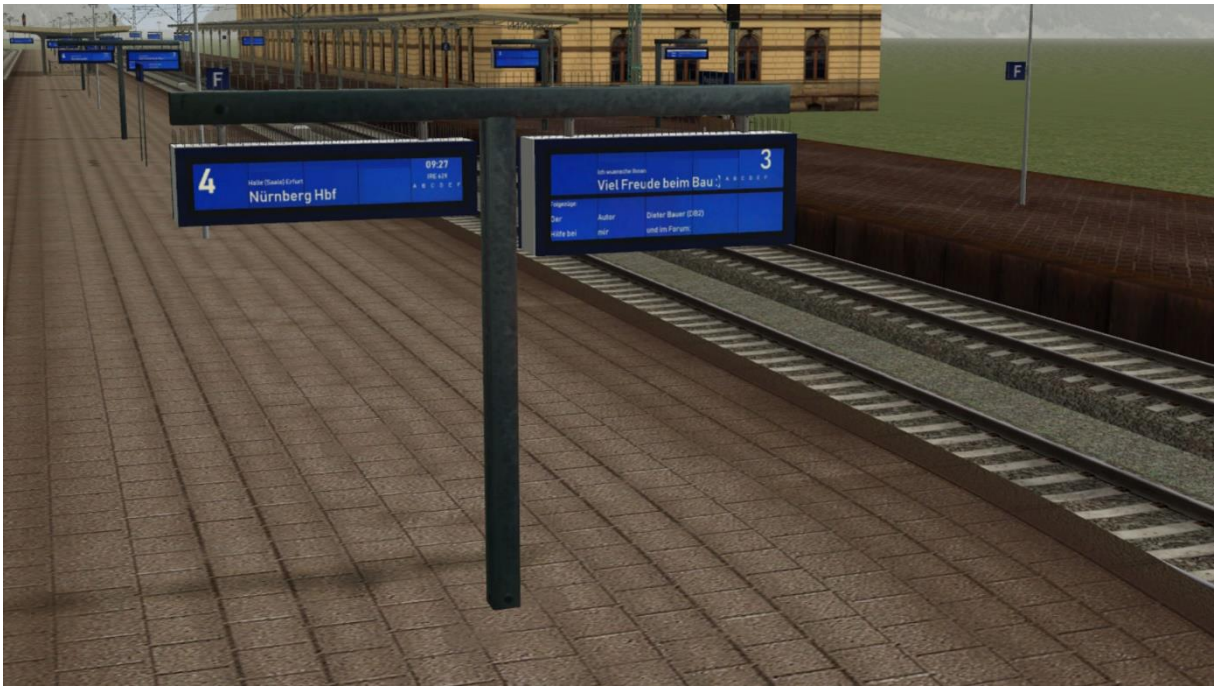
Documentation



Train destination indicator epoch VI of DB2

(V15NDB20015)

https://db-eep.de

Content

1 Model overview

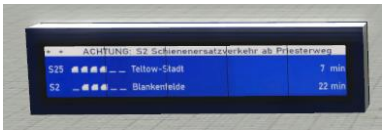
The set train destination displays epoch VI consists of the following models:

In normal font is the name as it is found in EEP.

The name of the file is written in small letters.

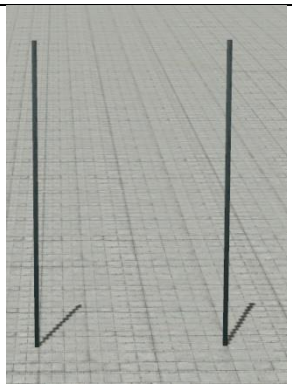
1.1 Train destination displays

	Real estate:	In Real Estate / Traffic / Platform Accessories
Train destination displays with 2 x 5 LCD displays **With labelling function** **and ticker (can be switched off)**	• Chariot run • Dimensions without holder: 188 x 77 x 44 cm • Default installation height: 2.40 m • Holder left and right individually extendable • Length of holder max. 74 cm	
	Train Dest. Ind. A, Era VI DB2 *ZZA6ADB2.3dm*	Typ A: • 2 lines of waypoints • with bird protection
	Train Dest. Ind. B, Era VI DB2 *ZZA6BDB2.3dm*	Typ B: • 2 lines of waypoints • without bird protection
	Train Dest. Ind. C, Era VI DB2 *ZZA6CDB2.3dm*	Typ C: • Display of follow-on trains • with bird protection
	Train Dest. Ind. D, Era VI DB2 *ZZA6DDB2.3dm*	Typ D: • Display of follow-on trains • without bird protection

Train destination displays with 1 x 5 LCD displays **With labelling function** **and ticker (can be switched off)**	• Dimensions without holder: 188 x 48 x 44 cm • Default installation height: 2.69 m • Holder left and right individually extendable • Length of holder max. 46 cm	
	Train Dest. Ind. E, Era VI DB2 *ZZA6EDB2.3dm*	Typ E: • Intermediate destinations • Chariot run • with bird protection
	Train Dest. Ind. F, Era VI DB2 *ZZA6FDB2.3dm*	Typ F: • Intermediate destinations • Chariot run • without bird protection
	Train Dest. Ind. G, Era VI DB2 *ZZA6GDB2.3dm*	Typ G: • two destinations (S-Bahn) • Chariot run • with bird protection
	Train Dest. Ind. H, Era VI DB2 *ZZA6HDB2.3dm*	Typ H: • two destinations (S-Bahn) • Chariot run • without bird protection
Poles for train destination indicators		Height: 3.67 m (+ 50 cm under ground) Mast thickness: 20 x 20 cm: 8 cm suitable for train destination indicator with 188 cm width
	Train Dest. mast 1-sided DB2 *ZZAT1DB2.3dm* Train Dest. Mast B 1-sided DB2 *ZZAT1DB2.3dm*	One-sided mast for train destination indicator width: 2,21 m, with bird protection without bird protection
	Train Dest. mast 2-sided DB2 *ZZAT2DB2.3dm* Train Dest. Mast B 2-sided DB2 *ZZAT2DB2.3dm*	Two-sided mast for train destination indicator width: 4,16 m without bird protection

1.2 Station signs

Station signs (station boards) - freely inscribable	Default installation height: 2.40, Thickness: 8 cm (variable by scaling) Epoch VI
	Station sign 100x40, Ep VI DB2 *SA6_7DB2.3dm*
	Station sign 150x40, Ep VI DB2 *SA6_6DB2.3dm*
	Station sign 120x50, Ep VI DB2 *SA6_1DB2.3dm*
	Station sign 150x50, Ep VI DB2 *SA6_2DB2.3dm*
	Station sign 200x50, Ep VI DB2 *SA6_3DB2.3dm*
	Station sign 250x50, Ep VI DB2 *SA6_4DB2.3dm*
	Station sign 300x60, Ep VI DB2 *SA6_5DB2.3dm*
	Station sign 40x40, Ep VI DB2 For track sections or track numbers *SAATDB2.3dm*

Stand beam for Station signs (and other signs)	Length 2.60 m plus 50 cm under the floor, thickness: 4 x 4 cm Galvanized steel
	Station sign stand 80cm gap DB2 *SA6T1DB2.3dm*
	Station sign stand 120cm gap DB2 *SA6T2DB2.3dm*
	Station sign stand 160cm gap DB2 *SA6T3DB2.3dm*
	Station sign stand 210cm gap DB2 *SA6T4DB2.3dm*

1.3 Fonts and other files

	File	In the folder Resourcen/Fonts
	BahnsymboleVI.ttf	Railway symbols Ep. VI *Die Grafik zeigt nur einen Ausschnitt aller in der Schrift enthaltenen Symbole*
	Bahnschrift.ttf	Bahnschrift
	Blocks	**In the real estate sector**
	ZZA_A_Mast_eins.bl3	
	ZZA_A_Mast_zweis.bl3	
	Demonstration unit	**In the folder Units/ZZA Demo**
	ZZA-Demo.anl3	
	Demovideo:	YouTube

2 Preface

These instructions describe the structure and functions of the models in as much detail as possible in order to give EEP beginners the opportunity to use the various possibilities of these new train destination displays as quickly as possible. Experienced users will surely find some paragraphs that don't convey anything new to them. They may, however, overlook the thoroughness of the description so that newcomers and inexperienced EEP´ler can have more fun with EEP.

If nevertheless something is unclear, I am attainable over the channels standing at the end of the documentation and stand gladly speech and answer.

3 Assembly

3.1 Train destination displays

The train destination displays can be suspended from the masts supplied for this purpose as well as from ceiling constructions, e.g. platform roofs.

The height of the lower edge is 2.40 m above the ground for the two-line displays and 2.69 m above the ground for the single-line displays. When using the supplied masts for suspension, this corresponds exactly to the correct height for suspension if the mast is used at the same position. The lateral offset of the indicators is 2.08 m, on the longitudinal axis the same value is to be selected as for the mast.

Of course, these values only apply to right-angled installation. Two blocks consisting of mast and indicator(s) are included to simplify installation in other angles. After setting up the block, the display can be exchanged in the 3D editor. For this purpose one selects the desired display in the model list, clicks with the right mouse button on the display, which is to be exchanged and then selects the point "Exchange against ....". The displays of this set are all compatible with each other, so that even the smaller displays are automatically inserted at the correct position.

Inside the displays there are two holders which can be extended to the desired length as required. To do this, select the left and then the right holder in the object properties and move it to the desired height using the slider. When suspended from the masts, these rods are placed exactly in the mounting plate, giving the impression that the displays are screwed to the mast.

3.2 Station signs

The station signs are also fitted with a lower edge of 2.40 m as standard. Matching stands are included in the set and are set with the same coordinates as the signs.

Of course the signs can also be adjusted in size. In the demo video, for example, a sign sits enthroned above a station bench and completely conceals the existing sign in the original bench. This was achieved by tripling the strength (Y axis) and slightly scaling the Z axis (1.02). The sign can also be attached to a station wall.

3.3 Track sections/track numbers

The signs 40 x 40 cm have a stable lateral holder, which can be mounted with 2 screws on lantern poles, platform roof supports or walls. The desired letter or number can be entered via the labelling function.

4 Labeling

4.1 Station signs

The labeling of the labels is very simple: Since EEP 15, the "Label" button has been included in the object properties. If this button is clicked, another window opens in which the desired text is entered in the lower area.

Further adjustments can be made if necessary: The font can be changed, also the font size and the representation (bold, light, italic). The check mark at "centered" changes the alignment of the text within the field specified by the designer.

Finally, the font and background color can be changed. Please note that the background color only affects the letters that are actually written. If you want to write a blue text on a white background, the area is only slightly higher white than the letter. If you want the white bar to continue on the right and/or left (as in the ticker texts of the train destination displays), you can fill the desired width with a corresponding amount of spaces. This requires some patience, as you first have to find out how many spaces are optimal.

4.2 Train destination displays

The train destination displays have a larger number of text fields. Some of them are filled in directly during insertion, others by Lua programming (see below).

4.2.1 Types A and B

The displays contain the following text fields:

Text No.	Field content front side	Text No.	Field content Back side
1	Train destination	2	Train destination
3	Interim destination 1	4	Interim destination 1
5	Interim destination 2	6	Interim destination 2
7	Time	8	Time
9	Train no.	10	Train no.
11	Chariot run	12	Chariot run
13	Platform sections	14	Platform sections
15	Track no.	16	Track no.
17	Information text (ticker)	Displayed on both sides	

For technical and optical reasons, the entries must be made on both sides. Fields 13 - 16 should be statically inscribed immediately. Information on the platform sections is described below under Railway symbols Era VI.

4.2.2 Types C and D

The displays contain more text fields, as the next two moves will also be announced:

Text No.	Field content front side	Text No.	Field content Back side
Entries for the main train			
1	Train destination	2	Train destination
3	Interim destination	4	Interim destination
5	Time	6	Time
7	Train no.	9	Train no.
9	Chariot run	10	Chariot run
Entries for the 1st follow-up train			
11	Time Follow-up train 1	12	Time Follow-up train 1
13	Train no. Follow-up train 1	14	Train no. Follow-up train 1
15	Destination Follow-up train 1	16	Destination Follow-up train 1
17	Information Follow-up train 1	18	Information Follow-up train 1
	(here a short text in blue on white is possible)		
Entries for the 2. Follow-up train			
19	Time Follow-up train 2	20	Time Follow-up train 2
21	Train no. Follow-up train 2	22	Train no. Follow-up train 2
23	Destination Follow-up train 2	24	Destination Follow-up train 2
25	Information Follow-up train 2	26	Information Follow-up train 2
General static information			
27	Platform sections	28	Platform sections
29	Track no.	30	Track no.
31	Heading „Follow-up trains“	32	Heading „Follow-up trains“
For technical reasons, the ticker with the general passenger information always appears at the end of the entries.			
33	Information text (ticker)	Displayed on both sides	

4.2.3 Types E and F

This display is as functional as the types A and B, but has only one LCD row and therefore fewer fields.

Text No.	Field content front side	Text No.	Field content Back side
1	Train destination	2	Train destination
3	Interim destination 1	4	Interim destination 1
5	Time	6	Time
7	Train no.	8	Train no.
9	Chariot run	10	Chariot run
11	Platform sections	12	Platform sections
13	Track no.	14	Track no.
15	Information text (ticker)	Displayed on both sides	

4.2.4 Types G and H

The display has a fundamentally different structure than the previous ones, which is why the displays do not have to be labeled separately on the front and rear sides.

Text No.	Feldinhalt erste Zeile	Text No.	Feldinhalt zweite Zeile
1	Line 1. train	5	Line 2. train
2	Chariot run 1. train	6	Chariot run 2. train
3	Destination 1. train	7	Destination 2. train
4	Departure 1. train	8	Departure 2. train
Statische Angaben			
9	Text “min” 1. train	10	Text “min” 2. train
Und wieder zum Schluss:			
11	Information text (ticker)		

5 Display control

Now it comes to the actually interesting part. What's special about the previous Train train destination displays is that they can be labeled at will - thanks to the new labeling function in EEP 15.

However, this only makes sense if the Lua script language contained in EEP is used. But don't worry, you don't have to be a programmer, not even an Ace of Math to control the displays. In the simplest case it is simply enough to copy the appropriate text below (program code), which is also included in the demo system, and enter your own goals and times. The code is specially written so that you don't have to puzzle where something is entered, but the entries are quite obvious.

If you have programming knowledge or experience with Lua, you will be able to optimize the code and extend it with further functions. But also less experienced people will be able to implement some additional functions with a little training.

The manual does not follow the logical order of types A, B, C ... H, but starts with the simplest display.

5.1 Basic entries in the Lua file

First of all I would like to recommend everyone to install the following free of charge:

1. notepad++ - editor
   In my opinion the safest and easiest download in German language is at Heise.de:
   https://www.heise.de/download/product/notepad-26659

2. the Lua command extension especially for EEP, download with instructions in the forum:
   https://www.eepforum.de/filebase/file/23-lua-xml-f%C3%BCr-notepad/

The colour highlights and indentations as well as the larger text editor make the code much clearer. The line numbers are useful if, for example, an error message is ejected due to a typing error, in which EEP specifies the number of the relevant line. Without Notepad++ you may be able to search for a very long time.

The program lines refer to the demo installation and can be traced there.

Green are comments. They begin with two hyphens:

```
-- The following line of code was created by Benjamin Hoge (BH2 - emaps-eep.de),
-- that parameters can be passed in the contact points when calling a Lua function.
-- In this example this is used to control the displays on track 3
```

```
setmetatable(_ENV,{__index=function(s,k) local n=k:gsub("%.",","); 
local p=load(n);if p then local f=function(z) local 
s=Zugname;Zugname=z;p();Zugname=s end;_ENV[k]=f;_ENV[n]=f;return f 
end;return nil end})
```

The program code is highlighted in gray - in Notepad++ it is divided into different colors, which is helpful for the basic understanding, but not so important. After all, this is not supposed to be a programming course ;) Only the function calls are in

dark blue, the transfer parameters are not printed bold.

5.2 Type E and F

-- ----------

-- The following block controls the displays on track 4 ----------

The following texts are to be changed automatically:

-- 1 Train destination frontside
-- 2 Train destination backside
-- 3 Interim destination frontside
-- 4 Interim destination backside
-- 5 Time frontside
-- 6 Time backside
-- 7 Train no. frontside
-- 8 Train no. backside

The other labels are either already set in the object properties or are not required in this example.

First comes a call function. This determines which values are to be entered for each move.

```
function IRE629_MD04()  -- IRE 629 travels in Magdeburg from platform 4.
```

-- This somewhat cumbersome name defines a unique function that does not (must not) occur a second time.

```
    ziel_ZZA_MD04 = "Nürnberg Hbf"                -- The destination of the journey
    zwziel_ZZA_MD04 = "Halle (Saale) Erfurt"      -- selected Interim destination
    zeit_ZZA_MD04 = "09:27"                       -- Time of departure
    zugnr_ZZA_MD04 = "IRE 629"                    -- Name of the train
```

-- Now these values are transferred to the actual function that makes the entries:

```
    ZZA_MD04(ziel_ZZA_MD04, zwziel_ZZA_MD04, zeit_ZZA_MD04,
zugnr_ZZA_MD04)
end                          -- each function must be terminated with an "end“.
```

-- (The ZZA for further trains are controlled according to the following block)

-- The "detour" quickly becomes clear when you look at the length of the following entry function. If the entry was made directly, this long block would have to be written each time, unnecessarily inflating the program code.

```
function ZZA_MD04(ziel_ZZA_MD04, zwziel_ZZA_MD04, zeit_ZZA_MD04,
zugnr_ZZA_MD04)
```

-- The variables in brackets give this function the values to be entered.

-- Now the first ZZA is filled with the correct text: The number after the hash is located in the object properties under "Lua Name". Only the number is needed, the rest is not important.

```
EEPStructureSetTextureText( "#345" , 1 , ziel_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 2 , ziel_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 3 , zwziel_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 4 , zwziel_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 5 , zeit_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 6 , zeit_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 7 , zugnr_ZZA_MD04 )
EEPStructureSetTextureText( "#345" , 8 , zugnr_ZZA_MD04 )
```

-- The second ZZA has the number 342

```
EEPStructureSetTextureText( "#342" , 1 , ziel_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 2 , ziel_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 3 , zwziel_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 4 , zwziel_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 5 , zeit_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 6 , zeit_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 7 , zugnr_ZZA_MD04 )
EEPStructureSetTextureText( "#342" , 8 , zugnr_ZZA_MD04 )
```

-- The remaining target displays are filled in the same way. In this case there are six more displays. These can be read completely in the Lua file of the system.

-- The experienced Lua programmer can shorten this with a loop still somewhat.

```
end
```

--As above the first IRE the following trains will be announced. Try to understand what is important for you:

```
function IRE627_MD04()
```

```
ziel_ZZA_MD04 = "Dresden Hbf"
zwziel_ZZA_MD04 = "Dessau Leipzig"
zeit_ZZA_MD04 = "09:32"
```

```
zugnr_ZZA_MD04 = "IRE 627"
ZZA_MD04(ziel_ZZA_MD04, zwziel_ZZA_MD04, zeit_ZZA_MD04,
zugnr_ZZA_MD04)
```

end

-- I am convinced that you now know that you only have to enter the few data at target, target, time and turnnr. The rest can simply be copied and tagged with one of the corresponding track and train numbers.

--You can find the remaining trains in the demo facility

-- End of program block Display track 4 ----------

-- ----------

5.3 Types A and B

-- ----------

-- The following block controls the displays on track 2 ----------

-- The block is similar to track 4, but it has more entries. The values of the ZZA in Magdeburg track 2 are entered in the following block. However, again only variables are used as "placeholders", which are then entered in the following functions by the individual trains.

In the case of track 2 I show how the entry is set by the train itself when it comes from the depot and crosses a contact point there.

If a train leaves the station, a contact point after the exit signal triggers the function "no display_MD02()", which empties the displayboard.

This type of programming only works if the exit from the depot, and thus the crossing of the contact point for display generation, is AFTER the departure of the previous train from the station.

```
function ZZA_MD02(ziel_ZZA_MD02, zwziel1_ZZA_MD02, zwziel2_ZZA_MD02,
zeit_ZZA_MD02, zugnr_ZZA_MD02, wagenlauf_ZZA_MD02)
```

```
EEPStructureSetTextureText( "#296" , 1 , ziel_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 2 , ziel_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 3 , zwziel1_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 4 , zwziel1_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 5 , zwziel2_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 6 , zwziel2_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 7 , zeit_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 8 , zeit_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 9 , zugnr_ZZA_MD02 )
```

```
EEPStructureSetTextureText( "#296" , 10 , zugnr_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 11 , wagenlauf_ZZA_MD02 )
EEPStructureSetTextureText( "#296" , 12 , wagenlauf_ZZA_MD02 )


EEPStructureSetTextureText( "#292" , 1 , ziel_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 2 , ziel_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 3 , zwziel1_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 4 , zwziel1_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 5 , zwziel2_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 6 , zwziel2_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 7 , zeit_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 8 , zeit_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 9 , zugnr_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 10 , zugnr_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 11 , wagenlauf_ZZA_MD02 )
EEPStructureSetTextureText( "#292" , 12 , wagenlauf_ZZA_MD02 )
```

-- the same for the other indicators on track 2.

end

-- The ICE 152 comes out of the depot at 9:19 controlled by a depot exit and drives via a vehicle contact which triggers this function:

function ICE152_MD02()

```
ziel_ZZA_MD02 = "Ostseebad Binz"
zwziel1_ZZA_MD02 = "Brandenburg Potsdam Berlin"
zwziel2_ZZA_MD02 = "Neubrandenburg Greifswald Stralsund"
zeit_ZZA_MD02 = "09:25"
zugnr_ZZA_MD02 = "ICE 152"
wagenlauf_ZZA_MD02 = "<222222s11>"
```

-- The list of symbols for the car run can be found below.

-- The following line transfers the values entered above to the above function ZZA_MD002, which labels the train destination indicators.

```
ZZA_MD02(ziel_ZZA_MD02, zwziel1_ZZA_MD02, zwziel2_ZZA_MD02,
zeit_ZZA_MD02, zugnr_ZZA_MD02, wagenlauf_ZZA_MD02)
```

end

-- In this way every train for track 2 will be entered here

```
function TEE36_MD02()
      ziel_ZZA_MD02 = "München Hbf"
      zwziel1_ZZA_MD02 = "Duisburg Köln Bonn"
      zwziel2_ZZA_MD02 = "Frankfurt(M) Würzburg Nürnberg"
      zeit_ZZA_MD02 = "09:34"
      zugnr_ZZA_MD02 = "TEE 36"
      wagenlauf_ZZA_MD02 = "<111111s11"
      ZZA_MD02(ziel_ZZA_MD02, zwziel1_ZZA_MD02, zwziel2_ZZA_MD02,
      zeit_ZZA_MD02, zugnr_ZZA_MD02, wagenlauf_ZZA_MD02)
end
-- etc. Until the function of the empty display comes at the end:

function keineAnzeige_MD02()
      ziel_ZZA_MD02 = ""
      zwziel1_ZZA_MD02 = ""
      zwziel2_ZZA_MD02 = ""
      zeit_ZZA_MD02 = ""
      zugnr_ZZA_MD02 = ""
      wagenlauf_ZZA_MD02 = ""
      ZZA_MD02(ziel_ZZA_MD02, zwziel1_ZZA_MD02, zwziel2_ZZA_MD02,
      zeit_ZZA_MD02, zugnr_ZZA_MD02, wagenlauf_ZZA_MD02)
end

-- End of program block Display track 2 ----------
-- ----------
```

5.4 Types C and D

Now it's getting a little more complicated. This means that it looks more complicated, but those who have managed with the previous advertisements will quickly understand this form.

In these displays, two follow-on trains will be added to the next moving train to improve passenger information. A "`if .. else`" loop is added, one of the simplest and most effective program routines ever.

```
-- ----------
The following block controls the displays on track 3 ----------

-- Basically, the call of this program routine differs from the previous ones in that it is triggered by
this train after leaving the station, except for the first train.
This means that the next train - together with the two following trains - is announced immediately.

First again the, now even more extensive, function for entering the values:
```

```
function ZZA_MD03(ziel0_ZZA_MD03, zwziel0_ZZA_MD03, zeit0_ZZA_MD03,
zugnr0_ZZA_MD03, wagenlauf0_ZZA_MD03, zeitZug1_ZZA_MD_03,
zugnrZug1_ZZA_MD_03, zielZug1_ZZA_MD_03, infoZug1_ZZA_MD_03,
zeitZug2_ZZA_MD_03, zugnrZug2_ZZA_MD_03, zielZug2_ZZA_MD_03,
infoZug2_ZZA_MD_03)
```

```
EEPStructureSetTextureText( "#347" , 1 , ziel0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 2 , ziel0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 3 , zwziel0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 4 , zwziel0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 5 , zeit0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 6 , zeit0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 7 , zugnr0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 8 , zugnr0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 9 , wagenlauf0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 10 , wagenlauf0_ZZA_MD03 )
EEPStructureSetTextureText( "#347" , 11 , zeitZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 12 , zeitZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 13 , zugnrZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 14 , zugnrZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 15 , zielZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 16 , zielZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 17 , infoZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 18 , infoZug1_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 19 , zeitZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 20 , zeitZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 21 , zugnrZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 22 , zugnrZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 23 , zielZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 24 , zielZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 25 , infoZug2_ZZA_MD_03 )
EEPStructureSetTextureText( "#347" , 26 , infoZug2_ZZA_MD_03 )
```

-- As usual, this block is copied and provided with the respective number of the display - or controlled by a loop.

```
end
```

-- Now follow the functions for determining the entries. If this function is called via the contact point, it is given the value 0. The Lua function call in the contact point (after the station) is "**ICE218_MD03(0)**". In the function, z assumes the passed value.

```
function ICE218_MD03(z)
```

-- First, the values for this move are transferred:

```
ziel_ZZA_MD03 = "München Hbf"
zwziel_ZZA_MD03 = "Leipzig Nürnberg"
zeit_ZZA_MD03 = "09:26"
zugnr_ZZA_MD03 = "ICE 218"
wagenlauf_ZZA_MD03 = "l22c1R"
```

```
if z == 0 then
```

-- the condition z == 0 is true, so the following lines are executed. The values from above are written to a variable with the addition 0.
OTHER: The double equal sign is not an error, but shows EEP that it should compare the value z currently has with the value 0. Otherwise the program might assume that we want to assign this value to the variable z, which is not the case.

```
ziel0_ZZA_MD03 = ziel_ZZA_MD03
zwziel0_ZZA_MD03 = zwziel_ZZA_MD03
zeit0_ZZA_MD03 = zeit_ZZA_MD03
zugnr0_ZZA_MD03 = zugnr_ZZA_MD03
wagenlauf0_ZZA_MD03 = wagenlauf_ZZA_MD03
```

-- Now the program routine of the first subsequent move is called. The value 1 is passed..

```
TEE103_MD03(1)
```

-- First the routine TEE103_MD03 is executed, so see what happens there.

-- The program now has the values of the main and first subsequent move in memory. So the second move is still missing. Call, entry and return of the values work the same as with the first subsequent move, only that this time the value 2 is passed and thus the second elseif is evaluated in the following function

```
IRE815_MD03(2)
```

-- So first to the „Subroutine“ …

-- We now have all the data we need for the display and pass it to the function for entering the values

```
ZZA_MD03(ziel0_ZZA_MD03, zwziel0_ZZA_MD03, zeit0_ZZA_MD03,
zugnr0_ZZA_MD03, wagenlauf0_ZZA_MD03, zeitZug1_ZZA_MD_03,
zugnrZug1_ZZA_MD_03, zielZug1_ZZA_MD_03, infoZug1_ZZA_MD_03,
zeitZug2_ZZA_MD_03, zugnrZug2_ZZA_MD_03, zielZug2_ZZA_MD_03,
infoZug2_ZZA_MD_03)
```

-- The following lines are now no longer read by EEP

```
        elseif z == 1 then
        zeitZug1_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug1_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug1_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug1_ZZA_MD_03 = ""
        return (zeitZug1_ZZA_MD_03), (zugnrZug1_ZZA_MD_03),
        (zielZug1_ZZA_MD_03), (infoZug1_ZZA_MD_03)
```

```
        elseif z == 2 then
        zeitZug2_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug2_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug2_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug2_ZZA_MD_03 = ""
        return (zeitZug2_ZZA_MD_03), (zugnrZug2_ZZA_MD_03),
        (zielZug2_ZZA_MD_03), (infoZug2_ZZA_MD_03)
    end
```

```
end
```

-- At the end, the other moves are entered in the same way, i.e. simply copy a block, add it below and adjust the data and calls.

→ to the next chapter

```
function TEE103_MD03(z)
```

-- First the values for this move are passed again:

```
        ziel_ZZA_MD03 = "Genf"
        zwziel_ZZA_MD03 = "Bonn Karlsruhe Basel"
        zeit_ZZA_MD03 = "09:31"
        zugnr_ZZA_MD03 = "TEE 103"
        wagenlauf_ZZA_MD03 = "1111s1>"
```

```
    if z==0 then
```

-- The condition z == 0 is false, which is why the next lines are not processed.

```
        ziel0_ZZA_MD03 = ziel_ZZA_MD03
        zwziel0_ZZA_MD03 = zwziel_ZZA_MD03
        zeit0_ZZA_MD03 = zeit_ZZA_MD03
        zugnr0_ZZA_MD03 = zugnr_ZZA_MD03
        wagenlauf0_ZZA_MD03 = wagenlauf_ZZA_MD03
```

```
        IRE815_MD03(1)
        RE67_MD03(2)
        ZZA_MD03(ziel0_ZZA_MD03, zwziel0_ZZA_MD03, zeit0_ZZA_MD03,
        zugnr0_ZZA_MD03, wagenlauf0_ZZA_MD03, zeitZug1_ZZA_MD_03,
        zugnrZug1_ZZA_MD_03, zielZug1_ZZA_MD_03, infoZug1_ZZA_MD_03,
        zeitZug2_ZZA_MD_03, zugnrZug2_ZZA_MD_03, zielZug2_ZZA_MD_03,
        infoZug2_ZZA_MD_03)
```

```
elseif z == 1 then
```

-- Now it is checked whether z = 1. Since this is the case with the first call, the following lines are processed. This time, the necessary values are written in variables with the addition 1...

```
        zeitZug1_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug1_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug1_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug1_ZZA_MD_03 = ""
```

-- ... and returned to the calling function (**ICE218_MD03(z)**) using `return`.

```
        return (zeitZug1_ZZA_MD_03), (zugnrZug1_ZZA_MD_03),
        (zielZug1_ZZA_MD_03), (infoZug1_ZZA_MD_03)
```

-- With `return` , this routine will be canceled. EEP don´t read the following lines and jumps back to the place, from whom the call came.

```
 elseif z == 2 then
        zeitZug2_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug2_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug2_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug2_ZZA_MD_03 = ""
        return (zeitZug2_ZZA_MD_03),
(zugnrZug2_ZZA_MD_03),(zielZug2_ZZA_MD_03), (infoZug2_ZZA_MD_03)
end
```

-- This `end` stands for the end of the `if..then` statement, i.e. the end of the loop. Otherwise, the program would search for further statements that do not exist.

```
end             -- This end terminates the current function.
```

```
function IRE815_MD03(z)
        ziel_ZZA_MD03 = "Würzburg Hbf"
        zwziel_ZZA_MD03 = "Nordhausen-Erfurt-Suhl"
        zeit_ZZA_MD03 = "09:35"
```

```
        zugnr_ZZA_MD03 = "IRE 815"
        wagenlauf_ZZA_MD03 = ""

    if z==0 then
-- This condition is false
        ziel0_ZZA_MD03 = ziel_ZZA_MD03
        zwziel0_ZZA_MD03 = zwziel_ZZA_MD03
        zeit0_ZZA_MD03 = zeit_ZZA_MD03
        zugnr0_ZZA_MD03 = zugnr_ZZA_MD03
        wagenlauf0_ZZA_MD03 = wagenlauf_ZZA_MD03
        RE67_MD03(1)
        TEE18_MD03(2)
        ZZA_MD03(ziel0_ZZA_MD03, zwziel0_ZZA_MD03, zeit0_ZZA_MD03,
        zugnr0_ZZA_MD03, wagenlauf0_ZZA_MD03, zeitZug1_ZZA_MD_03,
        zugnrZug1_ZZA_MD_03, zielZug1_ZZA_MD_03, infoZug1_ZZA_MD_03,
        zeitZug2_ZZA_MD_03, zugnrZug2_ZZA_MD_03, zielZug2_ZZA_MD_03,
        infoZug2_ZZA_MD_03)

    elseif z == 1 then
-- This condition is also wrong
        zeitZug1_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug1_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug1_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug1_ZZA_MD_03 = ""
        return (zeitZug1_ZZA_MD_03),
        (zugnrZug1_ZZA_MD_03),(zielZug1_ZZA_MD_03),
        (infoZug1_ZZA_MD_03)

    elseif z == 2 then
-- Finally a true result ;)
        zeitZug2_ZZA_MD_03 = zeit_ZZA_MD03
        zugnrZug2_ZZA_MD_03 = zugnr_ZZA_MD03
        zielZug2_ZZA_MD_03 = ziel_ZZA_MD03
        infoZug2_ZZA_MD_03 = ""
        return (zeitZug2_ZZA_MD_03), (zugnrZug2_ZZA_MD_03),
        (zielZug2_ZZA_MD_03), (infoZug2_ZZA_MD_03)
-- And back again
     end
end

-- End of program block Display track 3  ----------------------------------------------------------------

-- ------------------------------------------------------------------------------------------------------------
```

5.5 Types G and H

These displays are used especially for S-Bahn and U-Bahn trains. The programming is a bit more complex. If you get along with the first three models, you can use the Lua file of the demo system to control your own displays. The file itself contains explanations for this.

6 Symbol font Bahnsymbole Epoche VI

I developed the symbol font "Bahnsymbole Epoche VI" especially for use with train destination displays, but also for labelling other objects with symbols. It shows the platform sections and the car status.

- Each letter (from A to I) corresponds exactly to one wagon length.
- A space is exactly one quarter of a wagon's length.
- Exception: The length of the front or rear wagon is one letter plus one space.

In this way the platform sections can be adapted to the respective conditions. As a rule, the letters on the platforms are 53 metres = 2 wagons lengths apart, whereby the wagon length refers to the old IC trains (26.40 m per car).

6.1 Font „Bahnschrift“

The font "Bahnschrift" is a font that was developed by Microsoft and is included in current Windows-10 versions in Germany.

For older and other Windows versions the font is also included in the package.

6.2 Installing the Fonts

After installing the complete set with the EEP model installer, the fonts can be found in the EEP folder under Resources/Fonts. The installation is done by double-clicking on the respective font and then clicking on the "Install" button. If the font has already been installed, Windows will ask if the existing font should be overwritten. This query must be aborted with "No".

6.3 Symboltabelle

Wagenstände					Gleisabschnitte			
L		R			A	A	B	B
l		r			C	C	D	D
<		>			E	E	F	F
1		2			G	G	H	H
c		s			I	I		
f		h						
w		0						
S		T						
Weitere Symbole								
3		4			5		6	
7		8			9			
!		§			$		%	
&		/						

In the next weeks the font will be supplemented by several more symbols. On my website https://db-eep.de/Schriften you will find the corresponding updates.

I wish you lots of fun with this set ☺

Questions, suggestions or mistakes found?

It is best to set your request in the official EEP forum:

https://www.eepforum.de/forum/index.php?board/359-db2-dieter-bauer/

Legal Notice

The demonstration and removal system can not be passed on as part of this set. It can be used for demonstration and demonstration purposes of all kinds, also publicly and commercially, is expressly allowed in the original state as well as further developed and / or modified.

Source for the models:

One or more textures on this 3D-model have been created with images from Textures.com. These images may not be redistributed by default. Please visit www.textures.com for more information.

Some or several textures of these 3D models have been created using graphics "Designed by Freepik".

Author

Dieter Bauer

Frauenstraße 13

89537 Giengen an der Brenz
Germany

Web: https://db-eep.de

Email: dieter.bauer@db-eep.de