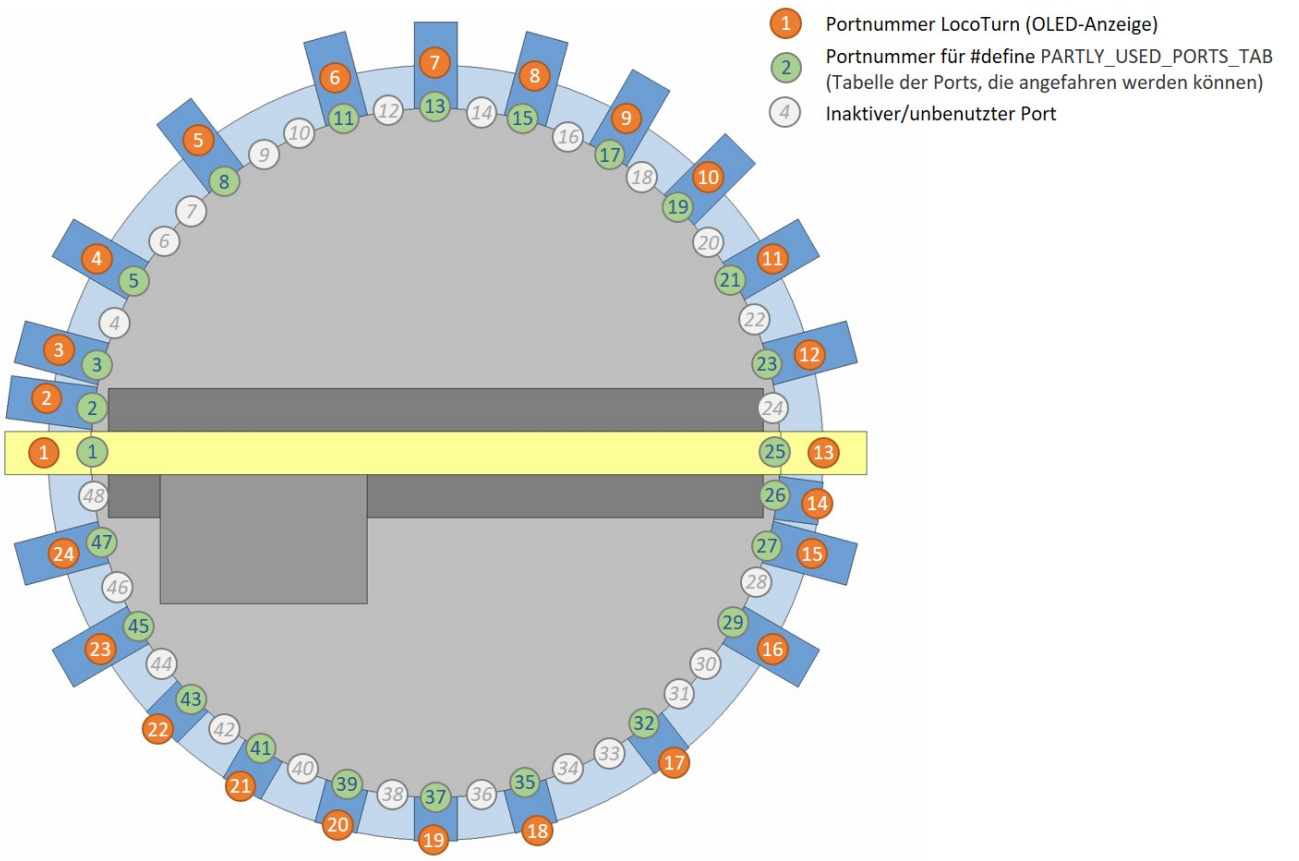
Parameter zur Einstellung der Drehscheibe

Aus der Vielzahl der Konfigurationsvariablen (#defines) hier ein paar wesentliche:

- Einige #defines wie **USE_DCC**, **USE_SOUND**, **USE_SERIAL_INPUT** etc. steuern, was der Sketch alles später an Funktionen bietet.
- Das #define **USE_POTI** muss auf 0 gesetzt werden, wenn kein Poti angeschlossen ist, sonst kann es sein, dass die Drehscheibe nach dem ersten Kalibrieren und Anfahren des ersten Ports „unkontrolliert herumschleicht“.
- Zentrale Parameter sind die Anzahl der benötigten Ports #define **PORT_CNT** bzw. #define **PARTLY_USED_PORTS** (falls nicht alle angefahren werden sollen) und #define **PARTLY_USED_PORTS_TAB** (Tabelle der Ports, die angefahren werden können).
- Die erste verwendete DCC Adresse ist einzugeben, wenn man von dem im Programm vorgegebenen DCC-Adressraum abweichen will: #define **FIRST_USED_DCC_ADDR**.
- Ebenso ist die letzte verwendete DCC Adresse zu definieren, wenn man nicht alle vorgesehenen DCC-Adressen benötigt und vom vorgegebenen DCC_Adressraum abweicht: #define **LAST_USED_DCC_ADDR**.
- Liste der DCC-Adressen für die Ports anpassen: #define **DCC_PORT_ADDR_LIST**. Die Liste muss **PORT_CNT**-Einträge enthalten. Wenn der Märklin-Modus (**DCC_MAERKLIN_7687_COMPATIBLE**) verwendet wird, reichen PORT_CNT / 2 Einträge.
- Das #define **ADVANCED_SIGNAL_CONTROL** = 1 steuert die Signale in Abhängigkeit, ob die DS bei Stillstand mit der jeweiligen Seite an einem aktiven/gültigen oder blinden Port steht. Ob es sich um einen aktiven oder inaktiven/blinden Port handelt, legt das #define **PORT_TYPE** fest.
- Zur Kalibrierung den DEBUG-Mode einschalten, damit im seriellen Monitor die Werte ausgelesen werden können: #define **ENABLE_DPRINTF** 1.
- Bei Bedarf Einstellungen der Ausrichtung bzw. Drehrichtung von Drehscheibe, Potentiometer, Dreh/Drückknopf und Display vornehmen.
- Ggf. Änderungen an den Einstellungen der verschiedenen Drehgeschwindigkeiten vornehmen.

Die Abbildung zeigt eine Drehscheibe mit 48 möglichen Ports, davon werden 24 genutzt:

- Die orange-farbigen Ports können später von LocoTurn angefahren werden, sie werden mit der entsprechenden Nummer von 1 - 24 fortlaufend durchnummeriert und im OLED angezeigt.
- Die grün-markierten Ports müssen in das #define **PARTLY_USED_PORTS_TAB** eingetragen werden und zwar genau mit den Nummern aus der Abbildung. Daraus berechnet LocoTurn dann die korrekte Lage des Ports.
- Die hellgrauen Ports sind inaktiv und werden später bei Bewegungen übersprungen und können nicht angefahren werden.



Die kleinste sinnvolle Anzahl Ports ist 2: Ein Gleisabgang plus der dazugehörige gegenüberliegende U-Turn-Port für das Wenden der Lok. Damit kann man dann aber wirklich nur Loks wenden 😵‍💫.

Nummerierung der Ports/Gleisabgänge

In der Software-Version V1.0 hängt die Nummerierung der Ports im wesentlichen vom #define **PARTLY_USED_PORTS** ab. Es werden entweder alle Drehscheiben-Raster mit einer Nummer versehen und sind anfahrbar oder nur aktive Ports erhalten eine Nummer und nur diese aktiven Ports sind anfahrbar.

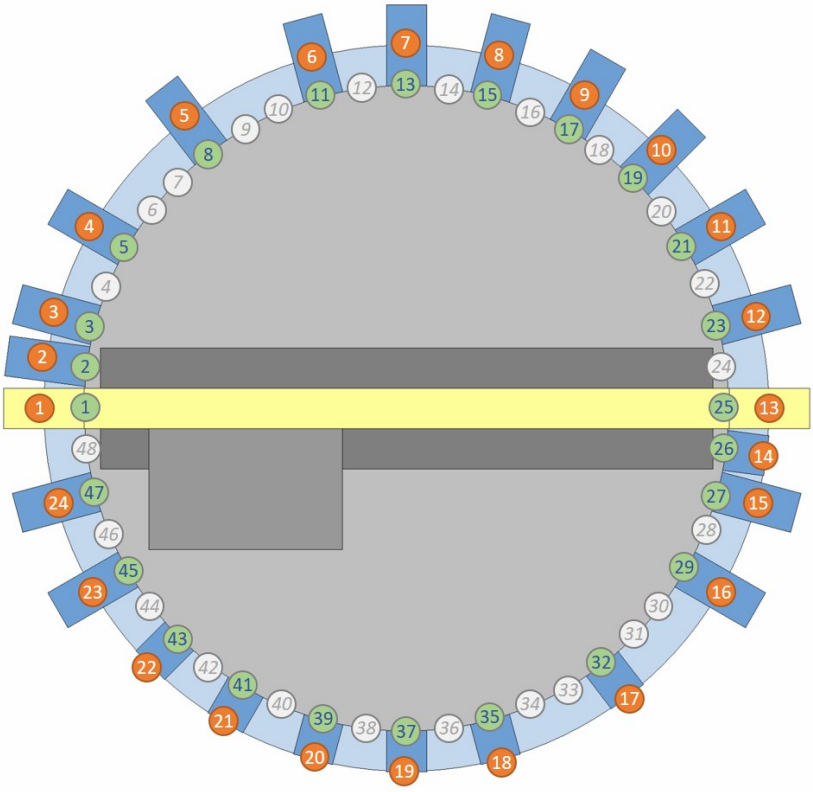
In der Version V1.1 gibt es darüber hinaus folgende Möglichkeiten, die über das neue #define **SHOW_REAL_PORT** eingestellt werden:

1. **Nur aktive Ports werden in die Nummerierung einbezogen**. Die Nummern in unserer Beispieldrehscheibe laufen von 1 - 24, nicht von 1 - 48. Das entspricht der Version 1.0 oben.

Nummerierung der Gleisabgänge/Ports bei:
- #define PARTLY_USED_PORTS_TAB = 1
- #define SHOW_REAL_PORT = 0

Nur aktive Ports sind anfahrbar, in der OLED-Anzeige werden die aktiven Ports durchnummeriert und entsprechend von 1 – 24 dargestellt.
Bei der Anzeige des Ports während einer Bewegung wird die Portnummer (1 - 24) dargestellt, d.h. die orangenen Nummern.

1 Portnummer LocoTurn (OLED-Anzeige)
2 Portnummer für #define PARTLY_USED_PORTS_TAB (Tabelle der Ports, die angefahren werden können)
4 Inaktiver/unbenutzter Port

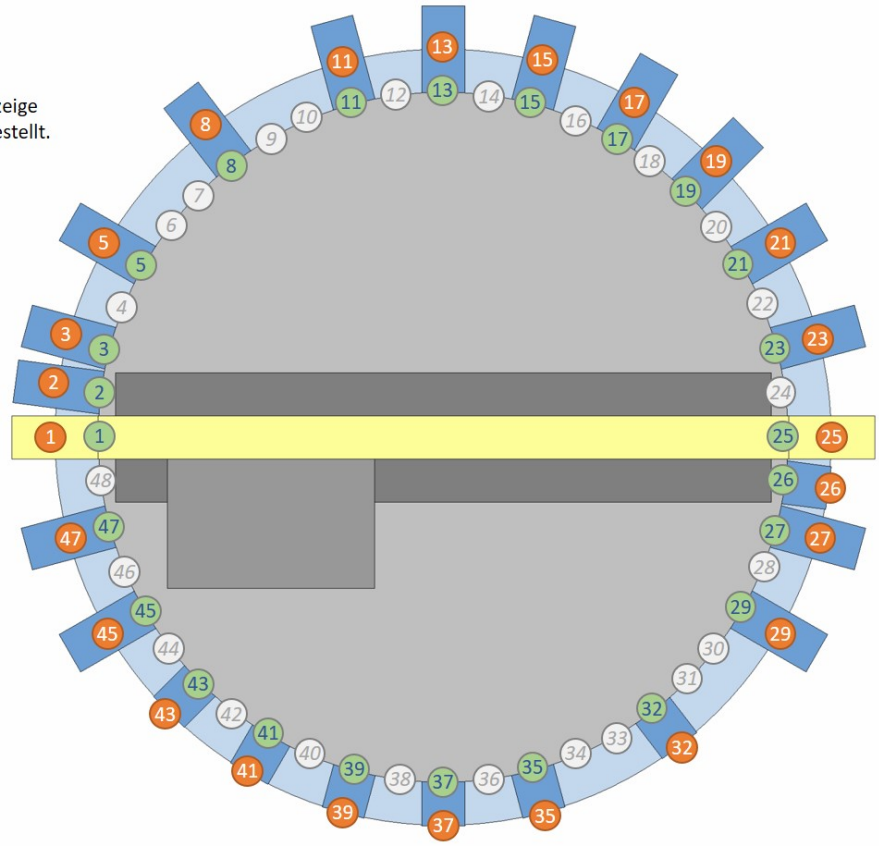


2. **Echte Drehscheiben nummerieren häufig alle Ports durch**, auch inaktive Ports, an denen keine Gleise angeschlossen ist, erhalten eine Nummer. Die Drehscheibe hat demzufolge dann 1 - 48 Ports, von denen nur bestimmte Ports genutzt werden.

Nummerierung der Gleisabgänge/Ports bei:
- #define PARTLY_USED_PORTS_TAB = 1
- #define SHOW_REAL_PORT = 1

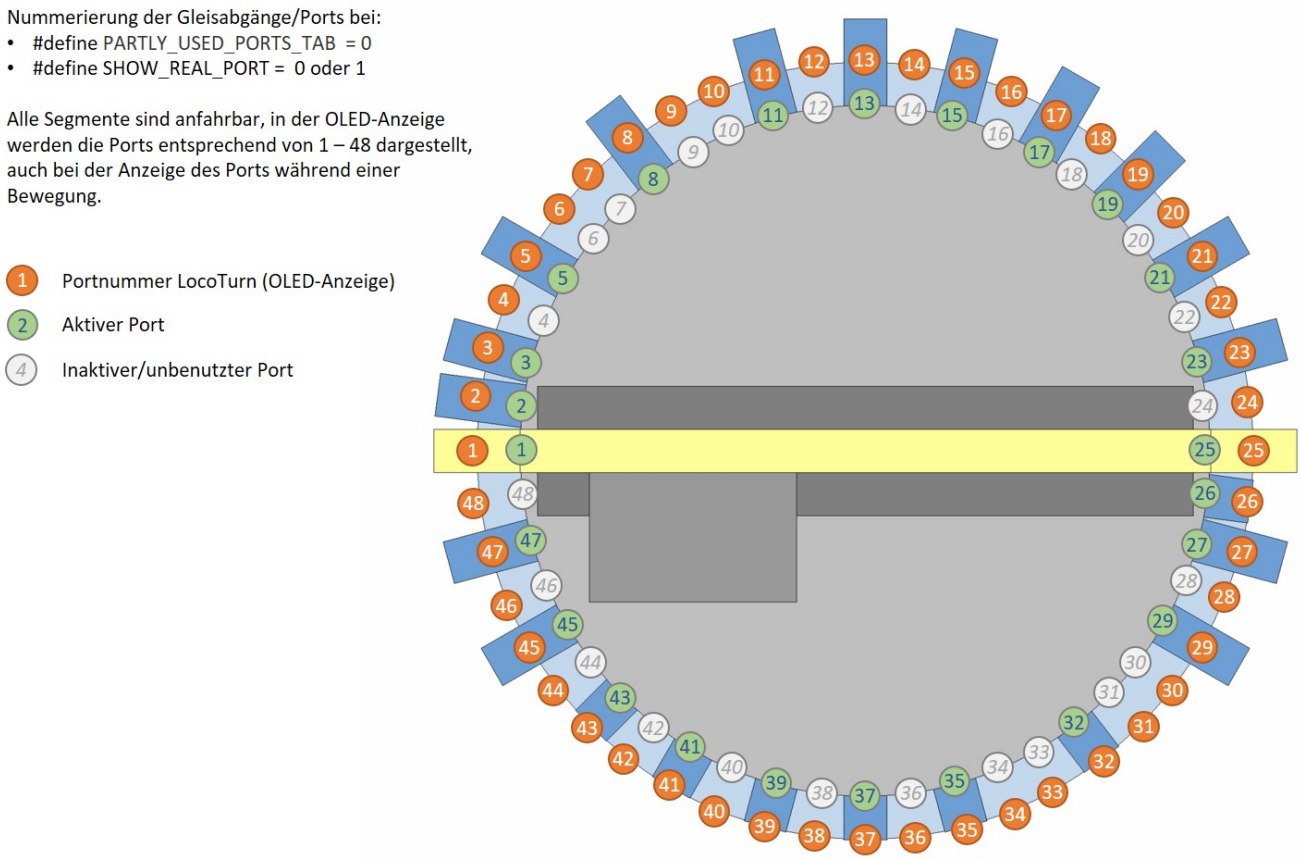
Nur aktive Ports sind anfahrbar, in der OLED-Anzeige werden die Ports entsprechend von 1 – 48 dargestellt.
Inaktive Nummern werden übersprungen.
Bei der Anzeige des Ports während einer Bewegung wird die echte Portnummer (1 - 48) dargestellt, d.h. die grünen/grauen Nummern.

1 Portnummer LocoTurn (OLED-Anzeige)
2 Aktiver Port
4 Inaktiver/unbenutzter Port (können nicht angefahren werden)



3. Wenn **alle potentiellen Drehscheibenabgänge** angefahren werden können, sieht die

Nummerierung folgendermaßen aus:



Beispielkonfigurationen

LocoTurn V1.0

Hier stehen alle wichtigen Parameter für den Betrieb einer Fleischmann-Drehscheibe mit einem Direktantrieb über einen 400 Step Motor. Genutzt werden 26 von 48 Ports. Am einfachsten orientiert man sich an diesem Beispiel und nimmt ggf. Änderungen vor:

```
//================================================================================
================================================================================
============
// Benutzer-spezifische Parameter
//================================================================================
================================================================================
============
/*
    Die Parameter hier dominieren diejenigen aus dem Hauptprogramm
"Turntable"!
    Falls ein bestimmter Parameter hier nicht definiert wurde, wird ein
default-Wert aus dem Hauptsketch verwendet!
```

```
    Hier stehen nur die wichtigsten Parameter, die am häufigsten
benutzerindividuell eingestellt werden müssen.

    Intention: Bei eine neuen Version des Hauptprogrammes bleiben die
Einstellungen aus diesem Reiter erhalten!
*/


//=========================================================================
=========================================================================
============
// Parameterset 1: Wantai-Stepper oder StepperOnline-Stepper ohne Getriebe,
400 Steps pro Umdrehung, 6400 Micro-Steps
// 26 Gleisabgänge (= Ports) domapi-Anlage
//=========================================================================
=========================================================================
============

//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------
// Zentrale #defines zum Ein-/Ausschalten von Funktionen, um evtl. Speicher
für Tests zu sparen!
//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------

#define USE_DCC                                 1           // 1 = use DCC
interface          0 = disable all DCC routines

#define USE_SOUND                               1           // 1 = Sound
features enabled     0 = Sound features disabled

#define USE_SERIAL_INPUT                        0           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
#define USE_VERBOSE                             0           // 1 = ausführliches
Menü         0 = gekürztes Menü / nur wichtigste Befehle

#define USE_OLED                                1           // 1 = use an OLED
Display (could be disabled ("0") to save memory for tests ~6200 Bytes FLASH
!!!). The outputs are sent to RS232 (serial monitor), then.
#define USE_BUTTONS                             1              // Nutzung der 4
Taster auf der roten Panelplatine über analogen Eingang A0; es können auch
zwei Taster gleichzeitig gedrückt sein

#define SHOW_STATUS                             1           // 1 = Status-Screen
im Menü aufrufbar, zeigt einige aktuelle Infos (Position, Ports,
Steps/Umdrehung usw.)     0 = nicht aufrufbar, spart Speicher
```

```
#define USE_POTI                                1           /* 0 = Drehen des
Poti löst keine Drehscheibenbewegung aus       1 = Poti für manuelle
Bewegungen wird benutzt
                                                            Achtung: Falls
kein Poti angeschlossen wird, ist dieser Wert auf 0 zu setzen. Andernfalls
schleicht die Drehscheibe u.U. vor sich hin.*/

//------------------------------------------------------------------------
------------------------------------------------------------------------
---------------------------------------
// *** Debugging***
//------------------------------------------------------------------------
------------------------------------------------------------------------
---------------------------------------
#define ENABLE_DPRINTF                          1           // Debug Ausgaben
ein, zu Beginn evtl. auf 1 lassen, damit man sich die gespeicherten
Positionswerte im seriellen Monitor ansehen kann


// Typische Konfigurationen:
// =========================
// Normalbetrieb auf der Anlage
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//#define USE_DCC                               1           // 1 = use DCC
interface          0 = disable all DCC routines
//#define USE_SOUND                             1           // 1 = Sound
features enabled     0 = Sound features disabled
//#define USE_SERIAL_INPUT                      0           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
//#define USE_VERBOSE                           0           // 1 =
ausführliches Menü         0 = gekürztes Menü / nur wichtigste Befehle
//define SHOW_STATUS                            1
//#define ENABLE_DPRINTF                        0           // Debug Ausgaben
aus

// Test der Steuerung und Einstellung der DS
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//#define USE_DCC                               0           // 1 = use DCC
interface          0 = disable all DCC routines
//#define USE_SOUND                             1           // 1 = Sound
features enabled     0 = Sound features disabled
//#define USE_SERIAL_INPUT                      1           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
//#define USE_VERBOSE                           1           // 1 =
ausführliches Menü         0 = gekürztes Menü / nur wichtigste Befehle
//#define SHOW_STATUS                           0
```

```
//#define ENABLE_DPRINTF                        1           // Debug Ausgaben
ein


//------------------------------------------------------------------------
------------------------------------------------------------------------
------------------
// Steuerung Beleuchtungseffekte
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------------
#define LEDS_ON_BOARD                         1                       //
Beleuchtungssteuerung über die LEDs auf der Platine (ohne WS28xx, ohne MLL)
#define WS281X_BOARD                          2                       // über
eine kleine Bühnenplatine
#define MOBALEDLIB                            3                       // über
die MLL

#define LIGHT_CONTROL                         WS281X_BOARD            //
MOBALEDLIB //WS281X_BOARD  //LEDS_ON_BOARD

#define HOUSE_LIGHT_MODUS                     2                       // nur
für direktes WS281x-Board: 0 = einfaches Ein/Aus  1 = Fade in/out  2 =
Neonflicker

//------------------------------------------------------------------------
------------------------------------------------------------------------
------------
// *** Turntable ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------

/*  - Die Anzahl verwendeter Ports muss bei PORT_CNT, die Portnummern bei
PARTLY_USED_PORTS_TAB eingetragen werden
    - Weiterhin pro Port 0 oder 1 bei der Signalsteuerung PORT_TYPE.
    - Die letzte genutzte DCC-Adresse wird automatisch über eine Formel
anhand der Anzahl Pors berechnet (LAST_USED_DCC_ADDR).
    - Und die Polarisierung muss unten eingestellt werden.
POLARISATION_CHANGE_PORT oder POLARISATION_RELAIS_LIST.
*/

#define PORT_CNT                              48                      //
Number of ports/Gleisabgänge, hier symmetrische Fleischmann-Drehscheibe
#define PARTLY_USED_PORTS                     1                       // 0 =
all ports can be selected, 1 = only valid ports can be used

#if PARTLY_USED_PORTS                                                 /* If
not all ports of the turntable should be selectable the following table
could define the used ports. The used ports are defined as a reference to a
fully defined table*/
```

```
    #define PARTLY_USED_PORTS_REFERENCE     48                      // which
has PARTLY_USED_PORTS_REFERENCE ports (Gesamtanzahl Ports der FLM
Drehscheibe = 48 entspricht 7,5° Raster)
    #undef  PORT_CNT                                                // The
number of real used ports is given in PORT_CNT.
    #define PORT_CNT                        26                      // If
the used ports should be given in degree PARTLY_USED_PORTS_REFERENCE is set
to 360; wir nutzen nur 26 von 48 Gleisabgängen/Ports

    // Gleis 1 = 9:00-Position Einfahrgleis von links (Sensor befindet sich
einen Port rechts davon) passt besser zur Zählung von TC9
    #define PARTLY_USED_PORTS_TAB           { 1, 2, 3, 5, 6, 8, 11, 13, 15,
17, 19, 21, 23, 25, 26, 27, 29, 30, 32, 35, 37, 39, 41, 43, 45, 47 }
// The table must have PORT_CNT entries which are sorted in ascending order

#endif

#define CIRCUMFERENCE                       980.15                  // 312
mm * Pi = circumference of the turntable [mm]; hier eine FLM-Drehscheibe mit
312 mm Bühnen-Länge



#define OFFSET_PORT_POSITION                0                      // Um die
Feinjustierung zu vereinfachen, wird bei Ports mit Defaultwerten die
Position um diese Anzahl Steps entfernt angefahren (CW negativ, CCW
positiv); nur bei unsymmetriwchen DS nutzen!
#define OFFSET_PORT_POSITION_U_TURN         0                      //
parameter used for U-turns



/*
Tipp:
-----

Man kann das Raster, in dem die Ports initial angelegt werden auch kleiner
machen, z.B. 0,1 Grad genau. Hierzu muss man folgendes eintragen:

#define PORT_CNT                            48                      // Der
Wert hier ist eigentlich egal, er wird weiter unten "überschrieben"
#define PARTLY_USED_PORTS                   1                       // 1 =
only valid ports can be used

#if  PARTLY_USED_PORTS
     #define PARTLY_USED_PORTS_REFERENCE    3600                    //
Einteilung der DS in 3600 Ports
     #undef  PORT_CNT
     #define PORT_CNT                       4
     #define PARTLY_USED_PORTS_TAB          {300, 1350, 2100, 3150} // davon
nutzen wir nur 4 Ports an diesen Positionen
#endif
```

```
Im Prinzip teilt das den Vollkreis in 3600 Ports/Gleisabgänge, von denen
aber nur 4 genutzt werden, die quasi an beliebiger Stelle liegen können. Das
Beispiel oben hat 4 anfahrbare Ports bei:

Grad                   Uhrzeit
30°                       1
135°                   "halb" 5
210°                      7
315°                   "halb" 11
*/


//------------------------------------------------------------------------
------------------------------------------------------------------------
---------
// *** Signalsteuerung ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------
#define ADVANCED_SIGNAL_CONTROL             1                       //
Steuerung der Signale in Abhängigkeit, ob die DS bei Stillstand an einem
aktiven/gültigen oder blinden Port steht (= 1). Einfache Signalsteuerung =
0.

// Gleisabgang                               1  2  3  4  5  6  7  8  9 10 11
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
#define PORT_TYPE                            1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0 // Port-Type = 1 if there is
an active / valid railway connected,  0 = blind port (no railway connected).
The table must have PORT_CNT entries. Required for advanced signal control
(signal = red if turntable is no at a valid port)
// als Beispiel hier: bei 26 Ports sind einige aktiv, einige sind blind!


//------------------------------------------------------------------------
------------------------------------------------------------------------
---------
// *** Polarization relays ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------
#define POLARISATION_RELAIS_PIN             A1                          //
Polarisations-Relais for dual rail system (Set to -1 if not used)
#define POLARISATION_RELAIS_INVERS          0                           // 0:
Pin is set to high, if  POLARISATION_RELAIS_LIST[Port] = 1      1: Pin is
set to low, if  0

#define POLARISATION_CHANGE_PORT_START      5
#define POLARISATION_CHANGE_PORT_END        20

/*  Teilt die Drehscheibe für die Polarisierung quasi in 2 Hälften.
```

```
    Ab dem hier definierten START-Port (externe Nummerierung wie OLED-
Anzeige) wird das Polarisationsrelais bis zum i.d.R. gegenüberliegenden END-
Port eingeschaltet.
    Die hier eingegebenen Zahlen müssen zwischen 1 und PORT_CNT liegen!
START muss < END sein!
    Die eine Hälfte der Scheibe ist Polung 1, die andere Polung 2.
    Spart einige Bytes ;-)
    -1 = Polarisation gemäß der unten stehenden Liste
*/

//#define POLARISATION_RELAIS_LIST            0, 1, 0, 1 , 0, 1, 0, 1 ...
// Pro genutztem DS-Port hier ein Eintrag in der Liste mit 0 oder 1


//-------------------------------------------------------------------------
-------------------------------------------------------------------------
---------
// *** Direction and control settings ***
//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------
#define ROTATIONSWITCH_DIRECTION            -1                      /* Set
from 1 to -1 to change the direction of the rotation switch
                                                                       -1 =
Drehen nach rechts höhere Port-Nr.; links = niedrigere Port-Nr.*/

#define USE_ROTARY_ENCODER_MOVEMENT         1                       // 0 =
Drehen des Encoders löst keine Drehscheibenbewegung aus, Menüs fkt. aber
1 = Encoder steuert die DS und Menü fkt.

#define SPEED_POTI_DIRECTION                -1                      // Set
to -1 to change the direction of the speed poti; bei domapi-Platine "-1"

#define ROTATIONSWITCH_MENU_DIR             1                       // Set
from 1 to -1 to change the direction of the rotation switch in the menu:
Rechts-Drehung = Zeile nach unten im Menü
#define TURNTABLE_DIRECTION                 1                       // Set 1
to -1 to change the rotation / port number direction, abhängig von der
Polung der Motorspulen --> einfach ausprobieren!


#define ENCODER_LOGIC                       0                       /*  1 =
Drehencoder drehen während Stillstand ermöglicht Portauswahl
- Bewegung wird erst gestartet, wenn Encoder-Taste gedrückt wird
- während der Bewegung kann die anzufahrende Portnummer nicht mehr geändert
werden
- Taste während Bewegung ruft das Menü auf und stoppt die Bewegung
                                                                       0 =
Encoder startet Bewegung, während Bewegung kann weiter gedreht werden*/
```

```
#define DIRECTION_CHANGE_POSSIBLE          0                       // 1 =
Richtungsänderung erlaubt, wenn anderer Befehl bei bereits gestarteter
Bewegung kommt    0 = keine Richtungsänderung, eine einmal eingeschlagene
Richtung wird beibehalten

#define MOVE_STATUS_INVERS                 1                       /*
Invert the level of the S88_MOVING_PIN:
                                                                      - 1
and turntable moving = Optokoppler sperrt.
                                                                      - 1 +
Stillstand: OK steuert durch und erzeugt eine Rückmeldung.
                                                                      Viele
Steuerungssoftwarepakete erwarten ein Signal, wenn die DS steht (d.h.
angekommen ist)*/

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------
// *** Schrittmotor ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
#define FIXED_STEPS_PER_ROUND              6400                   //
manuelle Vorgabe der Anzahl Steps pro Umdrehung; "0" = automatische
Ermittlung

#define FIXED_STEPS_HAS_CONTACT            0                      //
manuelle Vorgabe des Getriebespiels; "-1": dann wird das Spiel automatisch
ermittelt, Werte >= 0 --> manuelle Vorgabe

#define NOT_ENABLE_PIN                     6                       /* Set
to -1 if the stepper driver has an automatic power mode like the TMC2100
                                                                      The
pin of the module must be left open (std 6)
                                                                      Use
Pin 6 for TMC2208 */
#define ENABLE_ALWAYS_ON                   1                       /*
Vermeidet bei meinem aktuellen Steppermotor ohne Getriebe das Ruckeln in der
Start-/Endposition
Nachteil: Motor wird dauerhaft angesteuert und wird warm!
                                                                      Set
to 1 for a powerful stepper without gearbox because the magnetic field will
change the position when powered of
                                                                      If a
stepper driver like the TMC2100 is used it's better to disable the
NOT_ENABLE_PIN (-1) to activate the automatic power saving mode instead.
                                                                      Bei
Einsatz TMC2208 sollte man den Wert 1 verwenden! */

#define STEPPER_RAMP_LENGTH                150                     /* Steps
to speed up the stepper to prevent loosing steps
```

```
                                                                    Set
to 50 if 1/16 steps are used (MS1 - MS3 connected do +5V)*/

#define MOVE_SPEED1                         600                     //
Default speed and activated when DCC_SET_SPEED1_ADDR is received
#define TURNBACK_SPEED                      600                     // Speed
used for TurnBackAndSetZero

#define USE_TURNBACK                        0                       // 1 =
Turn back & set 0-point will be executed if 0-point is detected in negative
rotation direction    0 = not used

#define MOVE_SPEED2                         400                     // Speed
activated when DCC_SET_SPEED2_ADDR is received
#define MOVE_SPEED3                         200                     // Speed
activated when DCC_SET_SPEED3_ADDR is received
#define MOVE_SPEED4                         2600                    // Speed
activated when DCC_SET_SPEED4_ADDR is received

#define CLEARANCE_TEST_SPEED                1200                     //
Speed used in the clearance test
#define CALIBRATE_SPEED_FAST                1200                     //
Speed used for the zero point and total number of steps detection
#define CALIBRATE_SPEED_SLOW                200                      // Slow
speed used for the advanced zero point detection
#define ZERO_DETECTION_OFFSET               60                       // Used
for moving a little bit away from Hall-Sensor

#define MIN_STEPS_HAS_CONTACT               0                       /*
Typical value = 30.
                                                                      Don't
move fast to the contact point in the "Poti" mode if Steps_Has_Contact
                                                                      is
below this value. Set to 0 to disable the check (Always move fast to the
contact point)
Problem: The Steps_Has_Contact detection may be not accurate because of
sensor errors
                                                                      0 =
kein Anfahrruckler
                                                                      TODO:
evtl. muss die Routine, die das Getriebespiel "überbrückt" direkt in der
Loop aufgerufen werden, wenn die eigentlich Bewegung gestartet wird*/

#define ALWAYS_CALIBRATE_AT_START           0                       // Set
to 1 to calibrate the 0-position (Hallsensor) every time when the program is
started. Helps to compensate little jumps of the turntable when turned on.

#define ADVANCED_REFERENCE_DETECTION        1                       // "1"
fährt den Hallsensor beim 0-Punkt Kalibieren schnell an, dreht ein Stückchen
zurück und fährt ihn nochmals langsam an und speichert die Referenzposition
```

```
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------
// *** OLED ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
#define OLED_TYP                              13                        //
Tested with the following displays 87, 91, 96, 13
#define USE_u8x8                              1                         // 1 bei
96 und 13
#define SHOW_POSITION_MAINSCREEN              1                         //
Positionsanzeige auf dem Hauptscreen

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------
// ** Flashing & light setup ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
#define MOVING_FLASH_INVERS                   0                         //
Normal: 0 = LED connected to GND
#define MOVING_FLASH_MODE                     2                         // 1 =
Blink, 2 double flash
#define HOUSE_BRIGHTNESS                      255                       //
Brightness of house LED, valid values 0 - 255, 0 = off, 255 = very bright,
saves separate resistor ;-)

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
// *** sketch internal parameters ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
#define BITSCHIEBER                           1                          // 1 =
direkte Bitmanipulation    0 = digitalRead/Write, pinMode, FastPin  ToDo:
Prüfen, ob irgendwelche Libs noch digitlaRead etc. nutzen
#define SPEED_POTI_MID_RANGE                  35                         //
Range (+/-) of the speed poti which is considered as 0
#define SPEED_POTI_CENTER                     521                        //
Center position of the speed poti (Normally 512)

#define ANALOG_SPEED_DIVISOR                  40                         //
Divisor used to calculate the analog speed with the poti (std 8, 50)
#define MIN_ANALOG_SPEED                      40                         // 1 =>
10 sec reaction time ;-(   Reaction time = 10 sec / MIN_ANALOG_SPEED

#define ALWAYS_CHECK_STEPS_ONE_TURN           0                          //
Always check the steps for one turn at power on
```

```
#define ASK_TO_UPDATE_ALL                    0                          // = 1:
Ask to update all ports when Port 1 is redefined > 1: Ask to update all
ports for every port

#define ALWAYS_SET_ZERO_IN_POS_DIR           1                          // 1 =
Check and adjust the 0-position always, if the hall sensor has been detected
0 = only if 1 complete turn was moved



//------------------------------------------------------------------------
------------------------------------------------------------------------
-----------
// *** Sound Setup ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------
#define USE_JQ6500_SERIAL                    -1                         //
SMART_JQ6500_SERIAL_PIN --[1K]-- TX; -1 = hardware serial is used (TX-Pin)
#define JQ6500_VOLUME                        25                         //
Range: 0..30 (-1 = Don't change the volume)
#define DELAY_TURN_START_SOUND               7400                       //
Delay before start moving if sound is played, hängt ab vom verwendeten
soundfile; bei mir machen 7s Sinn!
#define SOUND1_FILENR                         1                         //
sound/file number of JQ6500 for turntable start and running (Hupe, Anfahren
und Drehen)
#define SOUND2_FILENR                         3                         //
sound for turntable stop (aktuell nur die Hupe)

//------------------------------------------------------------------------
------------------------------------------------------------------------
---------
// *** Pins ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------

// n/a --> alle aus dem Hauptsketch verwenden!



//------------------------------------------------------------------------
------------------------------------------------------------------------
------
// *** DCC ***
//------------------------------------------------------------------------
------------------------------------------------------------------------
------------

#define DCC_OFFSET                           0          // 0 = DCC addresses
start at #211, using this parameter you can use other DCC ranges in case of
```

```
using several turntables with one central station or you rewuire other
address ranges

#define DCC_WAITING_TIME                    350         // waiting time [ms]
for further DCC-tickets before carrying out the DCC command

#define DCC_MAERKLIN_7687_COMPATIBLE        1           // DCC-Befehle nur
für den 1. Halbkreis der DS verwenden. In Verbindung mit der Drehrichtung
kann trotzdem jeder Port so angefahren werden, dass das DS-Haus richtig
steht

#if DCC_MAERKLIN_7687_COMPATIBLE == 1
    #define LAST_USED_DCC_ADDR              DCC_CHKADDR((235 + DCC_OFFSET),
RED)   // die "235" ist an die tatsächlich verwendete letzte Adresse
anzupassen!
#else
    #define LAST_USED_DCC_ADDR              DCC_CHKADDR((229 + PORT_CNT / 2
- 1 + DCC_OFFSET), GRN)
#endif

// Momentan 26 (von 48) genutzte Ports = { 1, 2, 3, 5, 6, 8, 11, 13, 15, 17,
19, 21, 23, 25, 26, 27, 29, 30, 32, 35, 37, 39, 41, 43, 45, 47 }

#if DCC_MAERKLIN_7687_COMPATIBLE == 1
// Es werden nur für die Ports im ersten Halbkreis DCC-Adressen benötigt
// Bei 26 Ports braucht man hier also 13 DCC-Adressen (= 13 Zeilen)

#define DCC_PORT_ADDR_LIST              DCC_PORT_ADDR(1, (229 + DCC_OFFSET),
RED), \
    DCC_PORT_ADDR(2, (229 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(3, (230 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(4, (230 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(5, (231 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(6, (231 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(7, (232 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(8, (232 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(9, (233 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(10, (233 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(11, (234 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(12, (234 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(13, (235 + DCC_OFFSET), RED)

#else
// DCC-Adressen für den kompletten Vollkreis notwendig!
// Bei 26 Ports braucht man hier auch 26 DCC-Adressen (= 26 Zeilen)

#define DCC_PORT_ADDR_LIST              DCC_PORT_ADDR(1, (229 + DCC_OFFSET),
RED), \
    DCC_PORT_ADDR(2, (229 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(3, (230 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(4, (230 + DCC_OFFSET), GRN), \
```

```
    DCC_PORT_ADDR(5, (231 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(6, (231 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(7, (232 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(8, (232 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(9, (233 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(10, (233 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(11, (234 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(12, (234 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(13, (235 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(14, (235 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(15, (236 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(16, (236 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(17, (237 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(18, (237 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(19, (238 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(20, (238 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(21, (239 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(22, (239 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(23, (240 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(24, (240 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(25, (241 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(26, (241 + DCC_OFFSET), GRN)

// Bei mehr als 41 Einträgen/Zeilen in dem obigen #define spinnt der
Compiler!
// Workaround: dann müssen die zusätzlichen Zeilen in das #define ... LIST_2
aufgenommen werden ;-)

//#define DCC_PORT_ADDR_LIST_2         DCC_PORT_ADDR(27, (242 + DCC_OFFSET),
RED), \
//    DCC_PORT_ADDR(28, (242 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(29, (243 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(30, (243 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(31, (244 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(32, (244 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(33, (245 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(34, (245 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(35, (246 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(36, (246 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(37, (247 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(38, (247 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(39, (248 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(40, (248 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(41, (249 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(42, (249 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(43, (250 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(44, (250 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(45, (251 + DCC_OFFSET), RED), \
//    DCC_PORT_ADDR(46, (251 + DCC_OFFSET), GRN), \
//    DCC_PORT_ADDR(47, (252 + DCC_OFFSET), RED), \
```

```
//    DCC_PORT_ADDR(48, (252 + DCC_OFFSET), GRN)

#endif

// ggf. hier weitere Zeilen ergänzen, wenn zusätzliche DCC-Befehle für
weitere Ports benötigt werden
// die letzte DCC-Adresse (hier 241) muss sich oben aus der #define-Formel
bei LAST_USED_DCC_ADR ergeben !!!


//--------------------------------------------------------------------------
---------------------------------------------------------------------------
---------
// *** Tastenauswertung ***
//--------------------------------------------------------------------------
---------------------------------------------------------------------------
------------

// Das #define definiert die 4 Schwellwerte für die 4 Taster auf der
Panelplatine hier in der Config-Datei
// Bei Änderungen  in der ino-Datei bleiben die individuellen Werte unten
erhalten!

#define BUTTON_THRESHOLDS    {970, 670, 831, 319}, \
    {327, 284, 310, 319}, \
    {696, 670, 284, 622}, \
    {872, 831, 310, 622}

#define HOME_RUN_PORT 8         // wird bei der Tastersteuerung verwendet,
bei einem entsprechenden Tastendruck fährt die DS zum angegebenen Port

/*  folgende Aktionen stehen für die Taster zur Verfügung:
 *
    B_Toggle_House()              Hausbeleuchtung ein/aus
    B_Toggle_Sound()              Sound ein/aus
    B_Toggle_Signal_House()       Signal Hausseite rot/weiß
    B_Toggle_Signal_Opposite()    Signal Gegenüber rot/weiß
    B_Signal_House_red()          Signal Haussseite rot
    B_Signal_House_white()        dito. weiß
    B_Signal_Opp_red()            Gegenüber rot
    B_Signal_Opp_white()          dito. weiß
    B_U_Turn_CW()                 180°-Drehung CW
    B_Home_Run()                  Anfahren der Home-Position
    B_Toggle_Flash()              Warnleuchte ein/aus
    Play_Sound(n)                 Sound n auf dem JQ6500 abspielen
*/

// Hier den Tastern die Aktion zuordnen (kann man auch mehrfach machen, dann
";" dazwischen!):

#define BUTTON_1    B_Home_Run()
```

```
#define BUTTON_2    B_U_Turn_CW()
#define BUTTON_3    B_Toggle_House()
#define BUTTON_4    B_Toggle_Flash()
```

LocoTurn V1.1 / V1.2

In der Version V1.1 sind einige #defines dazugekommen oder wurden umbenannt. Die folgende Turntable_config.h ist nur mit der entsprechenden ino-Version (1.1 oder 1.2) lauffähig.

```
//================================================================================
================================================================================
============
// Benutzer-spezifische Parameter
//================================================================================
================================================================================
============
/*
    Die Parameter hier dominieren diejenigen aus dem Hauptprogramm
"Turntable"!
    Falls ein bestimmter Parameter hier nicht definiert wurde, wird ein
default-Wert aus dem Hauptsketch verwendet!
    Hier stehen nur die wichtigsten Parameter, die am häufigsten
benutzerindividuell eingestellt werden müssen.

    Intention: Bei eine neuen Version des Hauptprogrammes bleiben die
Einstellungen aus diesem Reiter erhalten!
*/



//================================================================================
================================================================================
============
// Parameterset 1: Wantai-Stepper oder StepperOnline-Stepper ohne Getriebe,
400 Steps pro Umdrehung, 6400 Micro-Steps
// 26 Gleisabgänge (= Ports) domapi-Anlage
//================================================================================
================================================================================
============

//--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------
// Zentrale #defines zum Ein-/Ausschalten von Funktionen, um evtl. Speicher
für Tests zu sparen!
//--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------
```

```
#define USE_DCC                                1           // 1 = use DCC
interface          0 = disable all DCC routines

#define USE_SOUND                              1           // 1 = Sound
features enabled     0 = Sound features disabled

#define USE_SERIAL_INPUT                       0           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
#define USE_VERBOSE                            0           // 1 = ausführliches
Menü         0 = gekürztes Menü / nur wichtigste Befehle

#define USE_OLED                               1           // 1 = use an OLED
Display (could be disabled ("0") to save memory for tests ~6200 Bytes FLASH
!!!). The outputs are sent to RS232 (serial monitor), then.
#define USE_BUTTONS                            1              // Nutzung der 4
Taster auf der roten Panelplatine über analogen Eingang A0; es können auch
zwei Taster gleichzeitig gedrückt sein

#define SHOW_STATUS                            1           // 1 = Status-Screen
im Menü aufrufbar, zeigt einige aktuelle Infos (Position, Ports,
Steps/Umdrehung usw.)     0 = nicht aufrufbar, spart Speicher

#define USE_POTI                               1           /* 0 = Drehen des
Poti löst keine Drehscheibenbewegung aus       1 = Poti für manuelle
Bewegungen wird benutzt
                                                              Achtung: Falls
kein Poti angeschlossen wird, ist dieser Wert auf 0 zu setzen. Andernfalls
schleicht die Drehscheibe u.U. vor sich hin.*/

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------
// *** Debugging***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------
#define ENABLE_DPRINTF                         1           // Debug Ausgaben
ein, zu Beginn evtl. auf 1 lassen, damit man sich die gespeicherten
Positionswerte im seriellen Monitor ansehen kann


// Typische Konfigurationen:
// =========================
// Normalbetrieb auf der Anlage
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//#define USE_DCC                              1           // 1 = use DCC
interface          0 = disable all DCC routines
//#define USE_SOUND                            1           // 1 = Sound
features enabled     0 = Sound features disabled
```

```
//#define USE_SERIAL_INPUT                  0           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
//#define USE_VERBOSE                       0           // 1 = ausführliches
Menü         0 = gekürztes Menü / nur wichtigste Befehle
//define SHOW_STATUS                        1
//#define ENABLE_DPRINTF                    0           // Debug Ausgaben
aus

// Test der Steuerung und Einstellung der DS
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//#define USE_DCC                           0           // 1 = use DCC
interface          0 = disable all DCC routines
//#define USE_SOUND                         1           // 1 = Sound
features enabled     0 = Sound features disabled
//#define USE_SERIAL_INPUT                  1           // 1 = Serial
monitor input activated   0 = no processing of serial commands  --> sollte
in finaler Version ausgeschaltet sein, da man auf der Anlage später nicht
über den PC steuert
//#define USE_VERBOSE                       1           // 1 = ausführliches
Menü         0 = gekürztes Menü / nur wichtigste Befehle
//#define SHOW_STATUS                       0
//#define ENABLE_DPRINTF                    1           // Debug Ausgaben
ein


//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------------
// Steuerung Beleuchtungseffekte
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------------
#define LEDS_ON_BOARD                       1                       //
Beleuchtungssteuerung über die LEDs auf der Platine (ohne WS28xx, ohne MLL)
#define WS281X_BOARD                        2                       // über
eine kleine Bühnenplatine
#define MOBALEDLIB                          3                       // über
die MLL

#define LIGHT_CONTROL                       WS281X_BOARD            //
MOBALEDLIB //WS281X_BOARD  //LEDS_ON_BOARD

#define HOUSE_LIGHT_MODUS                   2                       // nur
für direktes WS281x-Board: 0 = einfaches Ein/Aus  1 = Fade in/out  2 =
Neonflicker

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
```

```
------------
// *** Turntable ***
//---------------------------------------------------------------------------
---------------------------------------------------------------------------
------------

/*  - Die Anzahl verwendeter Ports muss bei PORT_CNT, die Portnummern bei
PARTLY_USED_PORTS_TAB eingetragen werden
    - Weiterhin pro Port 0 oder 1 bei der Signalsteuerung PORT_TYPE.
    - Die letzte genutzte DCC-Adresse wird automatisch über eine Formel
anhand der Anzahl Pors berechnet (LAST_USED_DCC_ADDR).
    - Und die Polarisierung muss unten eingestellt werden.
POLARISATION_CHANGE_PORT oder POLARISATION_RELAIS_LIST.
*/

#define PORT_CNT                            48                      //
Number of ports/Gleisabgänge, hier symmetrische Fleischmann-Drehscheibe
#define PARTLY_USED_PORTS                   1                       // 0 =
all ports can be selected, 1 = only valid ports can be used

#if PARTLY_USED_PORTS                                               /* If
not all ports of the turntable should be selectable the following table
could define the used ports. The used ports are defined as a reference to a
fully defined table*/
#define PARTLY_USED_PORTS_REFERENCE         48                      // which
has PARTLY_USED_PORTS_REFERENCE ports (Gesamtanzahl Ports der FLM
Drehscheibe = 48 entspricht 7,5° Raster)
#undef  PORT_CNT                                                    // The
number of real used ports is given in PORT_CNT.
#define PORT_CNT                            26                      // If
the used ports should be given in degree PARTLY_USED_PORTS_REFERENCE is set
to 360; wir nutzen nur 26 von 48 Gleisabgängen/Ports

// Gleis 1 = 9:00-Position Einfahrgleis von links (Sensor befindet sich
einen Port rechts davon) passt besser zur Zählung von TC9
#define PARTLY_USED_PORTS_TAB           { 1, 2, 3, 5, 6, 8, 11, 13, 15, 17,
19, 21, 23, 25, 26, 27, 29, 30, 32, 35, 37, 39, 41, 43, 45, 47 }        //
The table must have PORT_CNT entries which are sorted in ascending order

#define SHOW_REAL_PORT                      0                       // 0 =
im OLED wird bei PARTLY_USED_PORTS = 1 die fortlaufende Portnummer der
genutzten Ports angezeigt   1 = Anzeige des echten Ports (bei FLM-DS 1 -
48), Wert aus dem obigen Array

#endif

#define CIRCUMFERENCE                       980.15                  // 312
mm * Pi = circumference of the turntable [mm]; hier eine FLM-Drehscheibe mit
312 mm Bühnen-Länge
```

```
#define OFFSET_PORT_POSITION                0                       // Um die
Feinjustierung zu vereinfachen, wird bei Ports mit Defaultwerten die
Position um diese Anzahl Steps entfernt angefahren (CW negativ, CCW
positiv); nur bei unsymmetriwchen DS nutzen!
#define OFFSET_PORT_POSITION_U_TURN         0                       //
parameter used for U-turns



/*
    Tipp:
    -----

    Man kann das Raster, in dem die Ports initial angelegt werden auch
kleiner machen, z.B. 0,1 Grad genau. Hierzu muss man folgendes eintragen:

    #define PORT_CNT                            48                      //
Der Wert hier ist eigentlich egal, er wird weiter unten "überschrieben"
    #define PARTLY_USED_PORTS                   1                       // 1
= only valid ports can be used

    #if  PARTLY_USED_PORTS
     #define PARTLY_USED_PORTS_REFERENCE    3600                        //
Einteilung der DS in 3600 Ports
     #undef  PORT_CNT
     #define PORT_CNT                           4
     #define PARTLY_USED_PORTS_TAB          {300, 1350, 2100, 3150}     //
davon nutzen wir nur 4 Ports an diesen Positionen
    #endif

    Im Prinzip teilt das den Vollkreis in 3600 Ports/Gleisabgänge, von denen
aber nur 4 genutzt werden, die quasi an beliebiger Stelle liegen können. Das
Beispiel oben hat 4 anfahrbare Ports bei:

    Grad                  Uhrzeit
    30°                      1
    135°                  "halb" 5
    210°                     7
    315°                  "halb" 11
*/



//-------------------------------------------------------------------------
-------------------------------------------------------------------------
---------
// *** Signalsteuerung ***
//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------
#define ADVANCED_SIGNAL_CONTROL             1                       //
Steuerung der Signale in Abhängigkeit, ob die DS bei Stillstand an einem
```

```
aktiven/gültigen oder blinden Port steht (= 1). Einfache Signalsteuerung =
0.

// Option 1: wenn PARTLY_USE_PORTS = 0
// Gleisabgang                              1  2  3  4  5  6  7  8  9  10 11
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45 46 47 48
// #define PORT_TYPE                        1, 1, 1, 0, 0, 1, 0, 1, 0,  0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0     // Port-Type = 1 if there is an
active / valid railway connected,  0 = blind port (no railway connected).
The table must have PORT_CNT entries. Required for advanced signal control
(signal = red if turntable is no at a valid port)

// Option 2: wenn PARTLY_USE_PORTS = 1
#define PORT_TYPE                            1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0 // Port-Type = 1 if there is
an active / valid railway connected,  0 = blind port (no railway connected).
The table must have PORT_CNT entries. Required for advanced signal control
(signal = red if turntable is no at a valid port)
// als Beispiel hier: bei 26 Ports sind einige aktiv, einige sind blind!


//-------------------------------------------------------------------------
-------------------------------------------------------------------------
---------
// *** Polarization relays ***
//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------
#define POLARISATION_RELAIS_PIN             A1                          //
Polarisations-Relais for dual rail system (Set to -1 if not used)
#define POLARISATION_RELAIS_INVERS          0                           // 0:
Pin is set to high, if  POLARISATION_RELAIS_LIST[Port] = 1      1: Pin is
set to low, if  0

#define POLARISATION_CHANGE_PORT_START      5
#define POLARISATION_CHANGE_PORT_END        20

/*  Teilt die Drehscheibe für die Polarisierung quasi in 2 Hälften.
    Ab dem hier definierten START-Port (externe Nummerierung) wird das
Polarisationsrelais bis zum i.d.R. gegenüberliegenden END-Port
eingeschaltet.
    Die hier eingegebenen Zahlen müssen zwischen 1 und PORT_CNT liegen!
START muss < END sein!
    Die eine Hälfte der Scheibe ist Polung 1, die andere Polung 2.
    Spart einige Bytes ;-)
    -1 = Polarisation gemäß der unten stehenden Liste
*/

//#define POLARISATION_RELAIS_LIST           0, 1, 0, 1 , 0, 1, 0, 1 ...
```

```
// Pro genutztem DS-Port hier ein Eintrag in der Liste mit 0 oder 1



//--------------------------------------------------------------------------
-------------------------------------------------------------------------
---------
// *** Direction and control settings ***
//--------------------------------------------------------------------------
-------------------------------------------------------------------------
------------
#define ROTATIONSWITCH_DIRECTION            -1                      /* Set
from 1 to -1 to change the direction of the rotation switch
                                                                       -1 =
Drehen nach rechts höhere Port-Nr.; links = niedrigere Port-Nr.*/

#define USE_ROTARY_ENCODER_MOVEMENT         1                       // 0 =
Drehen des Encoders löst keine Drehscheibenbewegung aus, Menüs fkt. aber
1 = Encoder steuert die DS und Menü fkt.

#define SPEED_POTI_DIRECTION                -1                      // Set
to -1 to change the direction of the speed poti; bei domapi-Platine "-1"

#define ROTATIONSWITCH_MENU_DIR             1                       // Set
from 1 to -1 to change the direction of the rotation switch in the menu:
Rechts-Drehung = Zeile nach unten im Menü
#define TURNTABLE_DIRECTION                 1                       // Set 1
to -1 to change the rotation / port number direction, abhängig von der
Polung der Motorspulen --> einfach ausprobieren!



// Wenn fahrtdauerabhängige Sounds verwendet werden, dann muss ENCODER_LOGIC
= 1 sein, sonst passen die Sounds nicht zur Bewegung ;-)
#define ENCODER_LOGIC                       1                       /*  1 =
Drehencoder drehen während Stillstand ermöglicht Portauswahl
- Bewegung wird erst gestartet, wenn Encoder-Taste gedrückt wird
- während der Bewegung kann die anzufahrende Portnummer nicht mehr geändert
werden
- Taste während Bewegung ruft das Menü auf und stoppt die Bewegung
                                                                         0 =
Encoder startet Bewegung, während Bewegung kann weiter gedreht werden*/

#define DIRECTION_CHANGE_POSSIBLE           0                       // 1 =
Richtungsänderung erlaubt, wenn anderer Befehl bei bereits gestarteter
Bewegung kommt    0 = keine Richtungsänderung, eine einmal eingeschlagene
Richtung wird beibehalten

#define MOVE_STATUS_INVERS                  1                       /*
Invert the level of the S88_MOVING_PIN:
                                                                       - 1
and turntable moving = Optokoppler sperrt.
```

```
                                                                    - 1 +
Stillstand: OK steuert durch und erzeugt eine Rückmeldung.
                                                                    Viele
Steuerungssoftwarepakete erwarten ein Signal, wenn die DS steht (d.h.
angekommen ist)*/

//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------
// *** Schrittmotor ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
#define FIXED_STEPS_PER_ROUND               6400                    //
manuelle Vorgabe der Anzahl Steps pro Umdrehung; "0" = automatische
Ermittlung

#define FIXED_STEPS_HAS_CONTACT             0                       //
manuelle Vorgabe des Getriebespiels; "-1": dann wird das Spiel automatisch
ermittelt, Werte >= 0 --> manuelle Vorgabe

#define NOT_ENABLE_PIN                      6                       /* Set
to -1 if the stepper driver has an automatic power mode like the TMC2100
                                                                    The
pin of the module must be left open (std 6)
                                                                    Use
Pin 6 for TMC2208 */
#define ENABLE_ALWAYS_ON                    1                       /*
Vermeidet bei meinem aktuellen Steppermotor ohne Getriebe das Ruckeln in der
Start-/Endposition
Nachteil: Motor wird dauerhaft angesteuert und wird warm!
                                                                    Set
to 1 for a powerful stepper without gearbox because the magnetic field will
change the position when powered of
                                                                    If a
stepper driver like the TMC2100 is used it's better to disable the
NOT_ENABLE_PIN (-1) to activate the automatic power saving mode instead.
                                                                    Bei
Einsatz TMC2208 sollte man den Wert 1 verwenden! */

#define STEPPER_RAMP_LENGTH                 180                     /* Steps
to speed up the stepper to prevent loosing steps
                                                                    Set
to 50 if 1/16 steps are used (MS1 - MS3 connected do +5V)*/

#define MOVE_SPEED1                         600                     //
Default speed and activated when DCC_SET_SPEED1_ADDR is received
#define TURNBACK_SPEED                      600                     // Speed
used for TurnBackAndSetZero

#define USE_TURNBACK                        0                       // 1 =
```

```
Turn back & set 0-point will be executed if 0-point is detected in negative
rotation direction    0 = not used

#define MOVE_SPEED2                          400                     // Speed
activated when DCC_SET_SPEED2_ADDR is received
#define MOVE_SPEED3                          200                     // Speed
activated when DCC_SET_SPEED3_ADDR is received
#define MOVE_SPEED4                          2600                    // Speed
activated when DCC_SET_SPEED4_ADDR is received

#define CLEARANCE_TEST_SPEED                 1200                     //
Speed used in the clearance test
#define CALIBRATE_SPEED_FAST                 1200                     //
Speed used for the zero point and total number of steps detection
#define CALIBRATE_SPEED_SLOW                 200                     // Slow
speed used for the advanced zero point detection
#define ZERO_DETECTION_OFFSET                60                      // Used
for moving a little bit away from Hall-Sensor

#define MIN_STEPS_HAS_CONTACT                0                       /*
Typical value = 30.
                                                                       Don't
move fast to the contact point in the "Poti" mode if Steps_Has_Contact
                                                                       is
below this value. Set to 0 to disable the check (Always move fast to the
contact point)
Problem: The Steps_Has_Contact detection may be not accurate because of
sensor errors
                                                                       0 =
kein Anfahrruckler
                                                                       TODO:
evtl. muss die Routine, die das Getriebespiel "überbrückt" direkt in der
Loop aufgerufen werden, wenn die eigentlich Bewegung gestartet wird*/

#define ALWAYS_CALIBRATE_AT_START            0                       // Set
to 1 to calibrate the 0-position (Hallsensor) every time when the program is
started. Helps to compensate little jumps of the turntable when turned on.

#define ADVANCED_REFERENCE_DETECTION         1                       // "1"
fährt den Hallsensor beim 0-Punkt Kalibieren schnell an, dreht ein Stückchen
zurück und fährt ihn nochmals langsam an und speichert die Referenzposition



//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------
// *** OLED ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------
```

```
#define OLED_TYP                            13                          //
Tested with the following displays 87, 91, 96, 13
#define USE_u8x8                            1                           // 1 bei
96 und 13
#define SHOW_POSITION_MAINSCREEN            1                           //
Positionsanzeige auf dem Hauptscreen

//--------------------------------------------------------------------------
------------------------------------------------------------------------------
---------
// ** Flashing & light setup ***
//--------------------------------------------------------------------------
------------------------------------------------------------------------------
------------
#define MOVING_FLASH_INVERS                 0                           //
Normal: 0 = LED connected to GND
#define MOVING_FLASH_MODE                   2                           // 1 =
Blink, 2 double flash
#define HOUSE_BRIGHTNESS                    255                         //
Brightness of house LED, valid values 0 - 255, 0 = off, 255 = very bright,
saves separate resistor ;-)

//--------------------------------------------------------------------------
------------------------------------------------------------------------------
------------
// *** sketch internal parameters ***
//--------------------------------------------------------------------------
------------------------------------------------------------------------------
------------
#define BITSCHIEBER                         1                            // 1 =
direkte Bitmanipulation    0 = digitalRead/Write, pinMode, FastPin  ToDo:
Prüfen, ob irgendwelche Libs noch digitlaRead etc. nutzen
#define SPEED_POTI_MID_RANGE                35                           //
Range (+/-) of the speed poti which is considered as 0
#define SPEED_POTI_CENTER                   521                          //
Center position of the speed poti (Normally 512)

#define ANALOG_SPEED_DIVISOR                40                           //
Divisor used to calculate the analog speed with the poti (std 8, 50)
#define MIN_ANALOG_SPEED                    40                           // 1 =>
10 sec reaction time ;-(   Reaction time = 10 sec / MIN_ANALOG_SPEED

#define ALWAYS_CHECK_STEPS_ONE_TURN         0                            //
Always check the steps for one turn at power on

#define ASK_TO_UPDATE_ALL                   0                            // = 1:
Ask to update all ports when Port 1 is redefined > 1: Ask to update all
ports for every port

#define ALWAYS_SET_ZERO_IN_POS_DIR          1                            // 1 =
Check and adjust the 0-position always, if the hall sensor has been detected
```

```
0 = only if 1 complete turn was moved



//------------------------------------------------------------------------------
------------------------------------------------------------------------------
-----------
// *** Sound Setup ***
//------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------
#define DFPLAYER                            1                       // mit
DS-Kartenslot
#define JQ6500                              2                       // 2MB
Speicher onboard
#define USE_SOUNDMODULE                     DFPLAYER   //JQ6500     //
Auswahl des verwendeten Soundmoduls

#define USE_SOUND_SERIAL                    -1                      //
SMART_JQ6500_SERIAL_PIN --[1K]-- TX; -1 = hardware serial is used (TX-Pin)
for JQ6500 interface; DFPlayer uses softwareserialTX on Pin D12

#define SOUNDMODULE_VOLUME                  10                      //
Range: 0..30 (-1 = Don't change the volume)
#define DELAY_TURN_START_SOUND              7700                    /*
Delay before start moving if sound is played, hängt ab vom verwendeten
soundfile; bei mir machen 7,7s Sinn!
                                                                       Bei
Verwendung eines DFPlayers ca. 300 ms dazu addieren, da dieses Soundmodul
verzögert startet über softwareserial*/
#define SOUND1_FILENR                       1                       //
sound-file number of JQ6500 / DFPLayer for turntable start and running
(Hupe, Anfahren und Drehen)
#define SOUND2_FILENR                       3                       //
sound for turntable stop (aktuell nur die Hupe)
#define SOUND_CONTINUOUS_MOVE               1                       // 0 =
kein Abspielen von sounds bei DS-Bewegungen mit dem Poti        1 = Sound
wird automatisch abgespielt

#define FAHRTDAUER_MESSEN                   0
#define ADVANCED_SOUND                      1                       /* 0 =
es werden zwei Sounds abgespielt, einer beim Starten der Bewegung (gesampelt
aus Hupen, Anfahren und Fahren), einer beim Stoppen (Hupen); Bremssound ist
nicht verfügbar

                                                                       1 =
die Fahrtdauer wird berechnet und ein entsprechender langer Sound gestartet,
hierfür brauchen wir mehrere Files auf dem Soundmodul.
Jedes file entält Hupen, Anfahren, Fahren, Bremsen bis zum Stopp und
abschließendes Hupen. Der variable Anteil ist das Fahren, das muss solange
hintereinander kopiert werden, dass es für die entsprechende Fahrtstrecke
```

```
reicht
                                                                    Ein
separater Sound beim Stoppen (Hupen) wird nur bei manuellem Stopp
abgespielt, beim auto. Fahren ist der sound bereits im Fahrgeräusch
enthalten
ENCODER_LOGIC dann auf 1 setzen (siehe oben)*/
#define DCC_SOUNDFILE_1                     1                     //
File-Nr. auf dem Soundmodul, das abgespielt wird mit dem entsprechenden DCC-
Befehl
#define DCC_SOUNDFILE_2                     2                     //
J6500; Dateien stehen im Rootverzeichnis, Reihenfolge geht nach Reihenfolge
des Kopierens auf das Modul
#define DCC_SOUNDFILE_3                     3                     //
DFPlayer: Dateien müssen im \mp3-Folder stehen, Nomenklatur 0001_beliebiger
Text (4-stellige Nummer, führende Nullen + sprechender Text)
#define DCC_SOUNDFILE_4                     28
#define DCC_SOUNDFILE_5                     29
#define DCC_SOUNDFILE_6                     30
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------
// *** Pins ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------

// n/a --> alle aus dem Hauptsketch verwenden!


//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------
// *** DCC ***
//--------------------------------------------------------------------------
--------------------------------------------------------------------------
------------

#define DCC_OFFSET                          0          // 0 = DCC addresses
start at #211, using this parameter you can use other DCC ranges in case of
using several turntables with one central station or you rewuire other
address ranges

#define DCC_WAITING_TIME                    350        // waiting time [ms]
for further DCC-tickets before carrying out the DCC command

#define DCC_MAERKLIN_7687_COMPATIBLE        1          // DCC-Befehle nur
für den 1. Halbkreis der DS verwenden. In Verbindung mit der Drehrichtung
kann trotzdem jeder Port so angefahren werden, dass das DS-Haus richtig
steht

#if DCC_MAERKLIN_7687_COMPATIBLE == 1
```

```
#define LAST_USED_DCC_ADDR              DCC_CHKADDR((235 + DCC_OFFSET), RED)
// die "235" ist an die tatsächlich verwendete letzte Adresse anzupassen!
#else
#define LAST_USED_DCC_ADDR              DCC_CHKADDR((229 + PORT_CNT / 2 - 1
+ DCC_OFFSET), GRN)
#endif

// Momentan 26 (von 48) genutzte Ports = { 1, 2, 3, 5, 6, 8, 11, 13, 15, 17,
19, 21, 23, 25, 26, 27, 29, 30, 32, 35, 37, 39, 41, 43, 45, 47 }

#if DCC_MAERKLIN_7687_COMPATIBLE == 1
// Es werden nur für die Ports im ersten Halbkreis DCC-Adressen benötigt
// Bei 26 Ports braucht man hier also 13 DCC-Adressen (= 13 Zeilen)

#define DCC_PORT_ADDR_LIST              DCC_PORT_ADDR(1, (229 + DCC_OFFSET),
RED), \
    DCC_PORT_ADDR(2, (229 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(3, (230 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(4, (230 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(5, (231 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(6, (231 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(7, (232 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(8, (232 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(9, (233 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(10, (233 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(11, (234 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(12, (234 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(13, (235 + DCC_OFFSET), RED)

#else
// DCC-Adressen für den kompletten Vollkreis notwendig!
// Bei 26 Ports braucht man hier auch 26 DCC-Adressen (= 26 Zeilen)

#define DCC_PORT_ADDR_LIST              DCC_PORT_ADDR(1, (229 + DCC_OFFSET),
RED), \
    DCC_PORT_ADDR(2, (229 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(3, (230 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(4, (230 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(5, (231 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(6, (231 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(7, (232 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(8, (232 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(9, (233 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(10, (233 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(11, (234 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(12, (234 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(13, (235 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(14, (235 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(15, (236 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(16, (236 + DCC_OFFSET), GRN), \
```

```
    DCC_PORT_ADDR(17, (237 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(18, (237 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(19, (238 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(20, (238 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(21, (239 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(22, (239 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(23, (240 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(24, (240 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(25, (241 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(26, (241 + DCC_OFFSET), GRN)

// Bei mehr als 41 Einträgen/Zeilen in dem obigen #define spinnt der
Compiler!
// Workaround: dann müssen die zusätzlichen Zeilen in das #define ... LIST_2
aufgenommen werden ;-)

/*  #define DCC_PORT_ADDR_LIST_2         DCC_PORT_ADDR(27, (242 +
DCC_OFFSET), RED), \
    DCC_PORT_ADDR(28, (242 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(29, (243 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(30, (243 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(31, (244 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(32, (244 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(33, (245 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(34, (245 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(35, (246 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(36, (246 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(37, (247 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(38, (247 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(39, (248 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(40, (248 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(41, (249 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(42, (249 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(43, (250 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(44, (250 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(45, (251 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(46, (251 + DCC_OFFSET), GRN), \
    DCC_PORT_ADDR(47, (252 + DCC_OFFSET), RED), \
    DCC_PORT_ADDR(48, (252 + DCC_OFFSET), GRN)
*/

#endif

// ggf. hier weitere Zeilen ergänzen, wenn zusätzliche DCC-Befehle für
weitere Ports benötigt werden
// die letzte DCC-Adresse (hier 241) muss sich oben aus der #define-Formel
bei LAST_USED_DCC_ADR ergeben !!!


//--------------------------------------------------------------------------
--------------------------------------------------------------------------
```

```
---------
// *** Tastenauswertung ***
//-------------------------------------------------------------------------
-------------------------------------------------------------------------
------------

// Das #define definiert die 4 Schwellwerte für die 4 Taster auf der
Panelplatine hier in der Config-Datei
// Bei Änderungen  in der ino-Datei bleiben die individuellen Werte unten
erhalten!

#define BUTTON_THRESHOLDS    970, 327, 696, 872

#define HOME_RUN_PORT 26         // wird bei der Tastersteuerung verwendet,
bei einem entsprechenden Tastendruck fährt die DS zum angegebenen Port

/*  folgende Aktionen stehen für die Taster zur Verfügung:

    B_Toggle_House()              Hausbeleuchtung ein/aus
    B_Toggle_Sound()              Sound ein/aus
    B_Toggle_Signal_House()       Signal Hausseite rot/weiß
    B_Toggle_Signal_Opposite()    Signal Gegenüber rot/weiß
    B_Signal_House_red()          Signal Haussseite rot
    B_Signal_House_white()        dito. weiß
    B_Signal_Opp_red()            Gegenüber rot
    B_Signal_Opp_white()          dito. weiß
    B_U_Turn_CW()                 180°-Drehung CW
    B_U_Turn_CCW()                180°-Drehung CCW
    B_Home_Run()                  Anfahren der Home-Position
    B_Toggle_Flash()              Warnleuchte ein/aus
    Play_Sound(n)                 Sound n auf dem JQ6500 abspielen
    Auto_Calibrate(1)             führt die Kalibirierung des Nullpunktes
durch
*/

// Hier den Tastern die Aktion zuordnen (kann man auch mehrfach machen, dann
";" dazwischen!):

// kurzer Tastendruck:
#define BUTTON_1    Play_Sound(28)      // spielt Soundfile 28 ab
#define BUTTON_2    B_U_Turn_CW()
#define BUTTON_3    B_Toggle_House()
#define BUTTON_4    B_Toggle_Flash()

// langer Tastendruck:
#define BUTTON_1_L  B_Home_Run()
#define BUTTON_2_L  B_U_Turn_CCW()      // U-Turn gegen den Uhrzeigersinn
#define BUTTON_3_L  B_Toggle_Sound()    // Sound on/off
#define BUTTON_4_L  Auto_Calibrate(1)   // führt die Kalibrierung des
Nullpunktes durch
```

From:
https://wiki.mobaledlib.de/ - **MobaLedLib Wiki**

Permanent link:
https://wiki.mobaledlib.de/anleitungen/bauanleitungen/locoturn_v10/150_locoturn_parameter?rev=1707576887

Last update: **2024/02/10 15:54**