ISSUE: March 2015

PMBus Gives Designers New Options For Meeting Adaptive Voltage Scaling Requirements

by Peter James Miller, Texas Instruments, Manchester, N.H.

While adaptive voltage scaling (AVS) has been around for more than two decades, it has severely limited a designer's choice of power solutions. Today, an emerging, open-source serial communications protocol known as PMBus is opening new possibilities for flexibility in power design. In this article, we'll look at how PMBus can enable digital system and power designers to realize their AVS power needs.

What Is AVS?

Adaptive voltage scaling is a method by which a powered device is able to communicate with its power supply to control or "adapt" the voltage provided by the power supply to the needs of the powered device. As processor designers search for new ways to deliver value to their customers, an increasing variety of devices are offering AVS capabilities, demanding high-performance AVS power solutions.

AVS was once the domain of high-end CPU core processors that needed to reduce power dissipation to avoid overheating. Today, however, AVS is pervasive through most digital processors and their support chipsets. Adapting their supply voltage allows these devices to improve performance or reduce power consumption. However, most AVS solutions require specific power supply controllers designed to support their specific form of AVS communication.

Since the processor and power solution must offer compatible AVS communication, frequently designers are restricted in their choice of power solutions. These solutions must be specially designed to work with their processor's form of AVS, or use a custom bridge device to interface between the power supply and processor. This often means lower volume and higher-priced power solutions dedicated to powering specific processors.

However, now system designers are presented with a new choice. PMBus-enabled power supplies provide a standard interface that allows their output voltage to be digitally controlled. PMBus power supplies allow system designers to implement their own AVS bridge. In some cases, designers could use direct digital communication between their processor and its power supply to meet their AVS needs.

What Is Power Management Bus?

A power management bus, or PMBus, is an open-standard, multi-point-compatible, serial communication protocol that uses defined-register addresses and formats to simplify and standardize communication between power management devices. Built on the foundation of system management bus (SMBus) using a physical and network layer compatible with the I^2C two-wire serial communication standard, most digital processors and controllers can serve as a PMBus host and control a PMBus power solution using I^2C-based drivers in a simple point-to-point configuration.

The register address and data formats of common power management-related information and functions used in PMBus are defined by the PMBus standard.[1] These defined-register addresses are commonly called "commands" within PMBus with their "command codes" being the equivalent of their I^2C register address.

The basics of implementing AVS through PMBus require understanding the basic transaction structure used by PMBus and a few standard PMBus commands. However, once a digital designer takes the step of implementing PMBus, they can open a wealth of options provided by the digital communication between a digital processor and its power supply. This enables configuration, control and monitoring of the power solution.

Like I^2C, PMBus is a variable-length packet of 8-bit data bytes, each with their own receiver-acknowledge, wrapped between a start and stop bit. The first byte is always a 7-bit "slave address" followed by a 0 "write bit." Sometimes this is called the "even address" that identifies the intended receiver of the packet. The second byte is an 8-bit "command" byte that identifies the PMBus command being transmitted using that command's command code. After the command byte, the transmitter either sends data associated with the command to write to the receiver's command register, from lowest to highest byte, or sends a new start bit. This indicates

the desire to read the data associated with the command register from the receiver, after which the receiver transmits the data following the same lowest-byte first format.

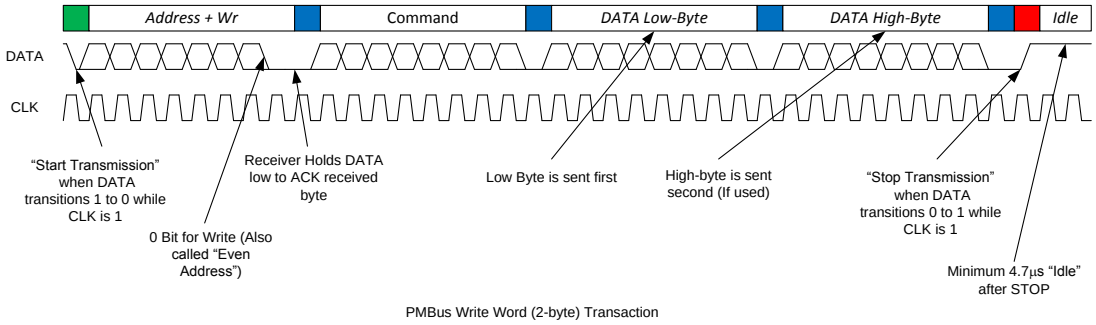

Fig. 1. PMBus timing diagram of a write word transaction shows the start and stop bits, packet contents, data byte order, and receiver acknowledgements.

Implementing AVS With PMBus

The PMBus protocol contains two standard methods for implementing the output voltage control required for AVS. The first uses the basic 100-kHz to 1.0-MHz I^2C-derived physical and transport layers with command code 21h (VOUT_COMMAND) to actively set the output voltage. The second uses a 5- to 50-MHz serial interface introduced as an option in PMBus Revision 1.3 Part III, called AVSBus. AVSBus provides a royalty-free, open standard, serial communications protocol at up to 50 MHz for AVS solutions requiring lower latency than can be achieved through the 100-kHz to 1-MHz standard PMBus protocol. While AVSBus will be covered in future articles, this article focuses on using the base PMBus protocol, and specifically the standard VOUT_COMMAND implementation.

AVS solutions using the basic PMBus protocol typically use three PMBus commands:

1. VOUT_MODE (20h), which establishes the format used for output voltage related commands;

2. VOUT_TRANSITION_RATE (27h), which sets the slew rate of the power supply's output when changing its output voltage; and

3. VOUT_COMMAND (21h), which provides a 16-bit word setting the target output voltage based on the format established in VOUT_MODE.

VOUT_MODE

PMBus supports four output-voltage formats: linear, VID, direct and IEEE half-precision floating point, plus a 5-bit parameter. In linear format, the parameter sets the exponent used to scale the 16-bit mantissa in VOUT_COMMAND. In VID format, the parameter selects the VID encoding used by the 16-bit VOUT_COMMAND. Not all of the 32 available VID codes are used, and several "future use" and "manufacturer-specific" codes are already reserved.

Direct format equally divides the 16 VOUT_COMMAND bits between the minimum and maximum allowed voltages to maximize the resolution within the range, but requires reading the COEFFICIENTS (30h) command. IEEE half-precision floating point splits the 16-bit VOUT_COMMAND into a sign-bit, 5-bit exponent and 10-bit mantissa, which contains all of the VOUT related information, but reduces the range and resolution to only 10-bits. The most commonly used VOUT_MODE format for implementing AVS is linear mode. This allows the designer to use all 16-bits of VOUT_COMMAND to specify the desired output voltage, providing an exceptional combination of range and resolution.

VOUT_TRANSITION_RATE

VOUT_TRANSITION_RATE sets the output-voltage slew rate used when the output voltage is changed though a PMBus command such as VOUT_COMMAND, in mV/µs. Before implementing an AVS solution using PMBus, a designer should know their processor's slew-rate requirements and their power supply's slew-rate capabilities, then set the VOUT_TRANSITION_RATE of the power supply as needed.

VOUT_COMMAND

VOUT_COMMAND sets the output voltage based on the format defined in VOUT_MODE. This 16-bit word is the heart of implementing AVS via PMBus. VOUT_COMMAND sets the absolute output-voltage level, in volts, as measured by the power supply. This could be translated from a parallel VID code, or using a direct or linear format as set by VOUT_MODE.

When implementing open-loop AVS solutions that command a target output voltage, VOUT_COMMAND can be directly written. However, when implementing closed-loop AVS solutions that monitor and adjust the power supply voltage based on the current power supply voltage's performance, be sure to read the current VOUT_COMMAND value before making adjustments to ensure the designer makes only the intended step in output voltage.

AVS process with PMBus VOUT_COMMAND

In general, the AVS process follows these command steps:

1. Read VOUT_MODE to confirm the data format used by the power supply.
    a. Write VOUT_MODE, if applicable.
2. Read VOUT_TRANSITION_RATE to check the slew rate the power supply will support.
    a. Write VOUT_TRANSITION_RATE, if applicable.
3. Read VOUT_COMMAND to confirm that the current output-voltage programming matches the expected VOUT_COMMAND, especially in closed-loop dynamic AVS solutions.
4. Write VOUT_COMMAND to change the voltage generated by the power supply.

For example, when using a power supply at PMBus address of 14h or 36d (001 0100b) that supports the linear data format with exponent –9, set the output voltage equal to 1.100 V as follows.

1. Read VOUT_MODE which contains the bytes:
    a. start-bit, signifies the start of a new transaction packet
    b. 28h (slave address 14h + 0 write bit)
    c. 20h (VOUT_MODE command code)
    d. second start-bit, signifies this is a "READ" transaction packet
    e. 29h (slave address 14h + 1 read bit)
    f. slave writes back 17h (000 10111b), indicates absolute linear-mode (000) and e**x**ponent = –9 (10111)
    g. stop-bit, signifies the end of the transaction

2. Write VOUT_COMMAND = 1.100 V, which would contain the bytes:
    a. start-bit, signifies the start of a new transaction packet

b. 28h (slave's address + 0 write bit)

c. 21h (VOUT_COMMAND command code)

d. 33h (the low-byte of the 563d mantissa for 1.100 V with exponent –9)

e. 02h (the high-byte of the 563d mantissa for 1.100 V with exponent –9)

f. stop-bit, signifies the end of the transaction

After that command is received and acknowledged, the output voltage transitions to 1.0996 V at the slew rate defined by VOUT_TRANSITION_RATE. Once VOUT_MODE, VOUT_TRANSITION_RATE and the current VOUT_COMMAND are known, further AVS routines can be run with just the VOUT_COMMAND transmission.

AVS With Manufacturer-Specific Output Control Commands

Since the PMBus protocol standard does not require PMBus-compatible power solutions to implement all PMBus commands, or even every part of PMBus commands they support, it may be desirable to implement AVS using manufacturer-specific PMBus commands.

For example, the TPS544C20[2] does not implement VOUT_COMMAND, but allows relative adjustment of the output voltage by using the MFR_SPECIFIC_04 (command code D4h), called VREF_TRIM. This two-byte word provides the user with the ability to adjust the output voltage from +10% to –20% of the initial output voltage using a least significant bit (LSB) of 0.33%. Translating an AVS voltage over this range can require some initial design effort, but allows a designer to implement an AVS solution when VOUT_COMMAND is not available.

For example, using a 10-kΩ and 15-kΩ feedback divider with the TPS544C20's fixed 0.6-V reference, the initial voltage is programmed to 1.0 V (Fig. 2.) A programmable output-voltage range of 0.8 V to 1.1 V with a 3.3-mV resolution is set by a 16-bit signed register from –3Ch to +1Eh (–60d to +30d).

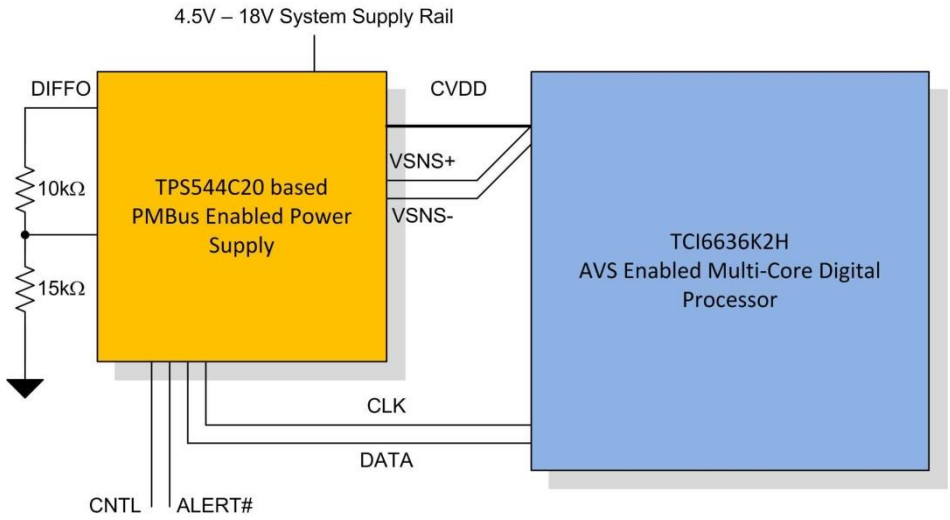

Fig. 2. Block diagram and PMBus connections for an AVS via PMBus power solution for a TCI6636K2H Multicore DSP+ARM KeyStone II System-on-Chip using a TPS544C20-integrated FET power converter.

The write process here is similar to writing VOUT_COMMAND, except the meaning of the bits has changed slightly. Instead of representing the absolute voltage as they did in VOUT_COMMAND, they now represent a signed voltage change from the initial boot voltage of 1.0 V. While this can be more complicated than using VOUT_COMMAND, it is generally easier to establish a first time power-up voltage without communicating with

the power supply via the PMBus. This is a valuable feature when the PMBus communication is controlled by the powered device.

Since manufacturer-specific commands like VREF_TRIM can vary from manufacturer to manufacturer or even device to device, designers should read a device-identifying register such as DEVICE_ID (command code ADh) or DEVICE_CODE (command code FCh using the TPS544C20) to confirm the power supply is using the correct device prior to issuing a MFR_SPECIFIC command.

For example, using a TPS544C20 at PMBus address 14h or 36d (001 0100b) with a 10-kΩ and 15-kΩ feedback divider, a user wants to set the output voltage equal to 1.050 V.

1. Read DEVICE_CODE which contains the bytes:
    a. start-bit, signifies the start of a new transaction packet
    b. 28h (slave address 14h + 0 write bit)
    c. FCh (DEVICE_CODE command code)
    d. second start-bit, signifies this is a "READ" transaction packet
    e. 29h (slave address 14h + 1 read bit)
    f. TPS544C20 would write back 53h, the low-byte of its device code
    g. followed by 01, the high-byte of its device code
    h. stop-bit, signifies the end of the transaction

2. Write VREF_TRIM = +5% (= 50 mV for VOUT = 1.05 V), which contains the bytes
    a. start-bit, signifies the start of a new transaction packet
    b. 28h (slave's address + 0 write bit)
    c. D4h (MFR_SPECIFIC_04 command code for VREF_TRIM on the TPS544C20)
    d. 0Fh (low-byte of the 15d mantissa 0 for +5% w/0.33% LSB)
    e. 00h (high-byte of the 15d mantissa for +5% w/0.33% LSB)
    f. stop-bit, signifies the end of the transaction

This process can then be repeated to continually adjust the output voltage as needed by the processor. Since the VREF_TRIM = 0, output voltage is programmed to 1.00 V, and the LSB is 3.33 mV. The process of calculating the required VREF_TRIM value is:

$$\text{VREF\_TRIM} = (\text{VOUT} - 1.000\ \text{V}) / 3.33\ \text{mV} \qquad (1)$$

The combination of standard PMBus commands, like VOUT_COMMAND and manufacturer-specific commands like VREF_TRIM, provide system designers with flexibility to adapt their power solutions to their specific needs and system requirements.

Now that we've established the basics of how to implement an AVS solution using the PMBus communications protocol, both with the standard VOUT_MODE and VOUT_COMMAND commands and using manufacturer-specific

implementations like VREF_TRIM in the TPS544C20, we'll build on this knowledge in the coming months. In another article, we'll look at sharing the PMBus among multiple processors and power supplies. We will use a PMBus host as an AVS bridge and a common interrupt so that multiple processors can share a common PMBus channel so each can talk to their own power supply.

References

1. Information about PMBus
2. TPS544C20 and TCI6636K2H datasheets.

About The Author

Since receiving his Master of Science degree in electrical engineering from Worcester Polytechnic Institute (WPI), Massachusetts, Peter James Miller has worked in TI's Power business group as an applications engineer, IC designer and systems engineer supporting nonisolated dc-dc controllers and converters. A member of TI's Technical Staff, Peter currently works in TI's High-Performance DC Solutions integrated FET team where he assists leading customers as they develop power solutions around TI's PMBus-enabled integrated FET converters. Peter is also involved with defining TI's next generation of PMBus-enabled power converters. Peter can be reached at ti_petermiller@list.ti.com.

For further reading on board-level power management, see the How2Power Design Guide's Advanced Search page, go to Search by Design Guide Category and select "Board-level Power Management" in the Design Area category.