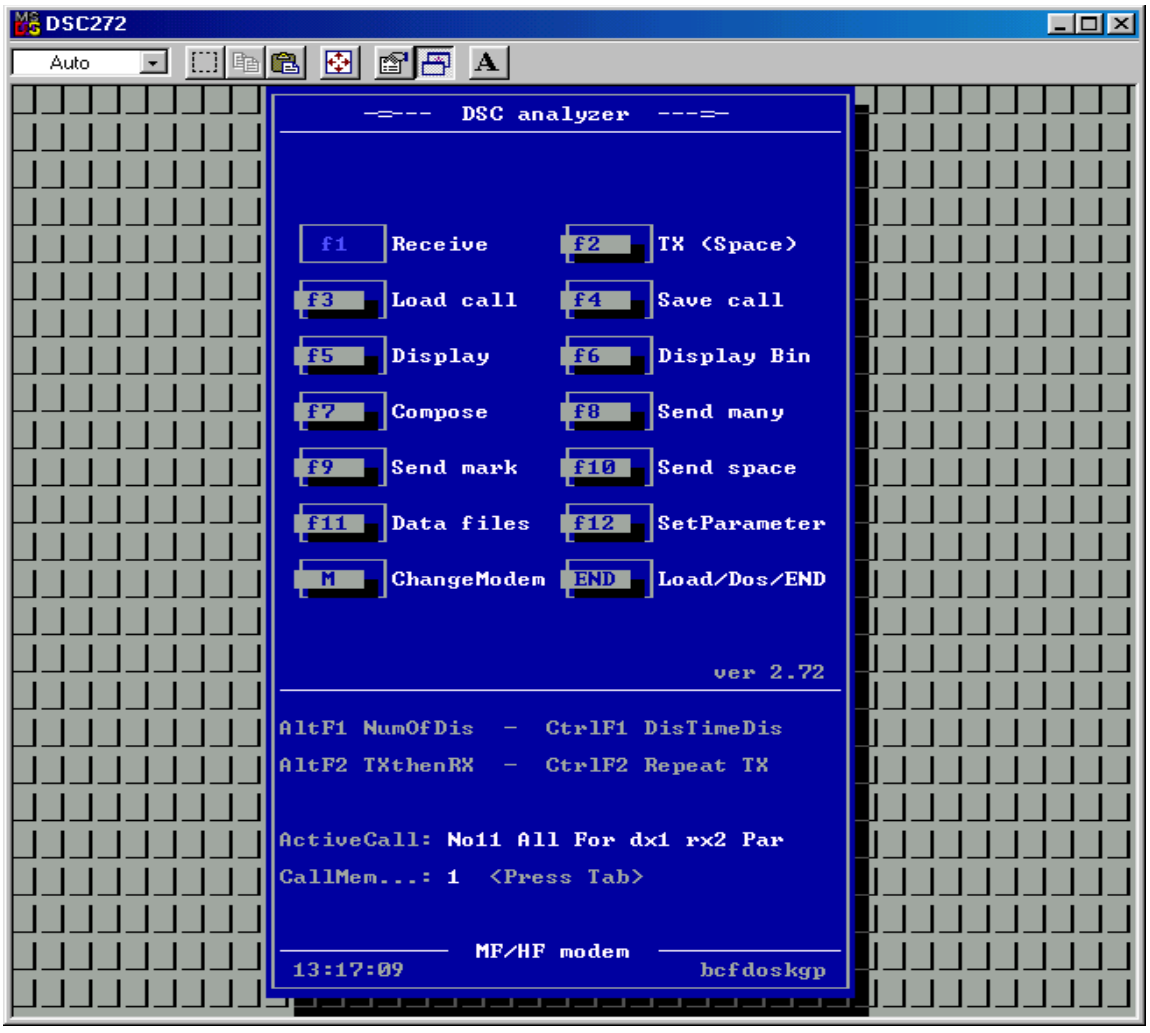


Fig. 7

F1, Receive: Programmet går i RX mode og venter på data tilført RS232 CTS (ben 8 & 5, hvis DB-9).

F2, TX: Programmet sender de data der er i hukommelsen ud på RS232 DTR (ben 4 & 5, hvis DB-9.

F3, Load call: Henter data fra fil.

F4, Save call: Gemmer data til fil.

F5, Display: Viser de data der er hentet via CTS eller fra fil. Data vises dekodet i klar tekst, og som de er modtaget enten i decimalt eller hexadecimalt talsystem. Det er muligt at editere data. Se Fig.8, og beskrivelsen side 7.

F6, Display Bin: Som 'F5' men data vises binært.

F7, Compose: Komponer en besked.

F8, Send many: Sender et valgt antal kald fra fil, med valgt pausetid imellem.

F9, Send mark: Sætter COM1 DTR høj, og starter en tone på 1615Hz (MF/HF) / 1300Hz (VHF) i PC'ens højttaler.

F10, Send space: Sætter COM1 DTR lav, og starter en tone på 1785Hz (MF/HF) / 2100Hz (VHF) i PC'ens højttaler.

F11, Data files: Kopierer og opretter data filer.

F12, SetParameter: Her defineres vigtige programparametre, bl.a. stien til data filerne.

M, ChangeModem: Vælg mellem MF/HF og VHF-mode.

END, Load/Dos/END: Tast to gange på 'End' tasten for at afslutte programmet.

<Alt>F1, NumOfDis: Tæller antallet af 'distress' gentagelser. (Krav fra standard).

<Ctrl>F1, DisTimeDis: Måler tiden der går mellem hvert distress kald. (Krav fra standard).

<Alt>F2, TxthenRX: Sender data og går derefter i receive mode. (Nyttigt for test af kaldtyper der udløser reply).

<Ctrl>F2, Repeat TX: Sender samme kald et antal gange.

F3, Load call

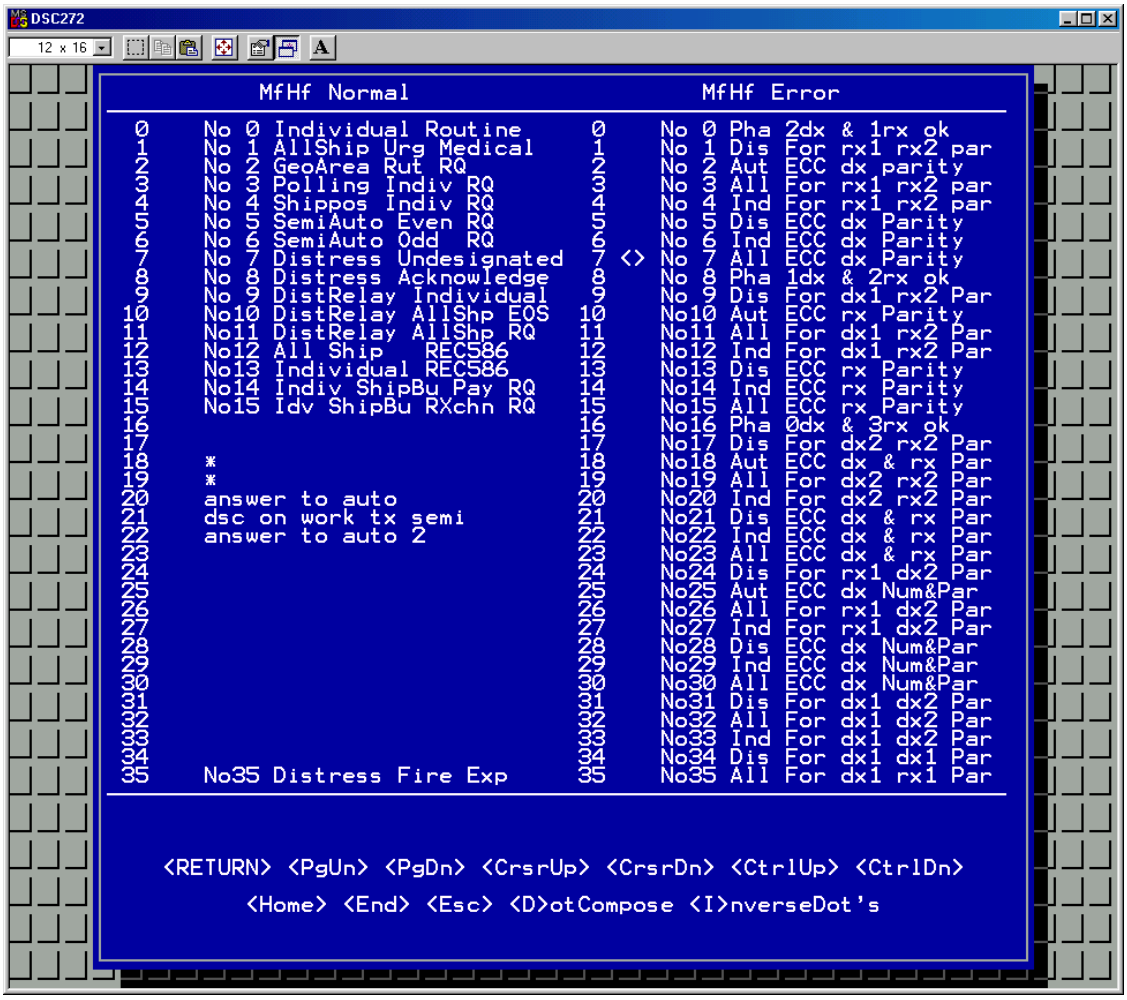


Fig.8. Eksempel på menuvalg '**F3, Load call**' i MF/HF mode.

Venstre kolonne 'MF/HF Normal' ('VHF Normal') indeholder fejlfri DSC-kald.

Højre kolonne 'MF/HF Error' ('VHF Error') indeholder DSC-kald med bitfejl tilføjet mere eller mindre kritiske steder i bitstrømmen, med henblik på at teste en DSC-dekoders reaktion på den fejlbehæftet bitstrøm. (Krav fra standard).

Markøren '<>' peger på kald no. 7, der er et 'All ships call' med påført paritetsfejl i ECC karakterens DX del. Se Fig.9.

Der skiftes mellem kolonnerne med højre/venstre piletast.

PS! Dette 'pseudofilsystem' er designet for at kunne bruge lange filnavne (der ikke var opfundet da programmet blev lavet), og arbejder med data fra følgende fire fysiske filer: MFNORM.RAN, MFERROR.RAN, VHFNORM.RAN, samt VHFERROR.RAN. Placeringen af disse filer skal fremgå af 'File path' under '**F12, SetParameter**'. 'File path' er default det samme katalog som programmet ligger i / starter fra.

F5, Display

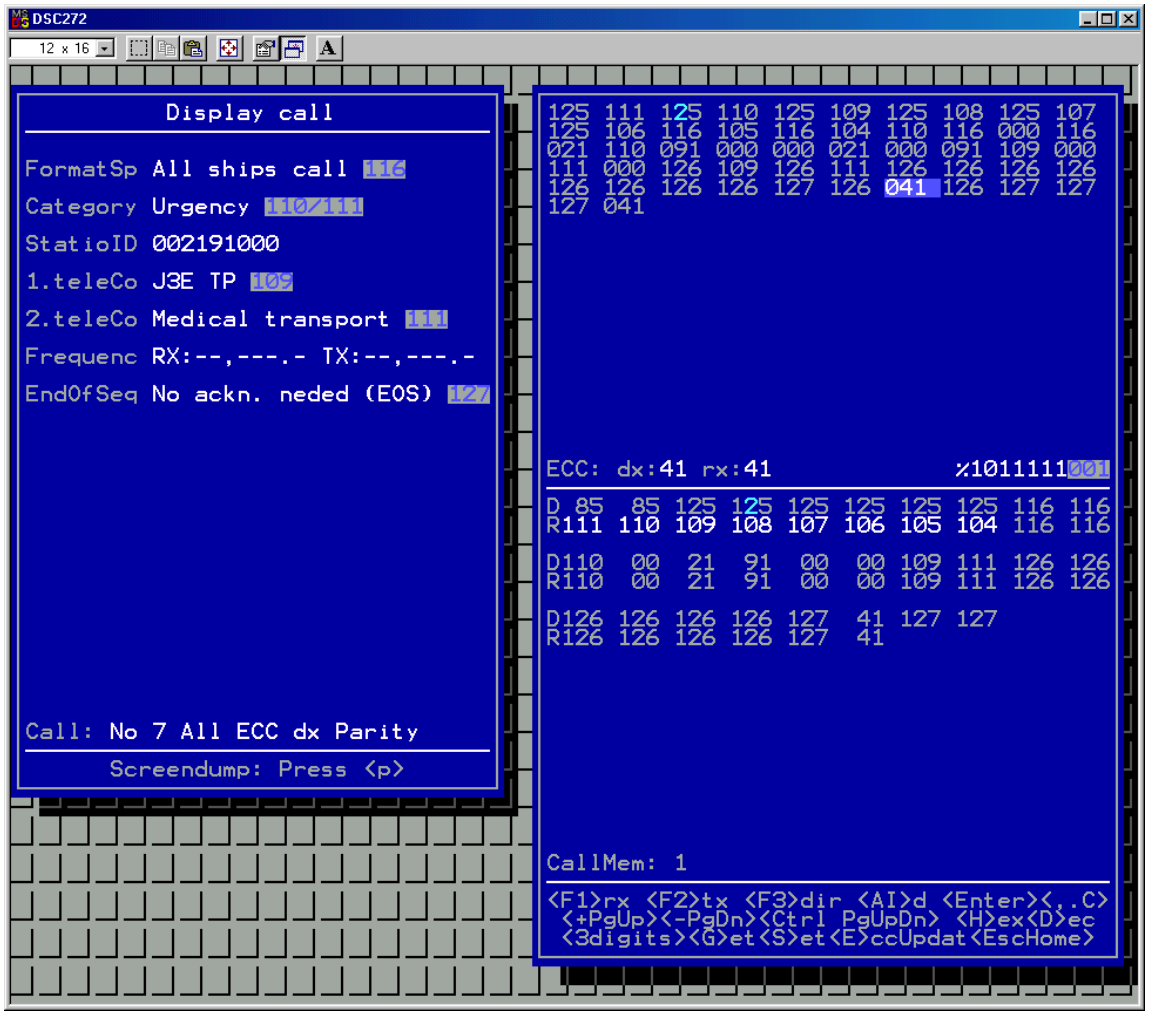


Fig.9. Eksempel på menuvalg '**F5, Display**'.

Venstre vindue: Her vises data i klar tekst, 'dekodet' i overensstemmelse med standarden.

Højre vindue, øverste del: Her vises data i den rækkefølge de er modtaget, i decimalt eller hexadecimalt talsystem. Hvert tal repræsenterer 10 bit (7 databit + 3 paritetsbit). Er der uoverensstemmelse mellem databit og paritetsbit (paritetsfejl), markeres dette som highlight af det berørte tal (her **041**).

%1011111*001* viser det markerede tal**1* på binær form**2*. Ved at trykke <Enter> kan det binære tal editeres. Ved fornyet tryk på <Enter>, opdateres det markerede tal, samt dekodningen i venstre vindue.

Indtastes der et trecifret tal, f.eks. '012' (uden brug af <Enter>), opdateres det markerede tal (og dekodningen i venstre vindue) direkte. (Tocifret tal, hvis i 'hex' mode).

1* Markeringen vises som en cyan-farvning af midterste tal (her 2-tallet i tredjeførste **125).

2* De syv databit (1011111**) er vist bagfra (mindst betydende bit først) da de, af grunde jeg har glemt, fysisk sendes på denne måde 'i luften'. De tre paritetsbit (*001*) angiver antallet af '0'-er i databit-delen.

ECC: dx:41 rx:41 viser resultatet af en modulo2 addition af alle data, på nær selve ECC-karakteren (som beskrevet i standarden).

Højre vindue, nederste del: Her vises data, parret DX/RX.

Er der uoverensstemmelse mellem DX og RX, markeres dette som en farveændring af RX-delen.

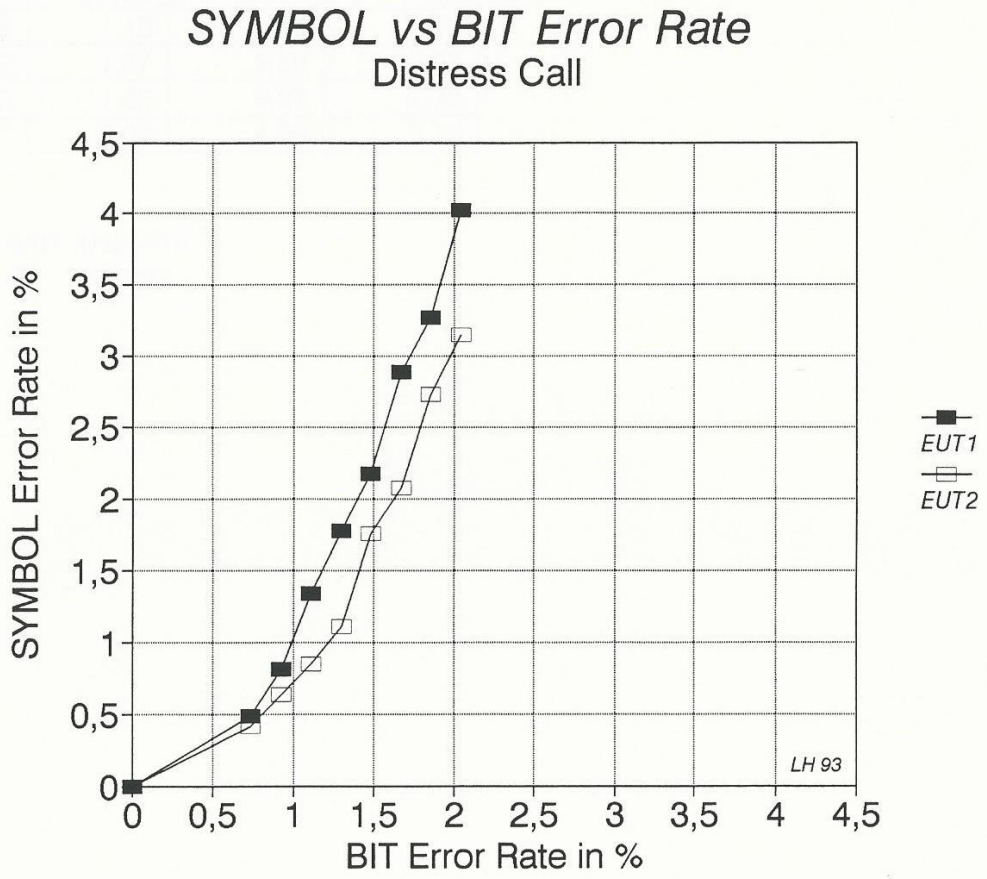
NB! Data vil blive sendt (**F2, TX**) og gemt (**F4, Save call**) som de vises under dette menupunkt.

Test Report of correlation between SER og BER

By Lars Hansen

Ref. No.
Client
Contact person

Date and signature 7/1-94


SYMBOL vs BIT Error Rate
Distress Call
SYMBOL Error Rate in %
BIT Error Rate in %
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
LH 93
EUT1
EUT2

Call type: Distress

No of BitError	BER	SER-EUT1 SER	SER-EUT2 SER
0	0,00	0,00	0,00
4	0,74	0,49	0,41
5	0,93	0,81	0,64
6	1,11	1,34	0,85
7	1,30	1,78	1,11
8	1,48	2,18	1,76
9	1,67	2,89	2,07
10	1,85	3,27	2,73
11	2,04	4,02	3,15

EUT1 =
EUT2 =

Example of a call supplied with errors, indicating the bit error position

FormatSp Individual stations 120

Address. 123456789

Category Routine 100

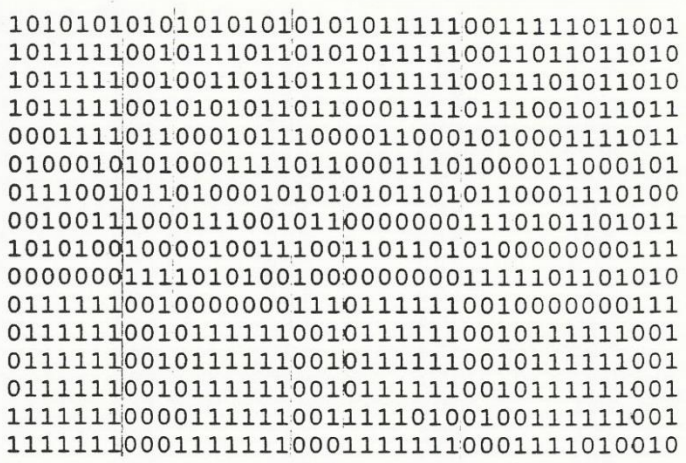
StatioID 002191000

1.teleCo No information 126

2.teleCo No information 126

Frequenc RX:--,---.- TX:--,---.-

EndOfSeq No ackn. neded (EOS) 127



Call: Init call

1993-11-03 11:41:23 EUT1

40*16=640 bits with no errors.

CallNo, ReceivedChars, NoOfErr, Bit position with error.

Init call	640	5	21	0,78			
1	21	0	472	79	191	119	136
2	21	0	475	381	243	218	320
3	21	0	182	306	393	399	544
4	21	0	210	613	348	553	3
5	21	0	538	598	416	58	158
6	21	0	245	522	566	239	381
7	21	0	618	376	209	445	468
8	21	0	305	361	277	589	622
9	21	0	231	623	357	431	92
10	21	0	73	124	137	480	447
11	21	0	101	431	93	635	547
12	21	0	449	356	243	176	130
13	21	0	136	341	310	320	284
14	21	0	509	195	594	526	371
15	21	0	376	280	375	117	424
16	21	0	200	557	483	160	121
17	21	1	572	411	126	366	208
18	21	0	228	224	438	315	221
19	21	1	575	149	588	496	444
20	21	0	633	325	247	107	501
21	21	0	558	587	328	589	611
22	21	1	471	541	325	241	597
23	21	1	618	447	161	215	603
24	21	0	11	54	327	442	156
25	21	0	384	548	610	8	243
26	21	1	161	263	215	515	346
27	21	1	309	170	50	490	353
28	21	0	277	616	172	378	460
29	21	0	74	271	499	282	633
30	21	0	447	125	143	488	80
31	21	0	224	480	387	355	183
32	21	0	105	240	507	535	277
33	21	0	362	434	479	353	551
34	21	0	600	50	370	135	116
35	21	0	333	543	13	340	203
36	21	0	98	4	462	132	483
37	21	0	515	359	67	640	586
38	21	0	396	120	186	180	40
39	21	0	161	221	635	612	320
40	21	1	579	576	240	480	424
41	21	0	86	482	76	454	430
42	21	0	459	336	359	20	517
43	21	0	236	51	604	528	621
44	21	0	477	65	439	355	172
45	21	0	358	466	558	535	265
46	21	0	615	20	531	353	539
47	21	0	213	275	421	134	104
48	21	0	585	129	65	340	191
49	21	0	363	484	309	208	295
50	21	0	575	177	321	363	552
51	21	0	533	632	280	317	555
52	21	0	310	348	524	185	18
53	21	0	639	394	494	203	390
54	21	0	596	209	452	157	392
55	21	0	521	471	533	639	501
56	21	0	94	164	544	155	119

Individual call.

CallNo, ReceivedChars, NoOfErr, Bit position with error.

347	21	0	379	542	512	550	247
348	21	0	156	257	116	418	350
349	21	0	529	111	400	624	437
350	21	0	306	466	4	491	541
351	21	0	352	367	385	111	384
352	21	0	592	328	573	463	274
353	21	0	157	389	585	133	587
354	21	0	415	584	558	591	221
355	21	0	147	437	201	157	308
356	21	0	285	353	514	197	209
357	21	0	5	592	362	327	473
358	21	0	211	14	374	637	145
359	21	0	496	476	523	11	52
360	21	0	68	169	534	167	310
361	21	0	126	324	71	502	616
362	21	0	544	39	316	370	79
363	21	0	276	533	599	575	167
364	21	0	534	87	572	393	440
365	21	0	266	581	215	599	528
366	21	0	2	133	541	336	384
367	21	0	169	597	337	431	69
368	21	0	542	451	620	637	156
369	21	1	107	512	633	306	468
370	21	0	65	349	74	176	221
371	21	0	232	174	510	271	546
372	21	0	483	104	98	2	151
373	21	0	170	89	166	146	305
374	21	0	238	213	383	500	326
375	21	0	546	320	271	482	628
376	21	0	233	305	338	626	142
377	21	0	301	430	556	340	163
378	21	0	91	460	498	157	307
379	21	0	239	345	212	215	563
380	21	0	566	330	280	360	77
381	21	0	97	500	372	385	87
382	21	0	424	485	440	530	242
383	21	0	157	339	83	95	329
384	21	1	504	263	233	276	552
385	21	0	202	240	464	355	171
386	21	0	70	325	246	586	224
387	21	0	442	179	529	151	312
388	21	0	130	164	597	296	466
389	21	0	342	497	608	452	83
390	21	0	570	142	458	215	574
391	21	0	258	128	526	360	88
392	21	1	470	460	537	515	345
393	21	1	348	537	482	40	503
394	21	0	406	73	141	292	560
395	21	0	633	359	631	55	411
396	21	0	321	344	58	200	565
397	21	0	533	37	70	355	182
398	21	0	401	122	492	586	235
399	21	0	133	616	135	152	322
400	21	0	461	601	203	296	477