```
//---------------------------------------------------------// DCC-Controlled-Kato-Turntable_v2.4
#include <DRV8835MotorShield.h>                            // Pololu DRV8835 Dual Motor Driver Shield for Arduino
#include <DCC_Decoder.h>                                   // Mynabay DCC library
#include <EEPROM.h>                                        // Standard Arduino EEPROM library
#define kDCC_INTERRUPT                   0                 // DCC Interrupt 0
DRV8835MotorShield               Turntable;                // Turntable Motor M1 = Bridge, Motor M2 = Lock
const uint8_t MAX_DCC_Accessories   =   13;                // Number of DCC Accessory Decoders
const uint8_t maxSpeed              =  120;                // Speed between -400 = Reversed to 400 = Forward (-5 to +5 VDC)
const uint8_t maxTrack              =   36;                // Total Number of Turntable Tracks
const uint8_t DCC_PIN               =    2;                // Arduino Output Pin  2 = DCC signal  = Interrupt 0
const uint8_t TURNTABLE_SWITCH_PIN  =    4;                // Arduino Output Pin  4 = Turntable Trigger = Cable Pin 1
                                                           // Arduino Output Pin 11 = Red LED     = Function Red
                                                           // Arduino Output Pin 12 = Green LED   = Function Green
const uint8_t LED_PIN               =   13;                // Arduino Output Pin 13 = Onboard LED = Bridge in Position
                                                           // Arduino Output Pin 14 = Yellow LED  = TURN 180
uint8_t EE_Address                  =    0;                // EEPROM Address storing Turntable bridge position
uint8_t Output_Pin                  =   13;                // Arduino LED Pin
uint8_t Turntable_Current           =    1;                // Current Turntable Track
uint8_t Turntable_NewTrack          =    1;                // New Turntable Track
int speedValue                      =    0;                // Turntable Motor Speed
int Turntable_NewSwitchState        = HIGH;                // New Switch Status (From HIGH to LOW = Turntable bridge in position)
int Turntable_OldSwitchState        = HIGH;                // Old Switch Status (HIGH = Turntable bridge not in position)
unsigned long Turntable_TurnStart   =    0;                // Start time to turn before stop
unsigned long Turntable_TurnTime    = 1000;                // Minimum time in ms to turn before stop
unsigned long Turntable_SwitchTime  =    0;                // Last time the output pin was toggled
unsigned long Turntable_SwitchDelay =    2;                // Debounce time in ms

const char* Turntable_States[] =                           // Possible Turntable States
{
  "T1CW",                                                  // Turn 1 Step ClockWise
  "T1CCW",                                                 // Turn 1 Step Counter ClockWise
  "TCW",                                                   // Turn ClockWise
  "TCCW",                                                  // Turn Counter ClockWise
  "T180",                                                  // Turn 180
  "STOP",                                                  // Stop Turning
  "POS",                                                   // Bridge in Position
  "MCW",                                                   // Motor ClockWise
  "MCCW",                                                  // Motor Counter ClockWise
  "NEXT"                                                   // Next Track
};

enum Turntable_NewActions                                  // Possible Turntable Actions
{
  T1CW,                                                    // Turn 1 Step ClockWise
  T1CCW,                                                   // Turn 1 Step Counter ClockWise
  TCW,                                                     // Turn ClockWise
  TCCW,                                                    // Turn Counter ClockWise
  T180,                                                    // Turn 180
  STOP,                                                    // Stop Turning
  POS,                                                     // Bridge in Position
  MCW,                                                     // Motor ClockWise
  MCCW,                                                    // Motor Counter ClockWise
  NEXT                                                     // Next Track
};

enum Turntable_NewActions Turntable_OldAction = STOP;      // Stores Turntable Previous Action
enum Turntable_NewActions Turntable_NewAction = STOP;      // Stores Turntable New Action
enum Turntable_NewActions Turntable_Action    = STOP;      // Stores Turntable Requested Action

typedef struct                                             // Begin DCC Accessory Structure
{
  int               Address;                               // DCC Address to respond to
  uint8_t           Button;                                // Accessory Button: 0 = Off (Red), 1 = On (Green)
  uint8_t           Position0;                             // Turntable Position0
  uint8_t           Position1;                             // Turntable Position1
  uint8_t           OutputPin1;                            // Arduino Output Pin 1
  uint8_t           OutputPin2;                            // Arduino Output Pin 2
  boolean           Finished;                              // Command Busy = 0 or Finished = 1 (Ready for next command)
  boolean           Active;                                // Command Not Active = 0, Active = 1
  unsigned long     durationMilli;                         // Pulse Time in ms
  unsigned long     offMilli;                              // For internal use // Do not change this value
}
DCC_Accessory_Structure;                                   // End DCC Accessory Structure

DCC_Accessory_Structure DCC_Accessory[MAX_DCC_Accessories];// DCC Accessory


void setup()
{
  Serial.begin(38400);
  Serial.println("DCC-Controlled-Kato-Turntable_v2.4");    // Show loaded sketch
  pinMode(TURNTABLE_SWITCH_PIN, INPUT);                    // Kato Turntable Pin 1
  pinMode(LED_PIN, OUTPUT);                                // Onboard Arduino LED Pin = Bridge in Position
  digitalWrite(LED_PIN,LOW);                               // Turn Off Arduino LED at startup
  pinMode(DCC_PIN,INPUT_PULLUP);                           // Interrupt 0 with internal pull up resistor
  DCC.SetBasicAccessoryDecoderPacketHandler(BasicAccDecoderPacket_Handler, true);
  DCC_Accessory_ConfigureDecoderFunctions();
  DCC.SetupDecoder( 0x00, 0x00, kDCC_INTERRUPT );
  for (int i = 0; i < MAX_DCC_Accessories; i++)
  {
    DCC_Accessory[i].Button = 0;                           // Switch off all DCC decoders addresses
  }
//  Turntable_Current = EEPROM.read(EE_Address);             // Read Turntable bridge position from EEPROM
} // END setup


void BasicAccDecoderPacket_Handler(int address, boolean activate, byte data)
{
  address -= 1;
  address *= 4;
  address += 1;
  address += (data & 0x06) >> 1;                           // Convert NMRA packet address format to human address
  boolean enable = (data & 0x01) ? 1 : 0;
  for(int i = 0; i < MAX_DCC_Accessories; i++)
  {
    if (address == DCC_Accessory[i].Address)
    {
      DCC_Accessory[i].Active = 1;                         // DCC Accessory Active
      if (enable)
      {
        DCC_Accessory[i].Button = 1;                       // Green Button
      }
      else
      {
        DCC_Accessory[i].Button = 0;                       // Red Button
      }
    }
  }
} // END BasicAccDecoderPacket_Handler


void DCC_Accessory_ConfigureDecoderFunctions()
{
  DCC_Accessory[0].Address        =   225;                 // DCC Address 225 0 = END, 1 = INPUT (For now both will stop Turntable)
  DCC_Accessory[0].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[0].Position0      =     0;                 // Turntable Position0 - not used in this function
  DCC_Accessory[0].Position1      =     0;                 // Turntable Position1 - not used in this function
// DCC_Accessory[0].OutputPin1      =    xx;                 // Arduino Output Pin xx = LED xx - not used in this function
// DCC_Accessory[0].OutputPin2      =    xx;                 // Arduino Output Pin xx = LED xx - not used in this function
  DCC_Accessory[0].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[0].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[0].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[1].Address        =   226;                 // DCC Address 226 0 = CLEAR, 1 = TURN 180
  DCC_Accessory[1].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[1].Position0      =     0;                 // Turntable Position0 - not used in this function
  DCC_Accessory[1].Position1      =     0;                 // Turntable Position1 - not used in this function
// DCC_Accessory[1].OutputPin1      =    xx;                 // Arduino Output Pin xx = LED xx - not used in this function
  DCC_Accessory[1].OutputPin2     =    14;                 // Arduino Output Pin 14 = Yellow LED
  DCC_Accessory[1].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[1].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[1].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[2].Address        =   227;                 // DCC Address 227 0 = 1 STEP CW, 1 = 1 STEP CCW
  DCC_Accessory[2].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[2].Position0      =     0;                 // Turntable Position0 - not used in this function
  DCC_Accessory[2].Position1      =     0;                 // Turntable Position1 - not used in this function
  DCC_Accessory[2].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[2].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[2].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[2].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[2].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[3].Address        =   228;                 // DCC Address 228 0 = Direction CW, 1 = Direction CCW
  DCC_Accessory[3].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[3].Position0      =     0;                 // Turntable Position0 - not used in this function
  DCC_Accessory[3].Position1      =     0;                 // Turntable Position0 - not used in this function
  DCC_Accessory[3].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[3].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[3].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[3].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[3].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[4].Address        =   229;                 // DCC Address 229 0 = Goto Track 1 , 1 = Goto Track 2
  DCC_Accessory[4].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[4].Position0      =     1;                 // Turntable Track 1
  DCC_Accessory[4].Position1      =     2;                 // Turntable Track 2
  DCC_Accessory[4].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[4].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[4].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[4].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[4].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[5].Address        =   230;                 // DCC Address 230 0 = Goto Track 3 , 1 = Goto Track 4
  DCC_Accessory[5].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[5].Position0      =     3;                 // Turntable Track 3
  DCC_Accessory[5].Position1      =     4;                 // Turntable Track 4
  DCC_Accessory[5].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[5].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[5].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[5].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[5].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[6].Address        =   231;                 // DCC Address 231 0 = Goto Track 5 , 1 = Goto Track 6
  DCC_Accessory[6].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[6].Position0      =     5;                 // Turntable Track 5
  DCC_Accessory[6].Position1      =     6;                 // Turntable Track 6
  DCC_Accessory[6].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[6].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[6].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[6].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[6].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[7].Address        =   232;                 // DCC Address 232 0 = Goto Track 7 , 1 = Goto Track 8
  DCC_Accessory[7].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[7].Position0      =     7;                 // Turntable Track 7
  DCC_Accessory[7].Position1      =     8;                 // Turntable Track 8
  DCC_Accessory[7].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[7].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[7].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[7].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[7].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[8].Address        =   233;                 // DCC Address 233 0 = Goto Track 9 , 1 = Goto Track 10
  DCC_Accessory[8].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[8].Position0      =     9;                 // Turntable Track 9
  DCC_Accessory[8].Position1      =    10;                 // Turntable Track 10
  DCC_Accessory[8].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[8].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[8].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[8].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[8].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[9].Address        =   234;                 // DCC Address 234 0 = Goto Track 11 , 1 = Goto Track 12
  DCC_Accessory[9].Button         =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[9].Position0      =    11;                 // Turntable Track 11
  DCC_Accessory[9].Position1      =    12;                 // Turntable Track 12
  DCC_Accessory[9].OutputPin1     =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[9].OutputPin2     =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[9].Finished       =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[9].Active         =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[9].durationMilli  =   250;                 // Pulse Time in ms

  DCC_Accessory[10].Address       =   235;                 // DCC Address 235 0 = Goto Track 13 , 1 = Goto Track 14
  DCC_Accessory[10].Button        =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[10].Position0     =    31;                 // Turntable Track 31
  DCC_Accessory[10].Position1     =    32;                 // Turntable Track 32
  DCC_Accessory[10].OutputPin1    =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[10].OutputPin2    =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[10].Finished      =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[10].Active        =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[10].durationMilli =   250;                 // Pulse Time in ms

  DCC_Accessory[11].Address       =   236;                 // DCC Address 236 0 = Goto Track 15 , 1 = Goto Track 16
  DCC_Accessory[11].Button        =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[11].Position0     =    33;                 // Turntable Track 33
  DCC_Accessory[11].Position1     =    34;                 // Turntable Track 34
  DCC_Accessory[11].OutputPin1    =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[11].OutputPin2    =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[11].Finished      =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[11].Active        =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[11].durationMilli =   250;                 // Pulse Time in ms

  DCC_Accessory[12].Address       =   237;                 // DCC Address 237 0 = Goto Track 17 , 1 = Goto Track 18
  DCC_Accessory[12].Button        =     0;                 // Accessory Button: 0 = Off (Red), 1 = On (Green)
  DCC_Accessory[12].Position0     =    35;                 // Turntable Track 35
  DCC_Accessory[12].Position1     =    36;                 // Turntable Track 36
  DCC_Accessory[12].OutputPin1    =    11;                 // Arduino Output Pin 11 = Red LED
  DCC_Accessory[12].OutputPin2    =    12;                 // Arduino Output Pin 12 = Green LED
  DCC_Accessory[12].Finished      =     1;                 // Command Busy = 0 or Finished = 1
  DCC_Accessory[12].Active        =     0;                 // Command Not Active = 0, Active = 1
  DCC_Accessory[12].durationMilli =   250;                 // Pulse Time in ms

  for (int i = 0; i < MAX_DCC_Accessories; i++)            // Configure Arduino Output Pin
  {
    if (DCC_Accessory[i].OutputPin1)
    {
      pinMode(DCC_Accessory[i].OutputPin1, OUTPUT);
      digitalWrite(DCC_Accessory[i].OutputPin1, LOW);
    }
    if (DCC_Accessory[i].OutputPin2)
    {
      pinMode(DCC_Accessory[i].OutputPin2, OUTPUT);
      digitalWrite(DCC_Accessory[i].OutputPin2, LOW);
    }
  }
} // END DCC_Accessory_ConfigureDecoderFunctions


void DCC_Accessory_CheckStatus()
{
  static int addr = 0;                                     // Begin loop through DCC Addresses
  DCC.loop();                                              // Loop DCC Library
  if (DCC_Accessory[addr].Finished && DCC_Accessory[addr].Active)
  {
    DCC_Accessory[addr].Finished = 0;
    DCC_Accessory[addr].offMilli = millis() + DCC_Accessory[addr].durationMilli;
    Serial.print("Address: ");
    Serial.print(DCC_Accessory[addr].Address);
    Serial.print(", ");
    Serial.print("Button: ");
    Serial.print(DCC_Accessory[addr].Button);
    Serial.print(" (");
    Serial.print( (DCC_Accessory[addr].Button) ? "Green" : "Red" ); // 0 = Red, 1 = Green
    Serial.print(")");
    Serial.println();
    switch (DCC_Accessory[addr].Address)
    {
      case (225):                                          // DCC Address 225 0 = END, 1 = INPUT
        if (DCC_Accessory[addr].Button == 0)               // Red Button    : 0 = END
        {
          Output_Pin = DCC_Accessory[addr].OutputPin1;     // Set Arduino Output Pin
          Turntable_NewTrack = Turntable_Current;          // Stop at current track
          Turntable_OldAction = STOP;                      // Action: Stop Motor M1
          Turntable_NewAction = STOP;                      // Action: Stop Motor M1
          Turntable_Action = STOP;                         // Requested Action = STOP
        }
        if (DCC_Accessory[addr].Button == 1)               // Green Button  : 1 = INPUT
        {
          Output_Pin = DCC_Accessory[addr].OutputPin2;     // Set Arduino Output Pin
          Turntable_OldAction = STOP;                      // Action: Stop Motor M1
          Turntable_NewAction = STOP;                      // Action: Stop Motor M1
          Turntable_Action = STOP;                         // Requested Action = STOP
        }
        break;

      case (226):                                          // DCC Address 226 0 = CLEAR, 1 = TURN 180
        if (DCC_Accessory[addr].Button == 0)               // Red Button    : 0 = CLEAR
        {
          Turntable_Current = 1;                           // Bridge in Home Position
          Turntable_NewTrack = 1;                          // Bridge in Home Position
          Output_Pin = DCC_Accessory[addr].OutputPin1;     // Set Arduino Output Pin
          Turntable_OldAction = Turntable_NewAction;
          Turntable_NewAction = STOP;                      // Action: Bridge in Position
          Turntable_Action = STOP;                         // Requested Action = STOP
        }
        if (DCC_Accessory[addr].Button == 1)               // Green Button  : 1 = TURN 180
        {
          Output_Pin = DCC_Accessory[addr].OutputPin2;     // Set Arduino Output Pin
          if (Turntable_Current < 19)
          {
            Turntable_NewTrack = Turntable_Current + (maxTrack / 2);
          }
          else
          {
            Turntable_NewTrack = Turntable_Current - (maxTrack / 2);
          }
          Turntable_OldAction = Turntable_NewAction;
          Turntable_NewAction = T180;                      // Action: Turn Motor M1 (maxTrack / 2) Steps
          Turntable_Action = T180;                         // Requested Action = T180
        }
        break;
      case (227):                                          // DCC Address 227 0 = 1 STEP CW, 1 = 1 STEP CCW
        if (DCC_Accessory[addr].Button == 0)               // Red Button   : 0 = TURN 1 Step ClockWise
        {
          Output_Pin = DCC_Accessory[addr].OutputPin1;     // Set Arduino Output Pin
          Turntable_NewTrack = Turntable_Current + 1;
          if (Turntable_NewTrack > maxTrack)               // From Track 36 to Track 1
          {
            Turntable_NewTrack = 1;                        // Track (1)
          }
          Turntable_OldAction = Turntable_NewAction;
          Turntable_NewAction = T1CW;                      // Action: Turn Motor M1 1 Step ClockWise
          Turntable_Action = T1CW;                         // Requested Action = T1CW
        }
        if (DCC_Accessory[addr].Button == 1)               // Green Button : 1 = TURN 1 Step Counter ClockWise
        {
          Output_Pin = DCC_Accessory[addr].OutputPin2;     // Set Arduino Output Pin
          Turntable_NewTrack = Turntable_Current - 1;
          if (Turntable_NewTrack = 0)                      // From Track 1 to Track 36
          {
            Turntable_NewTrack = maxTrack;                 // Track (maxTrack)
          }
          Turntable_OldAction = Turntable_NewAction;
          Turntable_NewAction = T1CCW;                     // Action: Turn Motor M1 1 Step Counter ClockWise
          Turntable_Action = T1CCW;                        // Requested Action = T1CCW
        }
        break;

      case (228):                                          // DCC Address 228 0 = Direction CW, 1 = Direction CCW
        if (DCC_Accessory[addr].Button == 0)               // Red Button   : 0 = Direction CW
        {
          Output_Pin = DCC_Accessory[addr].OutputPin1;     // Set Arduino Output Pin
          speedValue = maxSpeed;                           // Positive = Turn ClockWise
          SetDirection();                                  // Determine Turn ClockWise or Counter ClockWise
        }
        if (DCC_Accessory[addr].Button == 1)               // Green Button : 1 = Direction CCW
        {
          Output_Pin = DCC_Accessory[addr].OutputPin2;     // Set Arduino Output Pin
          speedValue = -maxSpeed;                          // Negative = Turn Counter ClockWise
          SetDirection();                                  // Determine Turn ClockWise or Counter ClockWise
        }
        break;

      default:
        if (DCC_Accessory[addr].Button == 0)               // Red Button   : 0 = Goto Track Position0
        {
          Output_Pin = DCC_Accessory[addr].OutputPin1;     // Set Arduino Output Pin
          Turntable_NewTrack = DCC_Accessory[addr].Position0;
          Turntable_Action = Turntable_NewAction;          // Requested Action = Depends on SetDirection
        }
        if (DCC_Accessory[addr].Button == 1)               // Green Button : 1 = Goto Track Position1
        {
          Output_Pin = DCC_Accessory[addr].OutputPin2;     // Set Arduino Output Pin
          Turntable_NewTrack = DCC_Accessory[addr].Position1;
          Turntable_Action = Turntable_NewAction;          // Requested Action = Depends on SetDirection
        }
        break;
    }
    Serial.print("DCC_Accessory_CheckStatus --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }
  if ((!DCC_Accessory[addr].Finished) && (millis() > DCC_Accessory[addr].offMilli))
  {
    DCC_Accessory[addr].Finished = 1;
    DCC_Accessory[addr].Active = 0;
  }
  if (++addr >= MAX_DCC_Accessories)                       // End loop through DCC Addresses
  {
    addr = 0;
  }
} // END DCC_Accessory_CheckStatus


void PrintStatus()
{
  Serial.print(Turntable_States[Turntable_Action]);
  Serial.print(": Old: ");
  Serial.print(Turntable_States[Turntable_OldAction]);
  Serial.print(", New: ");
  Serial.print(Turntable_States[Turntable_NewAction]);
  Serial.print(", Current: ");
  Serial.print(Turntable_Current);
  Serial.print(", NewTrack: ");
  Serial.print(Turntable_NewTrack);
  Serial.print(", Output_Pin: ");
  Serial.print(Output_Pin);
  Serial.print(", Speed: ");
  Serial.print(speedValue);
  Serial.println();
} // END PrintStatus


void loop()
{
  DCC_Accessory_CheckStatus();                             // Check DCC Accessory Status

  if (((millis() - Turntable_TurnStart) > Turntable_TurnTime) && (Turntable_NewAction != POS))
  {
    Turntable_CheckSwitch();                               // Check Kato Turntable Pin 1
  }

  if ((Turntable_OldAction != POS) && (Turntable_NewAction == POS))
  {
    if (Turntable_OldAction == MCW)                        // Move ClockWise
    {
      Turntable_Current = Turntable_Current + 1;
      if (Turntable_Current > maxTrack)                    // From Track 36 to Track 1
      {
        Turntable_Current = 1;                             // Track (1)
      }
      Serial.print("Loop: Check MCW           --> ");
      PrintStatus();                                       // Print Actions and Track Numbers
    }

    if (Turntable_OldAction == MCCW)                       // Move Counter ClockWise
    {
      Turntable_Current = Turntable_Current - 1;
      if (Turntable_Current = 0)                           // From Track 1 to Track 36
      {
        Turntable_Current = maxTrack;                      // Track (maxTrack)
      }
      Serial.print("Loop: Check MCCW          --> ");
      PrintStatus();                                       // Print Actions and Track Numbers
    }

    if (Turntable_Current == Turntable_NewTrack)           // Bridge in Position
    {
      Turntable_Stop();                                    // Motor M1 Stop
      Turntable_OldAction = Turntable_NewAction;
      Turntable_NewAction = STOP;                          // Action: Stop Motor M1
      Serial.print("Loop: Compare NewTrack    --> ");
      PrintStatus();                                       // Print Actions and Track Numbers
    }
    else                                                   // Bridge not in Position
    {
      Turntable_NewAction = Turntable_OldAction;
      Serial.print("Loop: Current is NewTrack --> ");
      PrintStatus();                                       // Print Actions and Track Numbers
    }
  }

  if ((Turntable_OldAction != T1CW) && (Turntable_NewAction == T1CW))
  {
    speedValue = maxSpeed;                                 // Positive = Direction ClockWise
    Turntable_MotorCW();                                   // Motor M1 Forward
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = MCW;                             // Action: Move Motor M1 ClockWise
    Serial.print("Loop: Check T1CW          --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }

  if ((Turntable_OldAction != T1CCW) && (Turntable_NewAction == T1CCW))
  {
    speedValue = -maxSpeed;                                // Negative = Direction Counter ClockWise
    Turntable_MotorCCW();                                  // Motor M1 Reverse
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = MCCW;                            // Action: Move Motor M1 Counter ClockWise
    Serial.print("Loop: Check T1CCW         --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }

  if ((Turntable_OldAction != T180) && (Turntable_NewAction == T180))
  {
    speedValue = maxSpeed;                                 // Positive = Direction ClockWise
    Turntable_MotorCW();                                   // Motor M1 Forward
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = MCW;                             // Action: Move Motor M1 ClockWise
    Serial.print("Loop: Check T180          --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }

  if ((Turntable_OldAction != TCW) && (Turntable_NewAction == TCW))
  {
    speedValue = maxSpeed;                                 // Positive = Direction ClockWise
    Turntable_MotorCW();                                   // Motor M1 Forward
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = MCW;                             // Action: Move Motor M1 ClockWise
    Serial.print("Loop: Check TCW           --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }

  if ((Turntable_OldAction != TCCW) && (Turntable_NewAction == TCCW))
  {
    speedValue = -maxSpeed;                                // Negative = Direction Counter ClockWise
    Turntable_MotorCCW();                                  // Motor M1 Reverse
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = MCCW;                            // Action: Move Motor M1 Counter ClockWise
    Serial.print("Loop: Check TCCW          --> ");
    PrintStatus();                                         // Print Actions and Track Numbers
  }
} // END loop


void DCC_Accessory_LED_OFF()
{
  for (int i = 0; i < MAX_DCC_Accessories; i++)
  {
    digitalWrite(DCC_Accessory[i].OutputPin1, LOW);        // LED OFF
    digitalWrite(DCC_Accessory[i].OutputPin2, LOW);        // LED OFF
  }
} // END DCC_Accessory_LED_OFF


void Turntable_Stop()                                      // Motor M1 Stop
{
  switch (Turntable_OldAction)
  {
    case MCW:                                              // Motor was turning ClockWise
      for (speedValue; speedValue > 0; --speedValue)       // Decrease speed to 0
      {
        delay(3);                                          // Delay to get better bridge to track position
        Turntable.setM1Speed(speedValue);                  // Motor M1 Speed 0
      }
      digitalWrite(LED_PIN, HIGH);                         // LED ON = Onboard Arduino LED Pin = Bridge in Position
      digitalWrite(Output_Pin, LOW);                       // LED OFF
//      EEPROM.update(EE_Address, Turntable_Current);        // Store Turntable bridge position into EEPROM
      break;
    case MCCW:                                             // Motor was turning Counter ClockWise
      for (speedValue; speedValue < 0; ++speedValue)       // Decrease speed to 0
      {
        delay(3);                                          // Delay to get better bridge to track position
        Turntable.setM1Speed(speedValue);                  // Motor M1 Speed 0
      }
      digitalWrite(LED_PIN, HIGH);                         // LED ON = Onboard Arduino LED Pin = Bridge in Position
      digitalWrite(Output_Pin, LOW);                       // LED OFF
//      EEPROM.update(EE_Address, Turntable_Current);        // Store Turntable bridge position into EEPROM
      break;
    case STOP:                                             // Immediate stop
      speedValue = 0;
      Turntable.setM1Speed(speedValue);                    // Motor M1 Speed 0
      digitalWrite(LED_PIN, HIGH);                         // LED ON = Onboard Arduino LED Pin = Bridge in Position
      digitalWrite(Output_Pin, LOW);                       // LED OFF
      EEPROM.update(EE_Address, Turntable_Current);        // Store Turntable bridge position into EEPROM
      DCC_Accessory_LED_OFF();
      break;
    default:
      break;
  }
} // END Turntable_Stop


void Turntable_MotorCW()                                   // Motor M1 Forward
{
  digitalWrite(LED_PIN, LOW);                              // LED OFF = Onboard Arduino LED Pin = Bridge in Position
  digitalWrite(Output_Pin, HIGH);                          // LED ON
//  speedValue = maxSpeed;                                   // Positive = Turn ClockWise
  Turntable.setM1Speed(speedValue);                        // Motor M1 Speed value
  Turntable_TurnStart = millis();                          // Time when turn starts
} // END Turntable_MotorCW
```

```
void Turntable_MotorCCW()                                  // Motor M1 Reverse
{
  digitalWrite(LED_PIN, LOW);                              // LED OFF = Onboard Arduino LED Pin = Bridge in Position
  digitalWrite(Output_Pin, HIGH);                          // LED ON
//  speedValue = -maxSpeed;                                  // Negative = Turn Counter ClockWise
  Turntable.setM1Speed(speedValue);                        // Motor M1 Speed value
  Turntable_TurnStart = millis();                          // Time when turn starts
} // END Turntable_MotorCCW


void Turntable_CheckSwitch()                               // From HIGH to LOW = Bridge in next position
{
  int SwitchState = digitalRead(TURNTABLE_SWITCH_PIN);
  if (SwitchState != Turntable_OldSwitchState)
  {
    Turntable_SwitchTime = millis();
  }
  if ((millis() - Turntable_SwitchTime) > Turntable_SwitchDelay)
  {
    if (SwitchState != Turntable_NewSwitchState)
    {
      Turntable_NewSwitchState = SwitchState;
      if (Turntable_NewSwitchState == LOW)
      {
        Turntable_OldAction = Turntable_NewAction;
        Turntable_NewAction = POS;                         // Bridge in next position
        Serial.print("Turntable_CheckSwitch     --> ");
        PrintStatus();                                     // Print Actions and Track Numbers
      }
    }
  }
  Turntable_OldSwitchState = SwitchState;
} // END Turntable_CheckSwitch


void SetDirection()
{
//  if (Turntable_NewTrack <= (Turntable_Current + (maxTrack / 2)))
  if (speedValue > 0)
  {
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = TCW;                             // Action: Turn Motor M1 ClockWise
  }
  else
  {
    Turntable_OldAction = Turntable_NewAction;
    Turntable_NewAction = TCCW;                            // Action: Turn Motor M1 Counter ClockWise
  }
  Serial.print("SetDirection              --> ");
  PrintStatus();                                           // Print Actions and Track Numbers
}
```