Analysis and Design of Automotive Body Control Module

Jianhui Ma, Zhixue Wang, Yanqiang Li and Liangjie Yu

Abstract In the BCM's industrialization process, we need design appropriate BCM for different car models. In order to reduce the complexity of the design while avoiding duplication of design work, This paper summarize the experience in the recent years design of the BCM, analyze the system structure, working mechanism, and basic design principles based on one particular BCM section, analyze its external interfaces attributes and complex control logic. Design generic, common embedded software structures for body control and basic modules that can be configured and assembled, These software components is flexible and configurable, based on the software structure and basic module library, we can quickly start the development of appropriate body controller software for BCM of different car models.

Keywords BCM · Software architecture · Portability

1 Introduction

Along with the development of automotive electronics and networking technology widely used in automobiles, Body Control Module (BCM) integrated body network gateway with lighting control, wiper control, window control, RKE key access control and door lock control function, is becoming a mainstream.

F2012-D01-005

J. Ma (✉) · Z. Wang · Y. Li · L. Yu
Shan Dong Key Laboratory of Automotive Electronics,
Automatic Institute of Shan Dong Academy, Jinan 250014, Shan Dong, China
e-mail: majialong@yahoo.com.cn

SAE-China and FISITA (eds.), *Proceedings of the FISITA 2012 World Automotive Congress*, Lecture Notes in Electrical Engineering 194,
DOI: 10.1007/978-3-642-33829-8_3,

Although the BCM have many input and output interfaces, and complex control logic, but the BCM of different car models is basically same in its works and system structure, only have certain differences in module combination or some specific module's design, so it is necessary to analyze the system structure and basic design principles of BCM based on one particular section, design generic, common embedded software structures for body control and basic modules that can be configured and assembled. Based on the software structure and basic module library, we can quickly start the development of appropriate body controller software for BCM of different car models, reduces development complexity and improves the development efficiency. This software must have following features.

1. Using scalable reactive software architecture, adding new features without breaking existing software structure, and won't influence the behavior of existing systems;
2. Establish an effective relationship between reuse and assembly, in the development of new BCM module, avoid duplication of development of basic software modules, and avoid increased costs and extended development cycle;
3. The software interface standards of the universal basic module have uniform agreement, ensuring the independence of modules and portability at the level of applications;

Combined with the development of BCM for a car model, the author analyzed the design principle of BCM and the specific implementation from BCM system structure, software architecture and the realization of part of the module.

2 System Structure

BCM is a typical body central controller combined centralized control and distributed control, its input interface includes a series of switch signals and driving pulse signal, output interface is a series of control objects that includes locks, lights, wipers, windows and alarm. At the same time, BCM communicates with remote control keys by RF signal, and exchange information of control command and status with sensor nodes and windows node via LIN bus. its external interface shown in Fig. 1.

BCM is a typical control system, through the detection of the switching signal and pulse signal and a series of combinational logic, achieve the load drive control, also achieved RKE Keyless entry and anti-theft alarm function. At the same time, as a Body Control LIN network master node, BCM scheduling the entire LIN network communication and network management. Its system structure show in Fig. 2, the output control is the core module of BCM, other function module also include input signal detection, LIN communication, the RKE communication, anti-theft alarm state management.

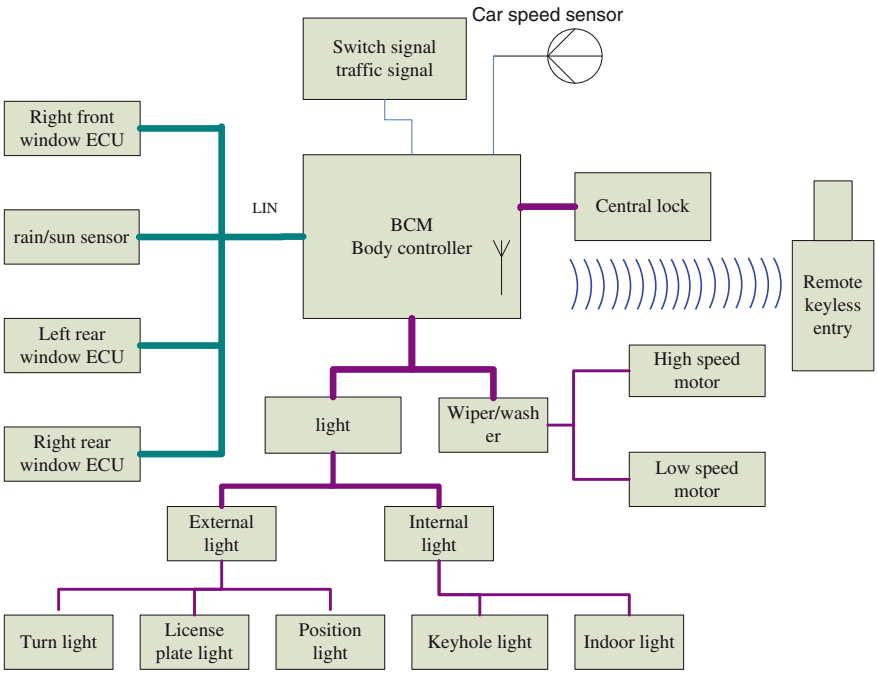

Fig. 1 BCM external interface diagram

3 Software Design

According to BCM design features and design resource requirements, while taking into account the cost factor, the author chose the Freescale 16-bit automotive-grade MCU to achieve its software design. As a basic software and design solutions of a series of BCM, this paper analyzes the BCM software architecture design and the programming of some modules, explain how to ensure software scalability and module reusability from the system level and the micro level. In the following, first analyzed BCM software architecture design, and then describe timer management and switch signal detection these basic module's realization.

3.1 Software Structure

In order to save the limited resources of MCU, the BCM software design does not use the operating system, and because different car model BCM's input detection, output control, communication and control logic is or less the same, it is necessary and feasible to design a common body control module software architecture. Based on the software architecture solutions, develop appropriate body controller

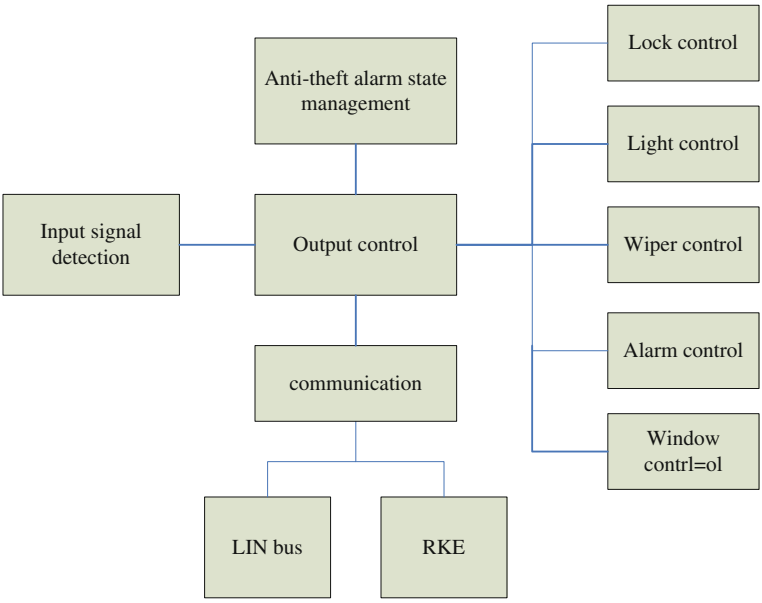

Fig. 2 BCM system structure

software for different models, improve ECU software reliability and development efficiency, reduce development complexity.

The software structure is in the form of interrupt + the main loop body. The system enters main loop after power-on initialization, the body of the loop including the following module: timer management, input signal detection and statistic, signal reception and the extraction of LIN application layer, RKE communication, anti-theft alarm status management, output control (including the window control, wiper control, door lock control, alarm control, light control), fill LIN send signals, clear event. The order of these modules in the loop is very important, reflecting the working principle of the BCM, the main body of the loop as follows:

```
for(;;)
{
  TimerTick();
  InputDetect();
  l_SignalDetect();
  Rke_Decrypte();
  AlarmStateManage();
  WindowControl();
  WiperControl();
  LockControl();
  AlarmControl();
```

```
  LightControl();
  l_app();
  l_Com();
  ClearEvent();
}
```

First, timing information is the input signal of all other modules, so put the timer management on the top of the loop body, and then, because switch control is BCM's main control logic, so followed by is the switch signal detection in the main loop. Input signal of other control logic is pulse, LIN signal and RKE signal. LIN data link layer is achieved in the UART receive interrupt service routine, and application layer signal receiving part in l_SignalDetect. Rke_Decrypte achieve remote control key's learning, the RKE key signal detection and statistics. The input signal of Alarm state management is switch signal and RKE signals, and also it is the input signal of the output control logic, so placed Alarm State Manage after Input Detect and Rke_Decrypte, followed by the output control and the the filling of the LIN send signal.

All the control modules are event-driven, if an event occurs, then perform the appropriate control logic. Since many events are shared, in order to ensure the event to digest more than one module, put clear event operation—Clear Event on the end of the main loop. Due to certain events is set in the interrupt, then it is need to introduced the concept of synchronization in logic circuit design to software architecture design, treat each entry of the main loop as a synchronous clock, the event set in the interrupt is synchronized in the main loop, thus avoiding instability that the event is cleared before been digested (Fig. 3).

3.2 Basic Module Design

3.2.1 Timer Management

BCM timing applications include statistics of input signal time characteristics, the output logic timing and timeout handling, LIN master node schedules the rotation of the time slice and IDLE time detection, which is characterized by the timing accuracy is not required, but the timing number is more, based on these features, designed timer management module.

Due to limited hardware timer and range of timing application, can't assign hardware timer for each timing applications, so use software timer simulate hardware timer. According to the timing characteristics and classification of the application, design software timer data structure in the form of structure, organized these software timer in the form of a structure array, the array member is software timer node. As all software timer reference clock source, the hardware timer is set to 1 ms cycle timing, manage the hardware timer in the interrupt service routine-cumulative global clock tick Jiffs, set clock synchronization flag TimerTicked to 1. In the main loop, function

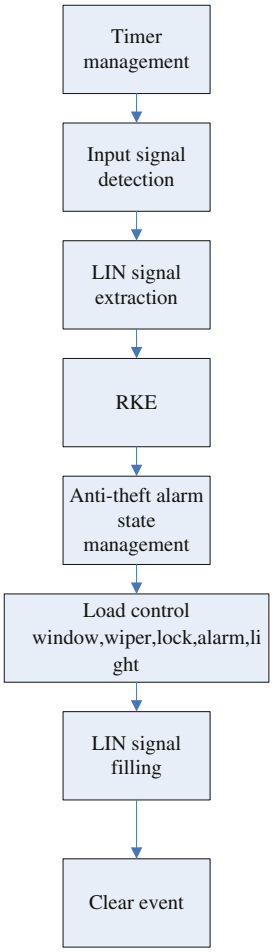

Fig. 3 BCM main loop software flow chart

TimerTick performed all the software timer management according to TimerTicked and Jiffs. So achieved simulate multiple software timer by a single hardware timer.

Software timer data structure is designed as follows:

```
typedef struct {
  TimerState timer_state;
  ulong timeout;
  ulong duration;
  unsigned cycle:1;
  unsigned cnt_times:8;
  unsigned overflow_flag:1;
  TimerId timer_id;
}Timer;
```

"Timer_state" means if a software timer is in running condition, "timeout" is overtime application's timeout threshold, "duration" is the timing of the software timer since its launch, "cycle" indicates whether the periodic timing, "cnt_times" indicates times of multiple timing when not a periodic timing, "overflow_flag" indicate whether application timeout occurs, "timer_id" is used to identify a software timer in the software timer array. Thus, the member variable describes all the "timing features" and provides a good read-write interface.

In function TimerTick of main loop, manage multiple software timers in order. The software timer only runs when tick occurs under the circumstances of their own status as RUNNING, its "duration" accumulate with the tick, when "duration" matches its timeout value, set overflow_flag, and then determine whether it is a cycle timer. If it is a cycle timer, restart the timer and clear the "duration", if not, determine whether multiple timing, to determine whether to restart the timer or stop the timer.

3.2.2 Switch Signal Detection

The switching signal detection is relatively simple in the hardware design, just current limiting + filter + voltage divider, then detect with MCU IO pin. In program design, need to determine switch current state and its changes. Because BCM needs to collect so much switch signal, that in order to program simple with clear logic, define a structure to unify each switch signal, structure is defined as follows:

```
typedef struct{
  unsigned switch_state:1;
  unsigned swon_event:1;
  unsigned swoff_event:1;
  unsigned cursw:1;
  uchar detect_cnt:3;
  e_SwId switch_id;
}s_Switch;
```

In the above structure, "switch_state" defined the current state of the switching signal, "swon_event" said switch changes from disconnected to connected, and "swoff_event" said switch changes from connected to disconnected, "cursw" and "detect_cnt" used in switch signal software debounce function.

In specific application, define a s_Switch structure array Sw[MAX_SWITCH], each switch corresponding to a structure variable, addressing with the member variable "switch_id" in above structure, the "switch_id" is defined as follows with enumerate type:

```
typedef enum{
```

```
  IGNITIONKEY_SWITCH,
  IGNITION_SWITCH,
  COLLISION_IO_SWITCH,
  SPEED_IO_SWITCH,
  FRONTDOORKEY_LOCK_SWITCH,
  FRONTDOORKEY_UNLOCK_SWITCH,
  LEFTFRONT_DOOR_SWITCH,
  RIGHTFRONT_DOOR_SWITCH,
  ….
  ….
}e_SwId;
```

So that if the status of the left front door switch is used, directly use Sw[LEFTFRONT_DOOR_SWITCH].switch_state, and if you want change of its status, directly use Sw[LEFTFRONT_DOOR_SWITCH].swon_event and Sw[LEFTFRONT_DOOR_SWITCH]. swoff_event. Specific procedures are not discussed here.

4 Conclusion

According to characteristics and working principle of BCM, analyzed the external interface and system architecture, designed a common software structure and basic module that has been applied successfully in the software design of a BCM for one car model, It has steady performance in the real vehicle test, with practical value and significance.