PMBus Offers Open-Standard Digital Power Management

By Bob White, Staff Engineer, Artesyn Technologies, Westminster, Colo.

This highly flexible, power-management protocol defines the transport and physical interface, as well as the command language required for communicating with power converters.

The need for digital power management has become more acute in recent years, due to several related factors. Many board designers have moved to intermediate bus power architectures, using multiple on-board dc-dc converters to generate the diversity of power rails needed by different silicon devices. One obvious consequence is that the task of configuring, controlling and monitoring these power sources—during design, production test and everyday in-system use—is now significantly more complicated. Simply controlling power up/down sequencing can demand dedicated programmable ICs and large numbers of additional components, to say nothing of the configuration or real-time feedback facilities needed for flexible system-level control and diagnostics.

Most modern dc-dc converters are still configured and controlled via analog signals derived from simple passive components. Even sophisticated high-functionality converters with state-of-the-art power-conversion topologies are likely to use external trim resistors and capacitors for defining values such as startup time, setpoint value and switching frequency. And of course, none of these parameters can easily be changed on the fly, making it virtually impossible to implement adaptive—let alone predictive—power-management schemes.

With the exception of a few specialist converters for microprocessors (which offer limited digital programmability in the form of voltage identification [VID] codes for output voltage control), most brick, intermediate bus and point-of-load (POL) converters on the market still operate in the analog control domain. The most urgent need is for digitally controlled nonisolated POL converters, because these are used extensively to provide the final voltages for devices on a board. However, the requirement also already embraces isolated converters, and designers will no doubt shortly be adding other digitally programmable power sources to their wish lists.

The reason for this seemingly odd scenario is simple: until now, there has been no industrywide consensus on digital power management. A number of power supply manufacturers have launched digitally programmable POL converters, which goes some way toward addressing the issue, but these are based on proprietary architectures and silicon.

What Exactly Is PMBus?

PMBus is an open-standard digital power-management protocol with a fully defined command language and transpost and physical interface. It facilitates communication with a power converter or other device. The protocol was founded by a coalition of power supply and semiconductor manufacturers that recognized that lack of a suitable standard was inhibiting the adoption of an all-digital power-management solution, and it is now rapidly gaining industry acceptance. In March 2005, Revision 1.0 of the protocol was placed in the public domain, and ownership

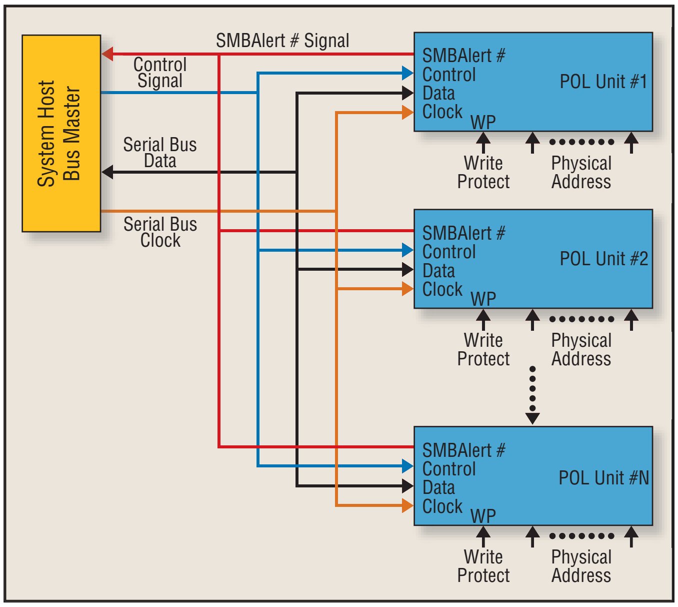

Fig. 1. *The PMBus protocol specifies clock, data and interrupt lines as in the SMBus standard, but adds two hardwired signals—a control signal to turn individual slave devices on and off, and an optional write-protect signal that prevents changes to memory-held data.*

transferred to an independent special interest group (SIG) known as the System Management Interface Forum, which is now responsible for further developing and promoting the standard.

It is worth noting that PMBus is not a standard for ac-dc power supplies or dc-dc converters. It does not specify attributes such as form factor or pin-out, which are better served by industry alliances such as POLA and DOSA; nor does it address communication between one power source and another—this remains the responsibility of semiconductor and power supply manufacturers.

Low-Cost Real-Time Control

The PMBus transport layer is based on Version 1.1 of the low-cost System Management Bus (SMBus), which is a more robust version of the industry-standard I^{2}C serial bus with packet error checking and host notification features. The I^{2}C bus was originally developed by Philips Electronics (New York) for Inter-IC communication, while the SMBus was defined by Intel (Santa Clara, Calif.) for system management communications in PCs and servers.

SMBus features a third signal line—SMBALERT—which allows slave devices such as POL converters to interrupt the system host/bus master. This arrangement is inherently more flexible than a system employing the master to constantly poll slave devices, and it imposes far less of a burden on the host processor, making it easier for designers to implement event-driven closed-loop control schemes. Furthermore, the PMBus protocol dictates that all slave devices must either store their default configuration data in nonvolatile memory or use pin programming, so that they power up without any bus communication. System startup times are consequently significantly shorter than with other digital control solutions on the market, which demand that the bus master configures all slave devices as part of the power-up initialization routine.

The physical address of each slave device is defined via dedicated pins. Silicon manufacturers are certain to offer a variety of innovative approaches, such as tri-state pins and resistor value programming. In addition to the SMBus' clock, data and interrupt lines, the PMBus protocol also specifies two hardwired signals for use with power-conversion devices: one is a control signal used in conjunction with commands received over the bus to turn individual slave devices on and off; the other is an optional write-protect signal that can be used to prevent any changes to memory-held data. The control signal is often driven by a power system controller. If not, the pin can be hardwired low or high as needed.

A typical PMBus implementation is shown in **Fig. 1**. SMBus uses the wired-AND connection of all devices on the bus to provide arbitration in the event of bus contention, and is electrically similar to the I^{2}C bus.

PMBus has the distinct advantage of the master device not being based on proprietary silicon, and it does not act as a translator. All communications between the host and the power sources are conducted entirely via the bus. This saves on implementation costs and provides a much more flexible control approach. The host can be the system's existing processor, a low-cost, general-purpose microcontroller, or even some gates in an FPGA. Of course, the host also can be different things at different stages of product development.

For example, during the board design phase, a laptop PC can be used as the host; then during production testing, this role can be assumed by automatic test equipment, to comprehensively verify board performance, and if necessary, dynamically change the operating parameters of individual power-conversion devices to accommodate the needs of the silicon on that board. The final select-on-test values can then be stored in the slaves' nonvolatile memory.

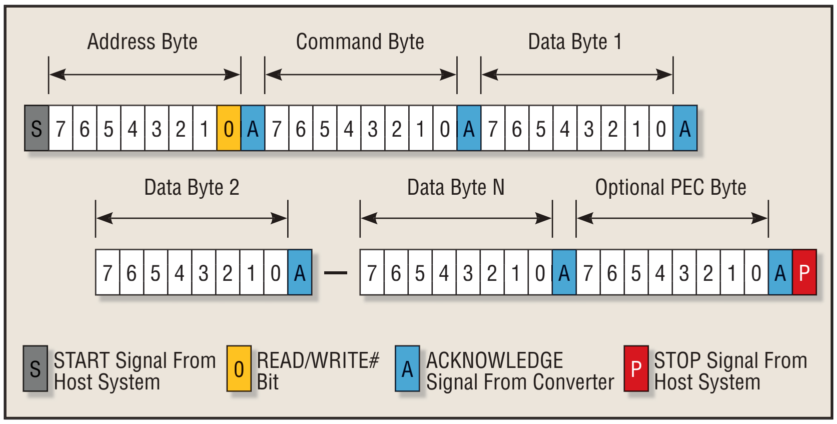
Address Byte
Command Byte
Data Byte 1
S 7 6 5 4 3 2 1 0 A 7 6 5 4 3 2 1 0 A 7 6 5 4 3 2 1 0 A
Data Byte 2
Data Byte N
Optional PEC Byte
7 6 5 4 3 2 1 0 A — 7 6 5 4 3 2 1 0 A 7 6 5 4 3 2 1 0 A P
S START Signal From Host System
0 READ/WRITE# Bit
A ACKNOWLEDGE Signal From Converter
P STOP Signal From Host System

Fig. 2. *In a standard master-to-slave PMBus communication sequence, the slave processes and executes a command immediately after it receives the "stop" bit.*

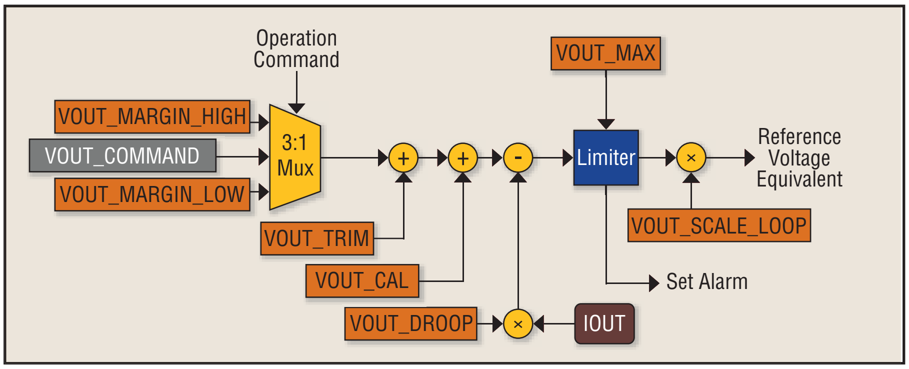

Fig. 3. *A series of PMBus commands are required to set the output voltage on a POL converter.*

Simple Command Language

PMBus communications are based on a simple command set. Every packet contains an address byte; followed by a command byte; zero, one or more data bytes; and an optional packet error code (PEC) byte. **Fig. 2** shows a typical host-to-slave information transfer; the master uses single "start" and "stop" conditions to indicate the beginning and end of the process, and the addressed slave device uses a single bit to acknowledge reception of each byte. To minimize response times and processor overheads, the slave processes and executes a command immediately after it receives the "stop" bit; unlike many other bus protocols, it is not forced to wait for a separate "execute" command.

While the protocol's one-byte command code implies that as many as 256 commands are potentially available, it is not a condition of compliance that PMBus devices have to support all commands, and in fact, many will use only a small subset to achieve their intended purpose. Considerable attention has been paid to "future proofing" the standard; there is provision for two command extensions that effectively permit two-byte commands. One extension is reserved for PMBus device manufacturers' own use, while the other is reserved for subsequent revisions of the protocol.

The PMBus provides a great deal of flexibility. It is not possible in this short article to list the more than 100 commands available to power converter and power system developers. There are several commands for setting the output voltage, as well as commands for setting warning and fault thresholds for input and output voltages, input and output currents, temperature and other parameters. In addition, how the unit responds to each fault, such as immediate shutdown or hiccup mode, can be programmed.

There are commands for retrieving status bits and reading data like output voltage or the unit's temperature. There also are commands for setting up the interleaving of paralleled converters, providing a software lock against accidental changes of data, configuring how the unit responds to on/off commands received from the SMBus port and CONTROL pin, and turn-on and turn-off sequencing. Commands are also available for storing and retrieving inventory information like a manufacturer's ID, model number, serial number and date code.

Though the focus of the following examples is on POL converters, it is important to note that the PMBus can be used with any type of power supply or dc-dc converter. The wide range of commands allow the configuration, control and monitoring of general purpose point-of-load converters, microprocessor-powering VRMs, standard isolated dc-dc converters, bus converters and ac-dc power supplies. The PMBus even has the flexibility and capability for use with rectifiers in telecommunications battery plants.

To illustrate the flexibility of the PMBus specification, let us take a look at the commands available to control the output voltage, which include:

- VOUT_COMMAND
- VOUT_MARGIN_HIGH
- VOUT_MARGIN_LOW
- VOUT_TRIM
- VOUT_CAL
- VOUT_DROOP (as a function of IOUT)
- VOUT_MAX
- VOUT_SCALE_LOOP.

Fig. 3 shows a conceptual view of how these commands are used; the actual implementation is left to the manufacturer of the POL converter, but the overall behavior of the device must be as shown.

The process of setting the output voltage starts with three basic commands: VOUT_COMMAND, VOUT_MARGIN_HIGH and VOUT_MARGIN_LOW. Each of these commands sends a value that is stored in a register in the PMBus device's memory. One of these three values is selected by the OPERATION command and passed on to the rest of the output voltage command processing. The next step is to add the value in the VOUT_TRIM register to the output of the conceptual multiplexer.

The value in the VOUT_TRIM register is a two's complement number that can either add to or subtract from the value from the conceptual multiplexer. The VOUT_TRIM register will typically be used by the end user to adjust the output voltage once the POL converter is assembled into the end user's system. This might be done, for example, to adjust the voltage at the pins of a critical IC to optimize its performance.

Next, the value from the VOUT_CAL register is added. This is also a two's complement number and can add to or subtract from the voltage command value. The VOUT_CAL register will typically be used by the POL converter manufacturer to adjust the output voltage in the factory.

If the POL converter has an output voltage droop characteristic, it is applied now. The VOUT_DROOP coefficients are always greater than or equal to zero, and droop is only applied if the output current is greater than zero. The value of the VOUT_DROOP coefficient and the value of output current are multiplied and the result is always subtracted from the voltage command. VOUT_DROOP can

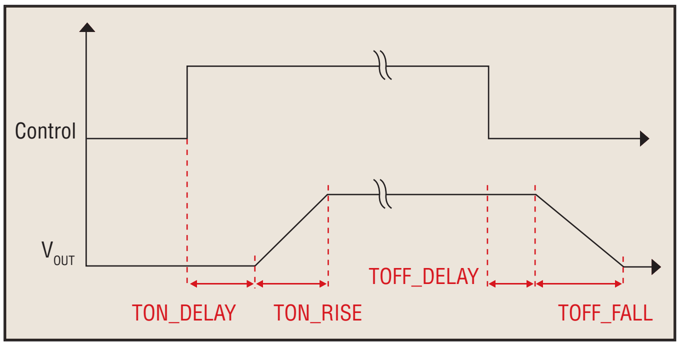

Fig. 4. *Power-up/-down sequencing typically involves just four PMBus commands.*

only act to reduce the output voltage, never to increase it.

The next step is to compare the commanded voltage developed so far with the maximum permissible output voltage set by the VOUT_MAX command. If the calculated voltage command would create an output voltage greater than the VOUT_MAX value, the POL converter limits the command voltage passed to the controller to the VOUT_MAX value, and also sets an alarm.

The same scaling factor that is applied to the external output voltage by a resistive divider is now applied to the calculated voltage command. This is done by multiplying the calculated voltage command by VOUT_SCALE_LOOP. At this point, the converter has a calculated value that is used as the equivalent to the reference voltage in a standard analog controller. This is the value to which the signal from the resistive divider on the output is compared for adjusting the duty cycle of the POL.

For setting the output voltage, and related commands like setting the output overvoltage fault threshold, the PMBus allows up to 16 bits of resolution. For other data, like reading back the input voltage or the output current, the PMBus specification allows up to 10 bits of resolution. 10 bits of resolution corresponds to a resolution of approximately ±0.05%, which is more than sufficient for the vast majority of the market.

Ease of Implementation

The rich command set of the PMBus protocol enables designers to write lean and efficient power-management programs, and implement the scheme easily and quickly. Voltage sequencing of POL converters provides an ideal example. Until now, many designers have chosen to use some of the excellent special-purpose controller ICs that are on the market to handle this task, accepting the fact that this involves developing programs using software provided by the IC manufacturer and using up valuable board space for the devices. Converters that can be directly controlled by PMBus potentially offer a more cost-effective and flexible solution, enabling a wide range of operating parameters to be changed at any point during the product's life cycle to accommodate engineering changes.

Only two PMBus commands are required for controlling the startup sequence of a POL converter, as shown in **Fig. 4**. TON_DELAY programs a time for the converter to wait after receiving the control signal before starting to produce an output, and TON_RISE programs the time for the output to increase from zero to the final programmed value. The user simply programs each converter with its own turn-on delay time and turn-on rise time. Similarly, only two commands (TOFF_DELAY and TOFF_FALL) are required for turn-off sequencing.

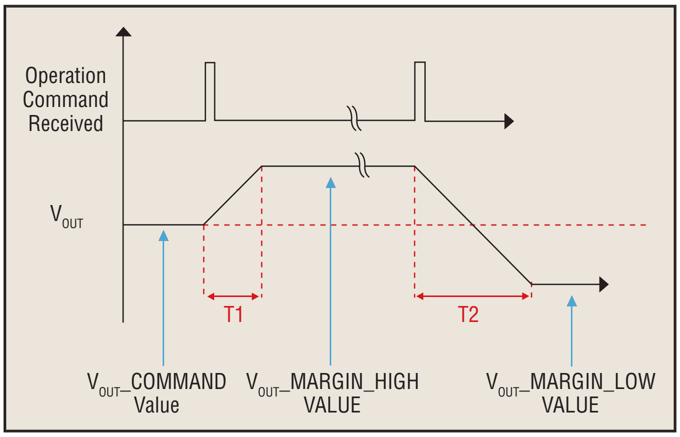

Fig. 5. *Voltage margining is controlled with one PMBus command.*

Voltage margining is another area where digitally programmable converters will make life easier for designers and production test personnel. Many board manufacturers now use this technique to evaluate the performance of ICs in the face of minor variations in their supply voltages; any marginal or below-spec devices can then be replaced as part of the normal production test process, before they become expensive, difficult-to-rectify field failures.

Until now, margin testing has been a highly iterative and time-consuming procedure, involving fitting different value resistors to dc-dc converters in order to vary their output voltage a few percent either side of nominal. PMBus-compliant POL converters simplify this process using just two commands: VOUT_MARGIN_HIGH and VOUT_MARGIN_LOW. As shown in **Fig. 5**, each converter can be instructed to deliver tightly controlled test voltages, while the effect on board performance is monitored. This can significantly reduce production test times, help eliminate ambiguity and produce clearly documented test results.

The T1 and T2 time periods are determined by the VOUT_TRANSITION_RATE command, and the rate is defined in mV/µs. Hence:

$$T1 = \frac{(\text{VOUT\_MARGIN\_HIGH - VOUT})}{\text{VOUT\_TRANSITION\_RATE}}$$

$$T2 = \frac{(\text{VOUT - VOUT\_MARGIN\_LOW})}{\text{VOUT\_TRANSITION\_RATE}}$$

Setting the VOUT_TRANSITION_RATE to a value of FF FFh implies transition as quickly as possible.

Further information about the open-standard PMBus digital power-management protocol and the SMBus transport layer can be obtained from the System Management Interface Forum's website at www.powerSIG.com. **PETech**