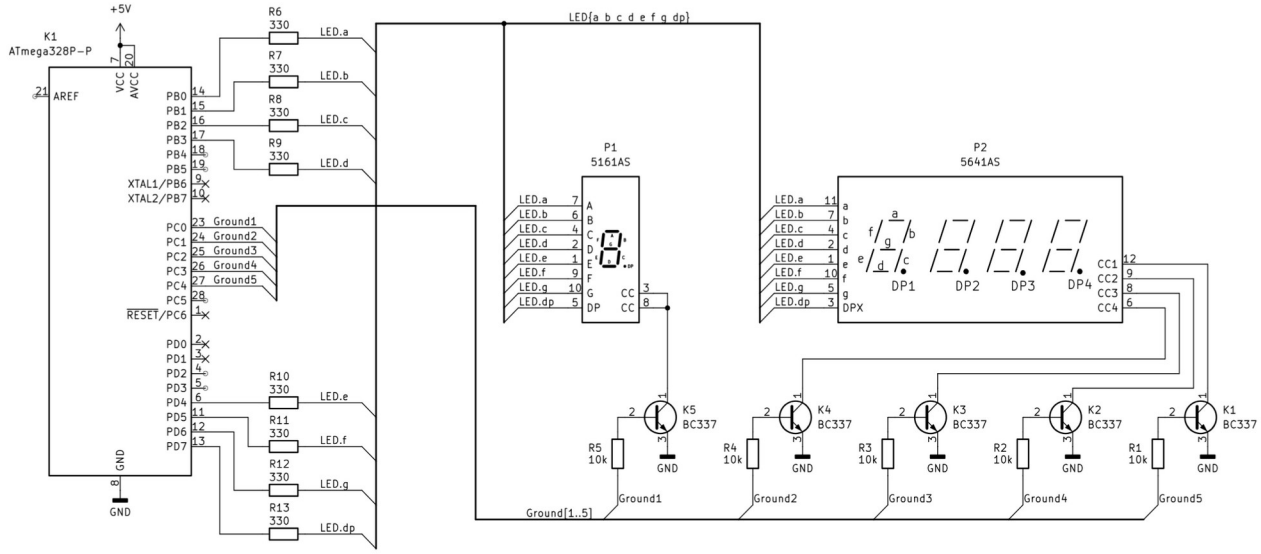



```
/*
 * 7 Segment Display Test mit 5161AS (1x Display) und 5641AS (4x Displays).
 * Beide Displays haben eine gemeinsame Kathode.
 * LEDs sind Positivaktiv
 * Vorwiderstand für LEDs ist jeweils 330 Ohm
 * Ein Schaltplan befindet sich im aktuellen Ordner.
 *
 * Pin Belegung
 * LED    5161AS    5641AS    Arduino    MC
 * A      7         11        8          PB0
 * B      6         7         9          PB1
 * C      4         4         10         PB2
 * D      2         2         11         PB3
 * E      1         1         4          PD4
 * F      9         10        5          PD5
 * G      10        5         6          PD6
 * DP     5         3         7          PD7
 * GND0             6         A0         PC0
 * GND1             8         A1         PC1
 * GND2             9         A2         PC2
 * GND3             12        A3         PC3
 * GND4   8                   A4         PC4
*/

// number_to_segments[TURN_OFF_DISPLAY] == Aus
#define TURN_OFF_DISPLAY 16
// Zeit zwischen Wechsel der Anzeigen. 3ms * 5 = 15ms entspricht ungefähr 60Hz
#define DISPLAY_TIME_MS 3
// Zeit zwischen Wechsel der LEDs in Test-Funktionen
#define TESTING_DELAY 200
```

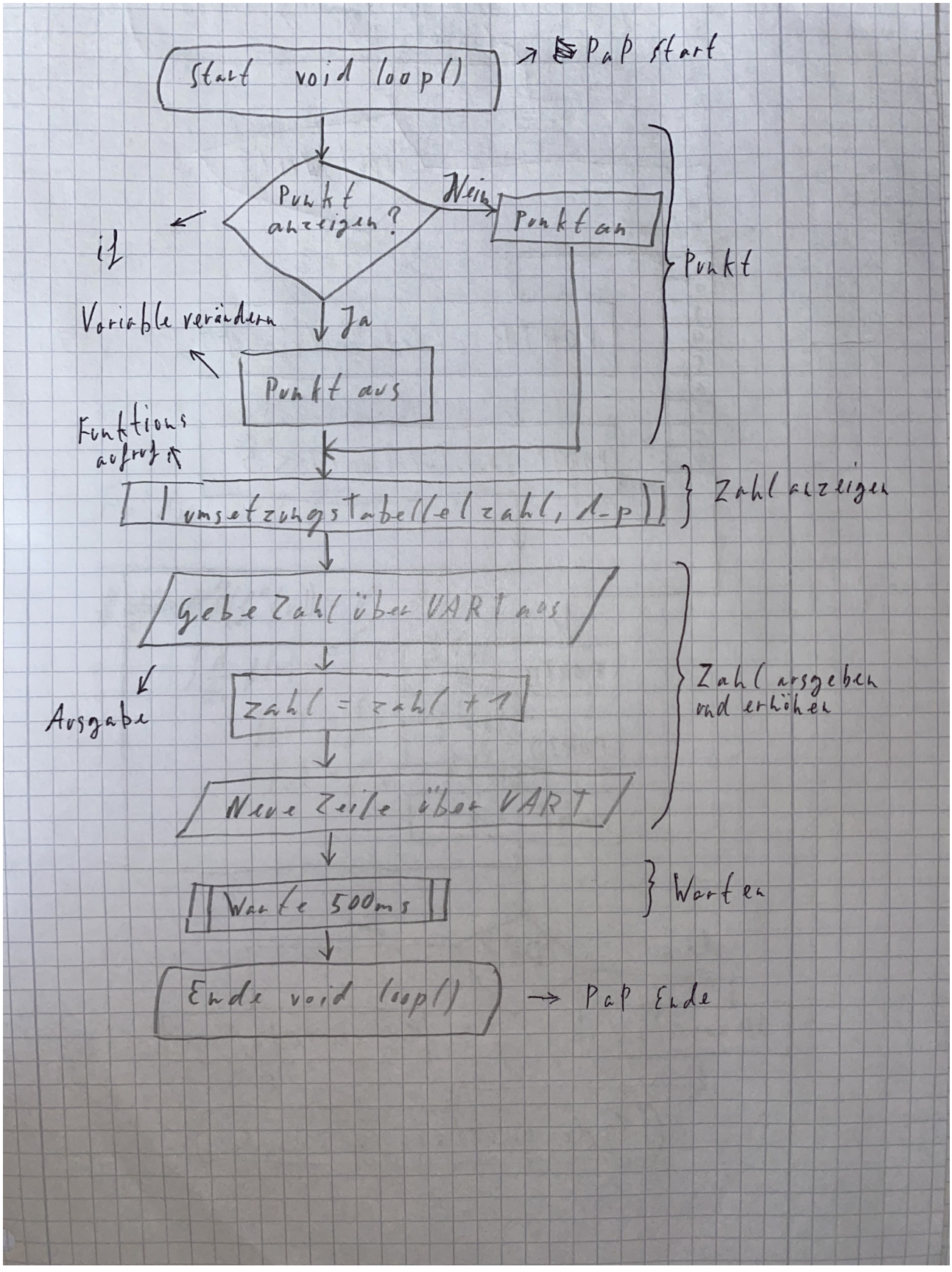
```
/* Zuweisungsliste: Hexadezimal (0 - 9, A - F) zu LED Zustände; [TURN_OFF_DISPLAY] => aus
 Format:
 0x[PD7 - PD4][PB3 - PB0]
 0b[DP,G,F,E,D,C,B,A]
*/
const uint8_t number_to_segments[] = {
  0x3F, // 0
  0x06, // 1
  0x5B, // 2
  0x4F, // 3
  0x66, // 4
  0x6D, // 5
  0x7D, // 6
  0x07, // 7
  0x7F, // 8
  0x6F, // 9
  0x77, // A
  0x7F, // B
  0x39, // C
  0x3F, // D
  0x79, // E
  0x71, // F
  0x00  // Aus
};
```

```c
/* Zeigt Zahl auf 7 Segment Display an. Unterstützt Hexadezimalzahlen.
 * @param decimal_place Aktiviert GND für einen jeweiligen PCx Pin
 * @returns 0 Erfolgreich; >1 Fehler
 */
uint8_t display_one_number(uint8_t number, bool decimal_point, uint8_t decimal_place)
{
  if (number > sizeof(number_to_segments) / sizeof(uint8_t))
  {
    return 1;
  }

  if (decimal_place > 4)
  {
    return 2;
  }

  // GND aktivieren; negativ Logik
  PORTC = ~(1 << decimal_place);

  if (decimal_point)
  {
    // Punkt anschalten
    PORTD |= 1 << PD7;
  }
  else
  {
    // Punkt ausschalten
    PORTD &= ~(1 << PD7);
  }

  // LED Zustand Code
  uint8_t segment_assignment = number_to_segments[number];

  // LED pins
  // Using PB0 - PB3
  PORTB = PORTB & 0xF0 | segment_assignment & 0x0F;
  // Using PD4 - PD6
  PORTD = PORTD & 0b10001111 | segment_assignment & 0b01110000;

  return 0;
}
```

```
/* Zahl von 0 bis 99999 auf 5 Segmenten Anzeigen
 * @param decimal_point_after_place Index des Displays wo Punkt angezeigt werden soll. Deaktivieren mit -1
 */
void display_number(uint16_t number, int8_t decimal_point_after_place, uint16_t show_for_ms)
{
  if (number > 99999)
  {
    return 1;
  }

  // Dezimalstellen auf die verschiedenen Anzeigen aufteilen
  uint8_t digit_1 = number % 10;
  uint8_t digit_10 = (number / 10) % 10;
  uint8_t digit_100 = (number / 100) % 10;
  uint8_t digit_1000 = (number / 1000) % 10;
  uint8_t digit_10000 = (number / 10000) % 10;

  // Nullen vor der ersten aktiven Anzeige ausschalten
  if (number < 1)
  {
    digit_10000 = TURN_OFF_DISPLAY;
    digit_1000 = TURN_OFF_DISPLAY;
    digit_100 = TURN_OFF_DISPLAY;
    digit_10 = TURN_OFF_DISPLAY;
    digit_1 = TURN_OFF_DISPLAY;
  }
  else if (number < 10)
  {
    digit_10000 = TURN_OFF_DISPLAY;
    digit_1000 = TURN_OFF_DISPLAY;
    digit_100 = TURN_OFF_DISPLAY;
    digit_10 = TURN_OFF_DISPLAY;
  }
  else if (number < 100)
  {
    digit_10000 = TURN_OFF_DISPLAY;
    digit_1000 = TURN_OFF_DISPLAY;
    digit_100 = TURN_OFF_DISPLAY;
  }
  else if (number < 1000)
  {
    digit_10000 = TURN_OFF_DISPLAY;
    digit_1000 = TURN_OFF_DISPLAY;
  }
  else if (number < 10000)
  {
    digit_10000 = TURN_OFF_DISPLAY;
  }
```

```
  // Über das Schalten von GND werden die verschiedenen Displays gesteuert.
  // Ungefähr +0.3ms für das Ausführen der Befehle.
  // Könnte mit interrupt besser gestoppt werden.
  for (uint16_t waited_for_ms = 0; waited_for_ms < show_for_ms; waited_for_ms += 5 * DISPLAY_TIME_MS)
  {
    display_one_number(digit_1, decimal_point_after_place == 0, 0);
    delay(DISPLAY_TIME_MS);
    display_one_number(digit_10, decimal_point_after_place == 1, 1);
    delay(DISPLAY_TIME_MS);
    display_one_number(digit_100, decimal_point_after_place == 2, 2);
    delay(DISPLAY_TIME_MS);
    display_one_number(digit_1000, decimal_point_after_place == 3, 3);
    delay(DISPLAY_TIME_MS);
    display_one_number(digit_10000, decimal_point_after_place == 4, 4);
    delay(DISPLAY_TIME_MS);
  }
}
```

```
// Von 0 bis 99999 Zählen
void test_multiple_displays(uint16_t start_number, uint8_t decimal_point_after_place)
{
  for (uint16_t current_number = start_number; current_number <= 99999; current_number++)
  {
    display_number(current_number, decimal_point_after_place, TESTING_DELAY);
  }
}

void setup()
{
  Serial.begin(9600);
  Serial.println("7 Segment Display");

  DDRB = 0x0F;
  DDRD = 0xF0;
  DDRC = 0x1F;

  // Ground Pins sind standardmäßig auf High
  PORTC = 0x1F;
}

void loop()
{
  // Punkt nach Einer anzeigen
  test_multiple_displays(0, 0);

  //test_one_display(0);
  //test_one_display(1);
  //test_one_display(2);
  //test_one_display(3);
  //test_one_display(4);
}
```


Start void loop()
PaP start
Punkt anzeigen?
Nein
Punkt an
if
Variable verändern
Ja
Punkt aus
Punkt
Funktions aufruf
[ ] umsetzungstabelle(zahl, 1_p)
Zahl anzeigen
Gebe Zahl über UART aus
Ausgabe
zahl = zahl + 1
Neue Zeile über UART
Zahl ausgeben und erhöhen
[ ] Warte 500ms
Warten
Ende void loop()
PaP Ende