Controller feature:

1. Speed PI regulation.
2. Controller reset on over current. When controller reset times more than 20, shutdown inverter's output.
3. Shutdown on battery voltage is high and low, upper valve and lower valve value is stored in the calibration table.
4. Maximum current weaken on high temperature (when higher than 60 degree), shutdown on over temperature higher than 120 degree.
5. Protection on other exceptions, e.g. motor hall sensors connectivity detection, motor stall detection, throttle position detection on controller power on. When an exception happened, LEDs shows the error code.

Software process:

The controller software contains two routine: the main loop and the PWM interrupt routine. Period of the PWM interrupt loop is 60us (which is a little more than 16k Hz).

For the Freescale HCS08 microcontroller, lower priority interrupt routine can interrupt higher priority interrupt routine to run, so in the software only PWM interrupt is enabled, other interrupt like RS232 receive/send interrupt is disabled, polling method is used instead.

The controller is using complementary unipolar control mode (It is good for servo but not good for e-bike motor controller). The upper bridge is controlled by PWM, and the lower bridge is on state. Phase current is sampled on lower bridge indirectly (the phase resistance and inductance is given approximately), so the phase current is not accurate, the current sensor can be added though, but is not used by customer for saving money.
The PWM ISR:

The PWM ISR process: A/D sampling, speed PI operation, motor speed calculation by interrupt counting in one phase (so the speed feedback is not accurate), phase position changing detection and charging phase exchanging if needed.

The A/D sampling is processed at first using polling method. Phase currents, phase voltage, battery voltage, throttle and brake A/D and chip voltage are sampled:

```
ADC1SC1 = Index_AD; // AD channel
while(!ADC1SC1_COCO) // wait for AD complete
{
    _asm NOP; // wait AD
}
AD_Array[Index_AD++] = ADC1RL; // update AD
```

The realization of PI with anti-saturation of integration:

```
error = ref - feedback;
u = sum + kp * error;
if(u > outmax)
    out = outmax;
else if(u < outmin)
    out = outmin;
```
else
    out = u;
    exceed = kt * (out - u);
    sum += ki * i + exceed;

    For the floating point is not used, so in actually we scale the speed signal(0 ~ 600) multiply by 64, and divided by 64 before the duty cycle setting. The speed command signal is come from the main loop.

    There is a charge pump needed of the upper bridge driver transistor, so the duty cycle can not be setted up to 100%, the upper duty cycle is setted to 98%.

    the gain kp, ki, kt, outmax and outmin is stored within the calibration table. The gain kp and ki can be obtained via simulation model with matlab/simulink or Z-G method. (I'm familiar with control theory, and study matlab/simulink modelling of BLDC motor by myself, unfortunately I have not used it at work)

    When detecting the Hall signal changed, the HallProcess sub function is called. The main thing is setting the new high bridge output register, low bridge output register, and disable uncharged phase.

    In the HallProcess sub function, the Hall_Table is indexed by the hall signal value. When the value is 0 or 7, a hall exception happed and outputs shutdown.

    typedef struct
    {
        uint8_t Output_Phase;    // charge enable/disable by controlling the SD pin of IR2104
        uint8_t High_PWM_Address; // high bridge controlling register
        uint8_t Low_PWM_Address;  // low bridge controlling register
        uint8_t Close_PWMAddress; // uncharged bridge controlling register
        uint8_t Sampling_AD; // phase voltage sampling A/D channel ID
        uint8_t CCW_Sampling_AD; // phase voltage sampling A/D channel ID at running reverse
        // other definitions...
    } Hall;

    typedef struct
    {
        Hall Position[8];
    } Hall_Table;

    The main loop:
First chip initiation and input configuration (throttle type, regeneration enable/disable, throttle and brake dead-zone, etc.).

After initiation, the main loop process start:

1. Exception detection and handling.
   a) High temperature and over temperature exception handling. There are 2 weaken percentage on PCB temperature: 60/100 -- 70%, 100/110 -- 50%. Shutdown temperature 110 degree, shutdown resume temperature 100 degree. The weaken formula is:
      \[ \frac{\text{PCB\_Temp} - \text{Low\_Valve}}{\text{High\_Valve} - \text{Low\_Valve}} \times I_{\text{Max}}. \]
   b) The output is shutdown on low battery voltage (Vst), when voltage is lower than 112% of Vst, the maximum current is weakened, the output resume value is 106% of Vst. When sampling voltage is lower than Vst, output shutdown.

2. Throttle potentiometer A/D sub function is called. The throttle A/D value is readed with a first-order low pass filter. Then dead-zone process is calculated, the A/D value is scaled to [0, 600], which is the range of PWM duty cycle. At last the throttle position array (32 elements) is updated, the sum of all the array elements is shifted right by 5 as the speed reference value used by PWM ISR.