The basics of automotive cluster device architectures and applications, part I

Deepak Mahajan, Vikas Agarwal, Arjun Pal Chowdhury - December 01, 2014

With the increased complexity of vehicle electronics, greater functionality requires status information to be displayed to the driver. The instrument cluster is the primary data source for the driver, delivering information about vehicle and engine status. Given system complexity, however, there is greater demand for a more user friendly, lucrative and cost-effective solution to support a wide range of automotive cluster applications. Here we will discuss various components of a cluster device that enable this support. The article classifies cluster architecture and applications into the following broad categories:

- Types of cluster devices
- Autosar application component
- Graphics application component
- Cluster security
- Device memory requirement
- Low power mode cluster architecture

Cluster Device Types

There are three basic cluster device platforms, including:

Cluster Graphics Platform

Car manufacturers and tier-1 suppliers are facing an increasing need for content to be displayed to the driver requiring instrument cluster solutions that can be dynamically reconfigured to display content based on context and driver preferences while presenting required information to ensure safe driving. Cluster graphics Platform is mainly used to fulfill the requirement. This device mainly consists of Graphics processor with 2D/3D accelerator, huge on-chip graphics RAM for high bandwidth graphics data access, high-speed external memory interface like SDR,DDR or Nand Flash Controller, display controller to display graphics information through TFT,LCD display. In addition, the set of integrated peripherals such as CAN, MOST MLB, LVDS, Ethernet and USB allow a direct connection to the rest of the car network. The device is generally used for high end cluster application by OEMs.

Cluster Controller Platform

This is the cluster MCU device that mainly handles the Cluster AUTOSAR stacks functionality. This device contains a stepper motor controller and driver for gauge driving, sound generation module, PWM channels for sound and cluster LED control, various analog sensors and communication
peripherals, HMI to communicate with driver, other body devices and gateway devices or other subsystem. The device can also have segmented LCD or even dot matrix LCD displays as an alternative display solution for low end cluster application. For high/mid range applications, most of the real time data and status captured by the device is communicated to cluster graphics device through fast SPI or EBI interface.

**Combined Cluster Platform**

The graphics controller and real time application controller (Cluster MCU) is combined in this platform to make single chip solution for both Graphics and Autosar application. This device keeps a balance between graphics performance and real time performance hence used by OEMs when optimum solutions are required. Some renowned semiconductor companies offer wide range of combined cluster platform microcontrollers with multi core based solutions to support from basic to premium line instrument cluster applications.

![Figure 1: A block diagram of a highly integrated Instrument Cluster Architecture](image)

**Module Use-Case**

1. **AUTOSAR Application Components**

   **Core:** The core is part of real-time application domain of the device. It is mainly used to run AUTOSAR applications such as basic a communication driver, PWM driver, handling and servicing of various peripherals etc. The high bandwidth motor control application driver can also be controlled through this core. The device can also have a dedicated core for motor control applications, which helps to share the bandwidth of the real time processor.

   **Stepper Motor Control/Driver:** In cluster application a stepper motor driver is mainly used to control gauges. Real-time information of vehicle speed, fuel level, tachometer, temperature etc., is communicated to cluster devices through a gateway device. The core then processes that information and updates the motor control driver to control the stepper motor. Two motor control
channels are mainly used to drive a single stepper motor.

**PWM:** PWM outputs are used to control the backlighting of LCD displays, gauge backlighting etc. The LED intensity of dashboard, odometer, etc., are controlled by PWM output. For low-end applications where few LED tell-tales are required, it is controlled by microcontroller PWM output. However in mid-range or high-range applications where more LEDs need to drive the system, there is typically a dedicated LED driver.

**ADC and Alarm Comparator:** These analog components are mainly used to monitor the current, voltages of various power supplies. They are also used to measure the current flow through Telltale LEDs to check that everything is working fine. Measuring the battery level, ignition level, air temperature and coolant are common ADC functions performed by either a cluster or body device. When performed by a body device, it is communicated to a cluster by using a LIN or CAN connector.

**Sound Generation Module:** The device can have sound generation module with I2S protocol support, mainly to produce a buzzer sound in a cluster to indicate the occurrence of some unwanted event. For example, a driver seat belt warning detected by a body-device sensor is communicated to a cluster. The sound generation module inside the cluster device will create a buzzing sound to alert the driver of the event. Similarly door lock/unlock and other common events can be communicated through a sound generation module. It can also be used to play MP3 audio data with the help of a CPU that converts MP3 into PCM data for the I2S interface.

**Components**

**The Communication Interface**

- CAN Bus is the most common high speed communication network in Automotive ECU to
communicate with body and gateway devices that are sitting far from a cluster/dashboard. 

- LIN provides cost-efficient communication in applications where the bandwidth and the versatility of CAN are not required. Communication with various sensors, actuators or body controllers is done via the LIN network. LIN is used to communicate to comparatively local devices such as communication between the steering wheel audio/music system control device with the cluster, headlight ON indication from switch to the cluster device, a side/parking indicator communication (side indicator/parking light switch to dashboard display and sound device) is handled by LIN. However these things are flexible and depend on end-user preference.

- SPI & IIC are mainly used for inter chip communication in short distances. SPI interface can be used to access slow serial flash, to communicate with power control device sitting near to the cluster, LED drivers etc.

**Graphics Application Component**

The use of LCDs started as a small “add-on-display” along with the electromechanical needles-based gauges to display additional information, but is now a trend in automotive instrument clusters as graphical displays replace traditional mechanical gauges altogether. The challenge is now how to give the user the same feel and realistic experience of the needles and pointers using graphics computation techniques. In addition, information that gets onto displays such as a seat belt indicator, tire pressure level, engine temperature, rear camera view, etc., is treated as safety relevant and considered safety critical. Other miscellaneous information such as navigation, song select, etc. are also part of these displays. The full color TFT LCD panels now allow automotive OEMs to provide state of the art interfaces with the following features:

- Menu-driven configurability
- Animated pointers on gauges
- Text and graphics to a similar standard as desktop or handheld devices

![Figure 3: A typical digital cluster display](image)

All of the above features require a minimum desired display size support, specific frame-rate requirements and the capability of handling complex animations. The following components play an important role in achieving the required functionality:

**Application Processor**

The applications processor is a high-performance core primarily used for managing the graphics
application in high-end cluster solutions. It is used to run the graphics drivers and generate commands for execution by the GPU. It is in general, a pipelined processor having precision Floating Point Unit (FPU) and Instruction/Data Caches for trapless and faster operations. It can also be equipped with an additional Media Processing Engine (MPE) extending the capabilities of the FPU for further acceleration of media and signal processing functions by adding instructions targeted at audio, video, 3-D graphics, image, and speech processing.

In the graphics subsystem, the application running will call different APIs to access the graphics hardware through programming calls to the operating system. When the application requests an image to be rendered on the display, the API calls the OS, which in turn invokes the GPU driver to communicate with GPU hardware to draw the image to be shown on the display. From the perspective of the application processor, it accumulates and sets up graphics commands that are dispatched to the GPU for processing and display rendering. It does this by means of writing the commands to volatile memory (typically DRAM) where the GPU can subsequently fetch the commands. For example in-case of digital display of needles (let’s say speedometer, fuel indicator, etc.), it will get the information from a real-time processor regarding the corresponding activity and instruct the GPU to rotate the needle at a certain angle. It will control all of the APIs for creating the needle rotation graphics. It instructs the GPU to rotate image, coloring the image, zooming the image, etc.

The MPE associated with it can also be used as JPEG decoder (or limited MPEG decoder) to display images through the display controller. For example, in an advanced speech and navigation system, the navigation unit can send images/frames through Ethernet to display to a cluster RGB display unit. Those JPEG images need to be decompressed and decoded by the MPE and then displayed through a display controller onto an LCD/TFT display. Similarly for audio applications, MP3 data can be sent through Ethernet to cluster devices and software can convert the MP3 format to a PCM format with the help of MPE to play it through speakers using I2S protocol to support in-car entertainment.

**Graphics Processing Unit**

A graphics-processing unit (GPU) is a specialized module designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display via a display controller. GPUs are very efficient at manipulating computer graphics and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. They have much deeper pipelines compared to traditional CPUs along with significantly faster and more advanced memory interfaces to shift around a lot more data than CPUs. It helps in offloading the main core by handling geometry and pixel processing, allowing resolution independent HMI.

In mid-end cluster solutions, the high performance GPU can be a 2D vector and raster graphics accelerator designed for hardware acceleration of vector graphics, which can be displayed using a display controller. The GPU can be accessed through its specific drivers. It can also provide scalability with high-quality rendering, including anti-aliasing, to different screen sizes without multiple bitmaps.

A GPU supports graphics application programming interfaces (APIs) that provide user controls for optimizing the acceleration capabilities available with the GPU. It is preferable to support a hardware platform and operating system (OS) independent software APIs standard rather than a
proprietary API standard for enhanced vector graphics. One such standard is OpenVG 1.1 API standard for vector graphics acceleration. OpenVG is a royalty-free, cross-platform API managed by a member-funded consortium. It provides a low-level hardware acceleration interface for vector graphics libraries such as Flash and Scalable Vector Graphics (SVG). It is used for acceleration of high-quality vector graphics for user interfaces and text on small screen devices. **Controllers**

Core API features of OpenVG include Coordinate systems and transformations (image drawing uses a 3x3 perspective transformation matrix); Viewport clipping, scissoring and alpha masking; Paths; Images; Image filters; Paint (gradient and pattern); Blending; Dithering. The VGU Utility Library provides Higher-level geometric primitives, Image warping.

Typical use cases in the instrument cluster are where a GPU will accelerate the needle rotation and display controller renders the rest of the scene, for infotainment where it can be used for User Interface (UI) acceleration, for native rendering of true-type fonts with Anti-Aliasing, additional graphics acceleration for dual display systems, etc.

**Display Controller**

This is the main graphics display controller of the device. It is designed to drive displays using direct blit graphics and video with RGB data. It generates all of the necessary signals required to drive the displays with up to 24-bit RGB data bus, Pixel Clock, Data Enable, Horizontal-Sync, and Vertical-Sync. There can be as many as two independent controllers in a high-end cluster solution for driving two different displays simultaneously. two is not the limit.

The specified internal memory of the device allows the controller to easily handle complex graphics contents (pictures, icons, languages, fonts) with a very low memory footprint. It can also be used for creating simple animation without any help of dedicated graphics processor, for e.g., image sequence animation (to display a rotating car), list scroll, object fade in/out, fly in/out, dynamic reveal/hide, etc.

Graphics are managed through planes blending using the merging of multiple programmable layers (objects) to optimize use of internal memory buffers. Since the CPU need only to manage the configuration of the layers, it is possible to create animations with low CPU overhead and minimal RAM footprint.

There can be a number of extensions to the display controller interfaces and capabilities to support wide range of flat panel or curvature displays available in market based on product requirement. A few of them include:

**RSDS**

It can also interface with on-chip TCON (Timing Controller) to generate the timing signals and to directly drive the row and column drivers of display panels via RSDS (Reduced Swing Differential Signaling) pads. RSDS is having a voltage swing of 200mv and is similar to the Low Voltage Differential Signal (LVDS). RSDS interface signals instead of TTL based signals leads to the reduction of EMI in between the Panel Timing Controller and the Column drivers. The reduced signal swing level of RSDS also results in low power consumption and EMI levels as compared to TTL based interface.
LVDS

It can also support the Open LVDS (Low Voltage Differential Signaling) Display Interface (OpenLDI) specification through an on-chip converter which converts the digital RGB into OpenLDI-compatible format of 4-data-pair and 1 clock pair LVDS interface output. It minimizes the number of wires that must be used to connect the display source and display device while also minimizing radiated emissions and susceptibility to electromagnetic interference. This enables the display controller to support a wide range of display formats, refresh rates, and pixel depths.

HUD

One of the recent and latest features that have come up is of the head-up display (HUD). A head-up display is a display technology projecting the content directly on the windscreen of the car. As the windscreen is not a flat projection area, the image gets distorted. So to compensate this effect, inline, real-time warping is being performed on the content to be displayed using the graphic accelerator or dedicated hardware. The fundamental advantage of such display is that the information is presented in the normal field of view of the driver looking onto the street. It will enhance safety while delivering navigational and other critical information to drivers such as speed, fuel level, etc. Head-up displays are usually used in combination with a TFT display in the instrument cluster.

![Image of an integrated HUD solution]

Video Input Unit

The Digital Video Input module is mainly used to take a compatible video stream or other supported formats as input from an external interface like reverse camera, driver monitor camera, etc. and then enabling a set of processing on the captured digital video to finally get it displayed through the Display Controller to TFT/LCD Display for the user/driver. Different kinds of processing on video include brightness adjustment, contrast adjustment, up-scaling or down-scaling of the picture, input data format conversion from one color space to other, horizontal mirroring for reverse camera adjustment, etc.

In case of analog camera input, the picture information is transmitted to some off-chip Video ADC which converts the signal to Digital and feed to this module. Usually RCA composite cables are used to connect the analog camera to the converter. Conclusion

RLE Decoder
This module is used to decode data that has been compressed using a Run Length Encoding (RLE) scheme. Run length encoding is a simple scheme that compresses data based on repetition of consecutive entries. Each block of data is preceded by a command byte that indicates whether compression is active and if so how many copies of the next data are required.

The RLE decoder performs extraction of run length encoded data and is optimized for extraction/decompression of graphic images and put it back in local memory for Display Controller to display. Data are fed to RLE typically using DMA and decompressed data are fetched typically by DMA. Any portion of a graphic can be decoded based on given start-pixel location and given end-pixel location. Apart from lossless decompression, it also supports the partial image decoding feature.

**Segmented LCD Controller**

This controller is equipped with multiple Front and Back plane drivers with the backplane drivers remapping feature to drive the external segmented LCD to display clock, odometer, trip meter, etc. or other related information. It is configurable to support multiple varieties of segmented displays. This module is also responsible for On-chip generation of all output voltage levels that are required to drive the display segments. It also supports programmable frame clock generator, programmable bias voltage level selector, programmable output current and an optional output current boost during transitions for displays along with the contrast adjustment feature. In a low cost solution, segmented LCD display can be an alternate option to graphical TFT display.

**Conclusion**

In our modern age with a highly featured digital solution, all existing luxury is transforming into necessity and creating scope for new innovation. Hence the immediate challenge is to provide integrated solutions that are cost effective and technically advanced to fulfill market requirements. The automotive cluster has transformed from analog gauges to digital displays to integrated cluster infotainment solutions. The future trend will be to move towards a highly integrated solution where basic cluster, infotainment, advanced driver assistance will all be integrated in a single device. Hence it is very important to understand the basic architecture to know how they can be used to create an integrated cost-effective solution for the future.

*Read Part II, which covers cluster security requirements and application, device memory requirements, and low power mode cluster architecture.*

**References**

1. Premium Line Instrument Cluster
2. Designing a Vehicle Instrument Panel Cluster—A Case Study
3. FPD-Link
4. Low-voltage differential signaling
5. SuperImaging HUD