Framework of the Motor Control System

Introduction

This document is to refactor the existing Architecture: ***Chip S32K142***.

The architecture of the control system is designed according to the ***V-model*** and ***Layered Software Design***. The Simple V-model and the layered structure are shown in the figure 1and figure 2 respectively.

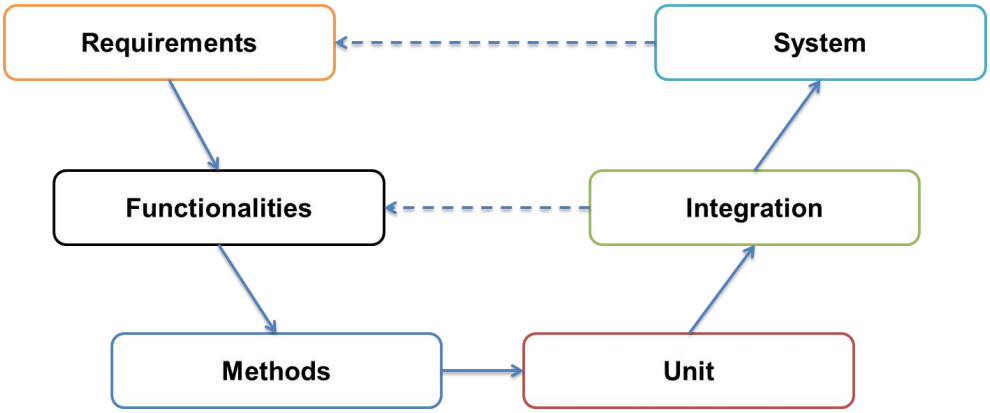

Figure 1 V-Model for Software Design

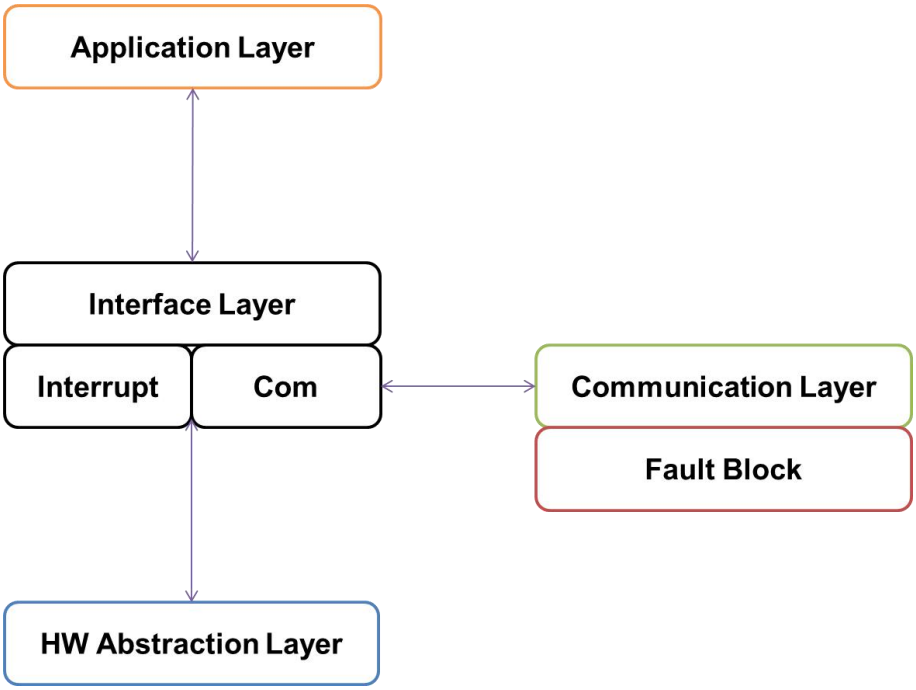

Figure 2 Layered Framework of the Control System

Layered Design

According to the pipeline, the layered software involves three parts: Hardware Abstract Layer (***HAL***), Interface Layer (***IL***), and Application Layer (***AL***).

In each layer, there may be ***workflow, dataflow,*** or ***stateflow*** (where necessary). Along the flow we can make the code clean and logic, sometimes we can refactor the whole control system by analysis.

There are some configurations of the Hardware in the ***HAL***: Clock, Pin, Interrupt Disable/Enable, Power, Trigmux, Hardware Initialization, PDB, ADC, DMA, UART, FTM, PWM, and Communication.

The blocks needed to be configured are shown below.

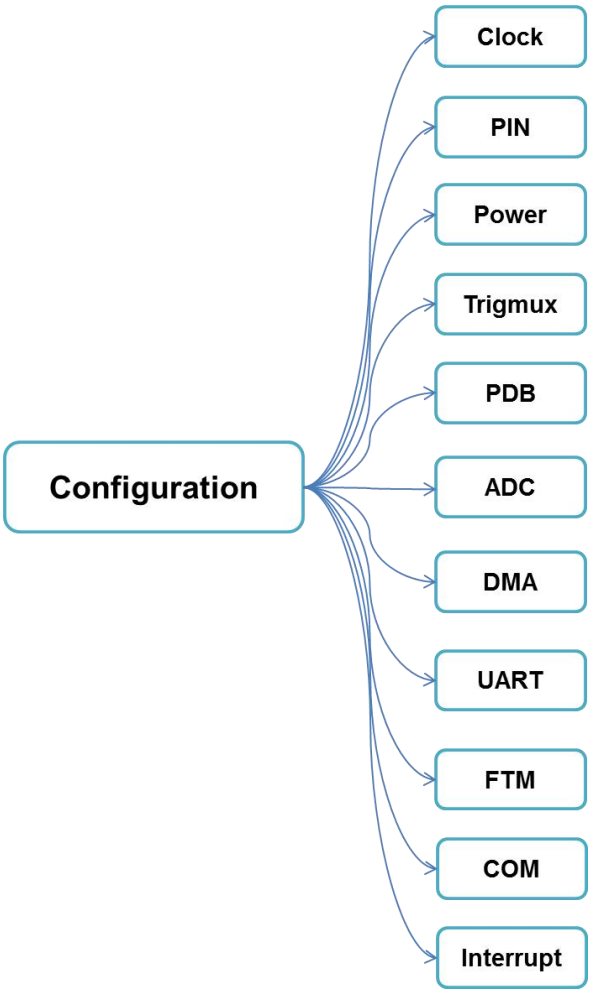

Figure 3 The Blocks needed to configured in the HAL

<HAL>

<CLOCK>

Name: McuclockConfig()

Description: This function initializes the system clock and other clocks.

<PIN>

Name: McuPinsConfig()

Description: This function initializes the PINs used.

<POWER>

Name: McuPowerConfig()

Description: This function initialize the POWER mode (HRUN, RUN, …)

<TRIGMUX>

Name: McuTrigmuxConfig()

Description: This function configures multi-triggers for FTM, PDB, etc.

<PDB>

Name: McuPdbConfig()

Description: This function configures PDB and pre-triggers for ADC.

<ADC>

Name: McuAdcConfig()

Description: This function configures ADC channels for sampling currents, voltage, Temperature, etc.

<DMA>

Name: McuDmaConfig()

Description: This function configures DMA for CAN and UART.

<UART>

Name: McuUartConfig()

Description: This function configures UART.

<FTM>

Name: McuFtmConfig()

Description: This function configures FTM/PWM and Interrupt mode.

Note: In order to integrate the code and for the convenience, Timeflow Interrupt and Reload Interrupt shall be modified and clarified.

<PWM>

Name: McuFtmConfig()

Description: This function configures PWM Outputs and operating mode

<CAN>

Name: McuCANConfig()

Description: This function configures CAN.

<Interrupt>

Name: intenable()

Description: This function configures FTM and ADC Interrupts, respectively.

<IL>

We are going to package all the configurations together to reduce the dependencies between layers. Hence, the interface functions between layers are **<Motor_Para_Init>**, **<intenable>**, and **<Communication_Service>**.

Therefore, the logic here is very straightforward. Along this pipeline, we minimize the dependencies between layers. Only the interface functions interact with each layer.

There are some interfaces to the AL in the IL. Here the interfaces are mostly structure variables which are links to AL. Phase currents and target frequency, for example.

Name: Init_MtrParaIni()

Description: This function assigns the values to the structured data.

```
void MtrPara_Init(void)
{
    int32_t *p;
    uint16_t lu16_i;
    p = (int32_t*)&ParaRam;
    for(lu16_i = 0u; lu16_i < TABLE_NUMBER; lu16_i++)
    {
        *p = (int32_t)TABLE_MTR_PARA[lu16_i];
         p++;
    }
}
```

Name: Ms_1_job()

Description: This is a call back function with a time base counting in the PWM Interrupt. This function involves the following functions.

Name: Inner_time1ms_mission()

Description: This is for LIN and to check if LIN is connected.

Name: Motor_Run_Time()

Description: This is a time counter that enables that each function is called at right time.

Name: CALCULATEP_I()

Description: This is to calculate the parameters like current, voltage, power, speed.

Name: Faultclear()

Description: This is to clear all the faults generated by the motor protection measures. Stop for 5s without doing anything then set all the fault-related bits to zeros.

Name: Start_after_2s_job()

Description: This is to increase or decrease to the target speed when switched to closed loop after 2s and check if SVPWM is in phase Shift mode.

Name: Ms_100_job()

Description: This is a call back function with a time base counting in the PWM Interrupt. The function involves the following functions.

Name: Freqset_with_condition()

Description: This is to set the frequency for the motor under the condition that no error is occurred.

Name: Can_send_id_change_count()

Description: This is to interchange the CAN_ID for Senddata() every 0.1s.

Name: motorParaChange()

Description: This is to change the LD and LQ online such that the motor can run smoothly at high speed.

The following functions will be communication-related. Here we focus on the LIN communication which is double_ communication() in the code.

Name: Communication_service()

Double_communicaiton()

Description: This is to receive the fault-clearing bit, the power-on bit, and the Target speed and transmit the state of the motor and parameters required via UART.

Name: Communication_Drive()

InCom_Uart_Drive()

Description: This is to realize the first RX and alternatively RX and TX.

There are two interrupt functions. These functions belong to AL.

Name: Intenable()

Description: This is to enable the interrupt functions required and call the ADC sampling function and output the updating PWM.

Name: ftmTimerISR()

Description: This is to calculate the duty cycles for three phases.

Int_PWM_after_shakehand()

Name: TrqComp()

Description: This is to calculate the amount for torque compensation.

Name: Fault_Warning_Detect()

Description: This is to detect whether fault occurs and stop the motor if necessary.

Name: TimeBaseCount()

Description: This is time counter.

Name: Int_AD_GetPhaseCurrent()

Description: This is to get offset of sample currents.

Name: Motor_state_machine()

Description: This is a state machine for motor control. In each case, do the corresponding action.

Name: adc1isr()

Description: This is to get the values from the ADC channels. Hence, the phase currents, DC bus voltage, IPM temperature, and PCB Temperature. All of these values

are scaling values and the range is 0~4096. adcRawValue_n to indicate the corresponding parameters.

<AL>

The first thing is to initialize the corresponding parameters. Here we adopt a strategy where values in the constant table are assigned to the structured data, and then assign the structured data to global variables.

Name: Init_Drive_HardwareandSoftware()

Description: This function assigns the values to the global variables for intended use.

Name: Init_MotorPara()

Description: This function assigns the values to the motor parameters (Resistance, Inductance, Poles, and Flux Linkage).

The idea here is to assign the values in the table to structure-ParaRam, then the assign the values to global variables where needed. Initialize the motor parameters independently.

Name: Motor_reset()

Description: This is to initialize or reset the structured data which are interface variables.

Name: ALLlowpass_init()

Description: This is to reset all the variables going through Lowpass Filter.

Name: Motor_U_I_Angle_clear()

Description: This is to reset current, voltage, and theta.

Name: Init_SVPWM()

Description: This is to initialize the SVPWM variables.

Name: Init_FLUXOBSERVER()

Description: This is to initialize the flux on the alpha and beta axes.

Name: Init_MtrAcr()

Description: This is to set PI for Current Loop, and Clear the flag for integral saturation.

Name: Init_MtrAsr()

Description: This is to set PI for Speed Loop and initialize lowpass filter for Omega.

Name: Init_MtrFwc()

Description: This is to set FWK_LVL and FWK_MAX and PI for the Field weakening control loop.

Name: Init_MtrPll()

Description: This is to set PI for Phase-locked Loop and some lowpass filters for motor speed and set the actual frequency.

Name: OpenandParkSet()

Description: This is to initialize the structured data for both openloop and parkloop.

Name: CarrierDeadCalc()

Description: This is to calculate the Carrier ticks and dead time for u, v, w phase and PWM frequency, and Time Base for other functions call.

Name: Init_DeadtimeComp()

Description: This is to initialize the Dead Time compensation for u,v,w phase.

Name: Intenable()

Description: This is to initialize interrupts and set the priorities for the corresponding interrupts.

Name: Communication_Init()

Description: This is to initialize CAN or UART.

Name: INT_SYS_EnableIRQGlobal()

Description: This is to enable the Global Interrupt.

In order to conveniently switch two PWM modes, here we are going to use a global variable (enumeration type) to indicate which mode is selected.

For example: gu8_PWM_Freq_Sel = 1, 2, 3, (1 = 8k, 2 = 16k), gu8_HS_nPWM = 1, 2,

```
typedef enum
{
    FTM3_Ovf_IRQn       = 0x00U,     /*!< Overflow interrupt */
    FTM3_Reload_IRQn    = 0x01U      /*!< Reload interrupt */
} ftm_INT_option_t;

ftm_INT_option_t FTM3_INT_Sel;      /*!< Select the Interrupt type */
```

AL is the core part which is employed to drive and control the motor such that control the electric compressor. The whole control system is based on the Field-oriented control algorithm. The idea is statistical estimation and inference where speed and position of the motor are estimated.

In the AL, the key component will be state machine which is the core control algorithms. The control algorithm can be seen as a state machine: PRECHARGE, RUN, STOP, ERROR. There are four states.