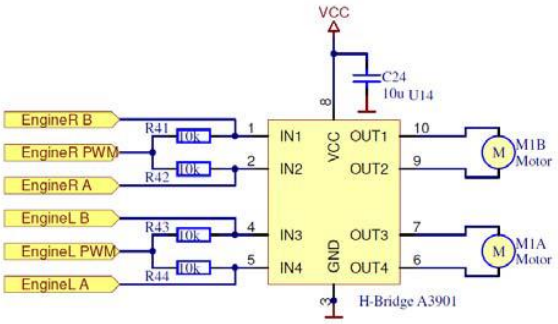
7.3: Timersystem im Edge-Aligned PWM Modus

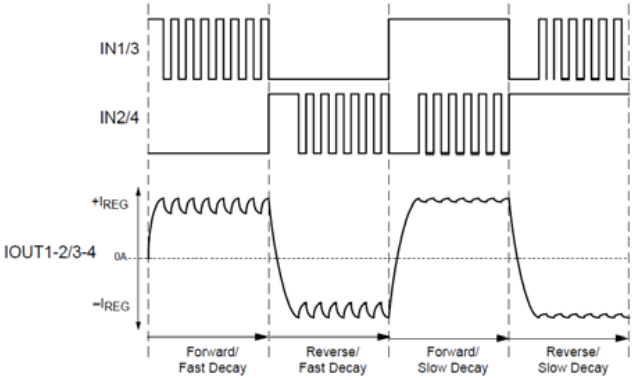
Sie verstehen die Betriebsart Edge-Aligned PWM des Timersystems im HCS08. Sie können das Timersystem zur Ansteuerung von Motoren einsetzen

1. Ansteuerung der DC-Motoren

Die beiden DC-Motoren des MC-Car werden über den H-Brücken Treiber A3901 angesteuert. Abbildung 1 zeigt den entsprechenden Ausschnitt aus dem Schema des MC-Car. Der A3901 erfordert die Generierung zweier PWM-Signale an seinen Eingängen IN1/3 für die Vorwärtsbewegung bzw. zweier PWM-Signale an seinen Eingängen IN2/4 für die Rückwärtsbewegung, siehe Abbildung 1 rechts. Da der eingesetzte MC9S08JM60 nicht genügend Timer-Kanäle besitzt, wurde eine Lösung mit insgesamt nur 2 PWM-Signalen und den 4 zusätzlichen Port-Pin Signalen EngineR/L_A/B implementiert.

Beispiel: Es soll der Modus Slow Decay des A3901 genutzt werden. Um in diesem Modus den rechten Motor (M1B) vorwärts drehen zu lassen, konfiguriert man EngineR_A als Input (Signal wird hochohmig) und treibt EngineR_B = 1 als Output. Die Motorgeschwindigkeit wird dann dem Duty Cycle des low-true PWMSignals EngineR_PWM entsprechen.





2. Timer mit Output Compare

Legen Sie in CW ein neues C-Projekt mit Copy/Paste an und nutzen Sie das gegebene File main_temp.c als Hauptdatei. Fügen Sie in der Bibliothek MC_Library zu den Verzeichnissen Lib_Headers bzw. Lib_Sources die gegebenen Files motor.h bzw. motor_temp.c zu.

Analysieren Sie den Code und implementieren Sie im File motor_temp.c die nötigen Konfigurationen für das Timer-Modul TPM2 (siehe CW Task-View), so dass die Ansteuerung der Motoren wie oben beschrieben funktioniert. Vervollständigen Sie das File main_temp.c, so dass man über die Tasten der Fernbedienung die Motordrehzahl in kleinen (+/−) bzw. grösseren Schritten (W/T) erhöhen und verringern kann.

main.c

```
#include "platform.h"   /* include peripheral declarations */
#include "ifrRx.h"
#include "motor.h"


/**
 * Switch on Rear LEDs on Port D2
 */
void initPorts(void)
{
    PTDDD = 0x04;
    PTDD = 0x04;
}


/**
 * TPM1: Counter running with frequency 1 MHz
 * - No TOF interrupt
 * - Modulo = default
 * - Prescale = 1
 */
void initTimer(void)
{
  TPM1SC = 0x10;
}
```

```
/**
 * main program
 */
void main(void)
{
  initPorts();             // Port init
  initTimer();             // Timer init
  ifrRxFrontInit();        // Infrared init
  motorInit();             // Motor init
  EnableInterrupts;        // Interrupts enable

  for(;;)
  {
    switch (ifrRxFrontGetKey())
    {
      case 'S' :
        motorSetPWMLeft(0);
        motorSetPWMRight(0);
        break;

      case 'W' :
        motorIncrementPWMLeft(20);
        motorIncrementPWMRight(20);
        break;

      case 'T' :
        motorIncrementPWMLeft(-20);
        motorIncrementPWMRight(-20);
        break;

      case '+' :
        motorIncrementPWMLeft(5);
        motorIncrementPWMRight(5);
        break;

      case '-' :
        motorIncrementPWMLeft(-5);
        motorIncrementPWMRight(-5);
        break;
    }
  }
}
```

motor.h

```
#ifndef MOTOR_H
#define MOTOR_H

int8 motorGetPWMLeft(void);
void motorSetPWMLeft(int8 value);
void motorIncrementPWMLeft(int8 value);

int8 motorGetPWMRight(void);
void motorSetPWMRight(int8 value);
void motorIncrementPWMRight(int8 value);

void motorInit(void);

#endif /* MOTOR_H_ */
```

motor.c

```
#include <stdlib.h>
#include <string.h>
#include "platform.h"
#include "motor.h"

#define MODULO          127 - 1 // Value 0..127 --> Module must be lower !!!

#define EngineR_A       PTDD_PTDD4
#define EngineR_B       PTDD_PTDD5
#define EngineL_A       PTDD_PTDD6
#define EngineL_B       PTDD_PTDD7

#define EngineR_A_Dir   PTDDD_PTDDD4
#define EngineR_B_Dir   PTDDD_PTDDD5
#define EngineL_A_Dir   PTDDD_PTDDD6
#define EngineL_B_Dir   PTDDD_PTDDD7

static int8 pwmLeft;
static int8 pwmRight;


/**
 * returns the pwm value of the left motor
 *
 * @returns
 *    the value 0..+/-127
 */
int8 motorGetPWMLeft(void)
{
  return pwmLeft;
}

/**
 * returns the pwm value of the right motor
 *
 * @returns
 *    the value 0..+/-127
 */
int8 motorGetPWMRight(void)
{
  return pwmRight;
}

/**
 * increments the speed of the left motor
 *
 * @param [in] value
 *    the desired offset
 */
void motorIncrementPWMLeft(int8 value)
{
  int16 v = pwmLeft + value;
  if (v > 127) v = 127;
  if (v < -127) v = -127;
  motorSetPWMLeft((int8)v);
}

/**
 * increments the speed of the right motor
 *
 * @param [in] value
 *    the desired offset
 */
void motorIncrementPWMRight(int8 value)
{
  int16 v = pwmRight + value;
  if (v > 127) v = 127;
  if (v < -127) v = -127;
  motorSetPWMRight((int8)v);
}

/**
 * sets the speed of the left motor
 *
 * @param [in] value
 *  +1..+127 => speed in forward direction
 *  -1..-127 => speed in backward direction
 *  0 => stop
 */
void motorSetPWMLeft(int8 value)
{
  if(value < 0)               // backward
  {
    EngineL_A_Dir = 0;        // EngineL B = Input = PWM
    EngineL_B_Dir = 1;        // EngineL A = Output
    EngineL_B = 1;

    TPM2C1V = abs(value);

  }
```

```
  else if(value > 0)          // forward
  {
    EngineL_A_Dir = 1;        // EngineL B = Output
    EngineL_B_Dir = 0;        // EngineL A = Input = PWM
    EngineL_A = 1;

    TPM2C1V = value;
  }
  else                        // stop
  {
    EngineL_A_Dir = 1;        // EngineR A = Output
    EngineL_B_Dir = 1;        // EngineL B = Output
    EngineL_A = 1;
    EngineL_B = 1;

    TPM2C1V = MODULO;
  }
  pwmLeft = value;
}

/**
 * sets the speed of the right motor
 *
 * @param [in] value
 *  +1..+127 => speed in forward direction
 *  -1..-127 => speed in backward direction
 *  0 => stop
 */
void motorSetPWMRight(int8 value)
{
  if(value < 0)               // rückwärts
  {
    EngineR_A_Dir = 1;        // EngineR A = Output
    EngineR_B_Dir = 0;        // EngineR B = Input = PWM
    EngineR_A = 1;

    TPM2C0V = abs(value);
  }
  else if(value > 0)          // vorwärts
  {
    EngineR_A_Dir = 0;        // EngineR A = Input = PWM
    EngineR_B_Dir = 1;        // EngineR B = Output
    EngineR_B = 1;

    TPM2C0V = value;
  }
  else                        // stop
  {
    EngineR_A_Dir = 1;        // EngineR A = Output
    EngineR_B_Dir = 1;        // EngineR B = Output
    EngineR_A = 1;
    EngineR_B = 1;

    TPM2C0V = MODULO;
  }
  pwmRight = value;
}

/**
 * Initializes the motor driver as follows:
 * - TPM2: Clocksource = 24 MHZ, Prescale = 4
 * - TPM2CH0: edge aligned pwm with low-true pulses
 * - TPM2Ch1: edge aligned pwm with low-true pulses
 */
void motorInit(void)
{
  EngineL_A_Dir = 1;
  EngineL_B_Dir = 1;
  EngineR_A_Dir = 1;
  EngineR_B_Dir = 1;

  EngineR_A = 1;
  EngineR_B = 1;
  EngineL_A = 1;
  EngineL_B = 1;

  // Clocksource = 24 MHZ, Prescale = 4
  TPM2SC_CLKSx   = 1;         // Clocksource = 24 MHZ
  TPM2SC_PS      = 2;         // Prescale = 4
  TPM2MOD        = MODULO;    // Module = 126 (MaxValue - 1)

  // Motor Rechts
  // TPM2CH0: edge aligned pwm with low-true pulses
  TPM2C0SC_MS0B  = 1;         // Edge-aligned PWM
  TPM2C0SC_ELS0A = 1;         // Low-true pulses (set output on compare)

  // Motor Links
  // TPM2Ch1: edge aligned pwm with low-true pulses
  TPM2C1SC_MS1B  = 1;         // Edge-aligned PWM
  TPM2C1SC_ELS1A = 1;         // Low-true pulses (set output on compare)
}
```