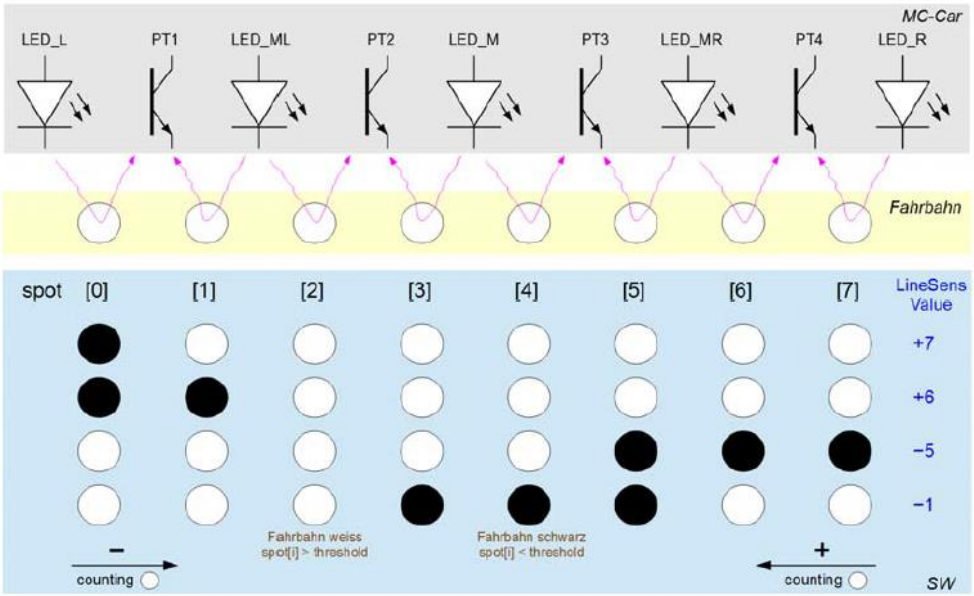
9: Analog-Digital Wandler

Sie verstehen die AD-Wandlung nach dem Prinzip der sukzessiven Approximation und können das AD-Wandlersystem im MC9S08JM60 zur Auswertung von analogen Signalen einsetzen

1. Liniensensoren des MC-Cars

Nachfolgend ist die geometrische Anordnung der 5 Infrarot-LEDs und 4 Phototransistoren des MC-Cars sowie der in SW implementierte Auswerte-Algorithmus zur Linienerkennung dargestellt.



2. AD-Wandlung

Legen Sie in CW ein neues C-Projekt mit Copy/Paste an und nutzen Sie das gegebene File main.c als Hauptdatei. Fügen Sie in der Bibliothek MC_Library zu den Verzeichnissen Lib_Headers bzw. Lib_Sources die gegebenen Files linesens.h und adc.h bzw. linesens.c zu (Drag/Drop/Copy).

Implementieren Sie im File adc_temp.c an den 4 gekennzeichneten Stellen (siehe CW Task-View) die fehlende Funktionalität, so dass das Auto der gegebenen Testlinie folgen kann.

adc.c

```
#include "platform.h"
#include "adc.h"


#define ADC_RES_8BIT      0
#define ADC_RES_10BIT     2
#define ADC_RES_12BIT     1


/**
 * Performs one A/D conversion for the specified channel
 * with 12 bit resolution. The function blocks until the
 * conversion has been finished.
 */
uint16 adcGet12BitValue(AdcChannels ch)
{
  ADCCFG_MODE = ADC_RES_12BIT;  // 12-bit conversion
  ADCSC1_ADCH = ch;             // select channel --> start conversion
  while (ADCSC1_COCO == 0) {}   // wait until conversion complete
  return ADCR;                  // converted value -> return result
}

/**
 * Performs one A/D conversion for the specified channel
 * with 10 bit resolution. The function blocks until the
 * conversion has been finished.
 */
uint16 adcGet10BitValue(AdcChannels ch)
{
  ADCCFG_MODE = ADC_RES_10BIT;  // 10-bit conversion
  ADCSC1_ADCH = ch;             // select channel --> start conversion
  while (ADCSC1_COCO == 0) {}   // wait until conversion complete
  return ADCR;                  // converted value -> return result
}

/**
 * Performs one A/D conversion for the specified channel
 * with 8 bit resolution. The function blocks until the
 * conversion has been finished.
 */
uint8 adcGet8BitValue(AdcChannels ch)
{
  ADCCFG_MODE = ADC_RES_8BIT;   // 8-bit conversion
  ADCSC1_ADCH = ch;             // select channel --> start conversion
  while (ADCSC1_COCO == 0) {}   // wait until conversion complete
  return ADCRL;                 // converted value -> return result
}


/**
 * Configures the A/D converter as follows:
 * - high speed mode with long sample time
 * - set ADCK = 6 MHz (bus clock / 4)
 * - enables the four line sensors
 */
void adcInit(void)
{
  APCTL1 = 0xF0;        // Enabled Pin 4..7 (line sensors)

  ADCCFG_ADLPC = 0;     // High speed Mode
  ADCCFG_ADLSMP = 1;    // long sample time
  ADCCFG_ADIV = 2;      // clock divide -> 4
  ADCCFG_ADICLK = 0;    // input clock -> Bus Clock
}
```