Using PMBus™ for Improved System-Level Power Management

Kurt Hesse

ABSTRACT

This topic provides a brief high-level introduction to the PMBus Standard for controlling a power supply using an enhanced serial interface, and then describes the more common and basic PMBus commands. Several system level tasks are presented with possible ways to implement them using the facilities that may be available to the designer using a PMBus-enabled converter or controller with the salient features of the PMBus highlighted. Finally, an example specification and the PMBus commands required for its implementation are illustrated with an application incorporating a suitable controller.

I. INTRODUCTION

Power-supply technology keeps evolving, albeit at a slower general pace than some of the more dynamic fields like processor development. Over the last few years, requirements in some applications for power-supply size, integration, functionality, flexibility, monitoring, and control have brought about the need for a system to go beyond the traditional power supply of the past. One of the results of this has been the introduction of the Power Management Bus, or PMBus™.

What is this PMBus? Simply put, it is a specification to allow digital control of a power supply over a specified physical bus, communications protocol, and command language (see Reference [1]). A conceptual diagram of several PMBus-capable power supplies controlled from a central location is shown in Fig. 1. Figs. 2 and 3 respectively show the block diagrams of typical nonisolated and isolated converters that might be found in the system shown in Fig. 1.
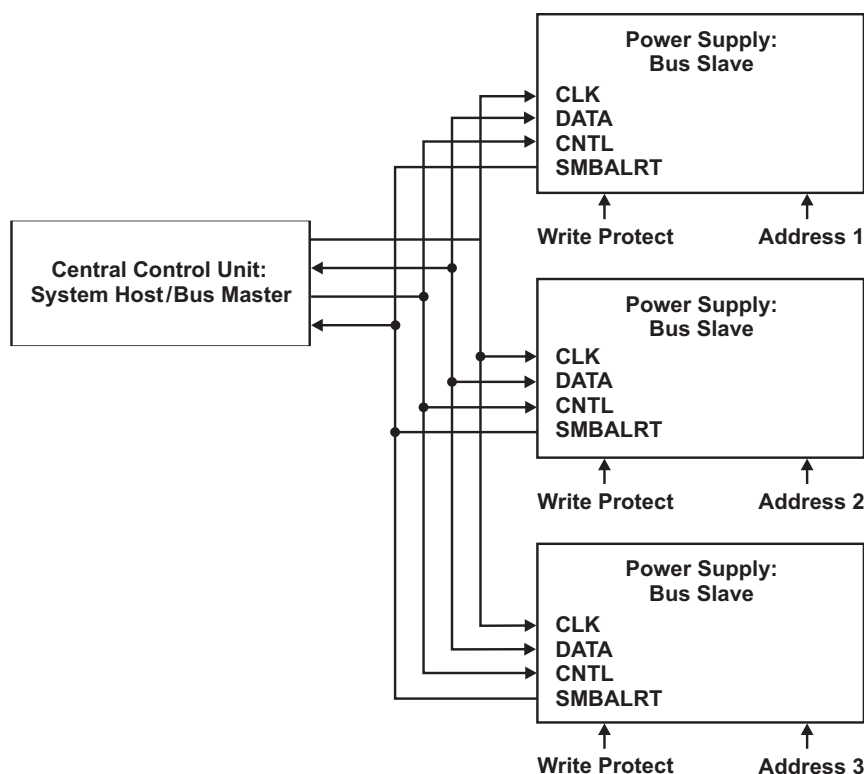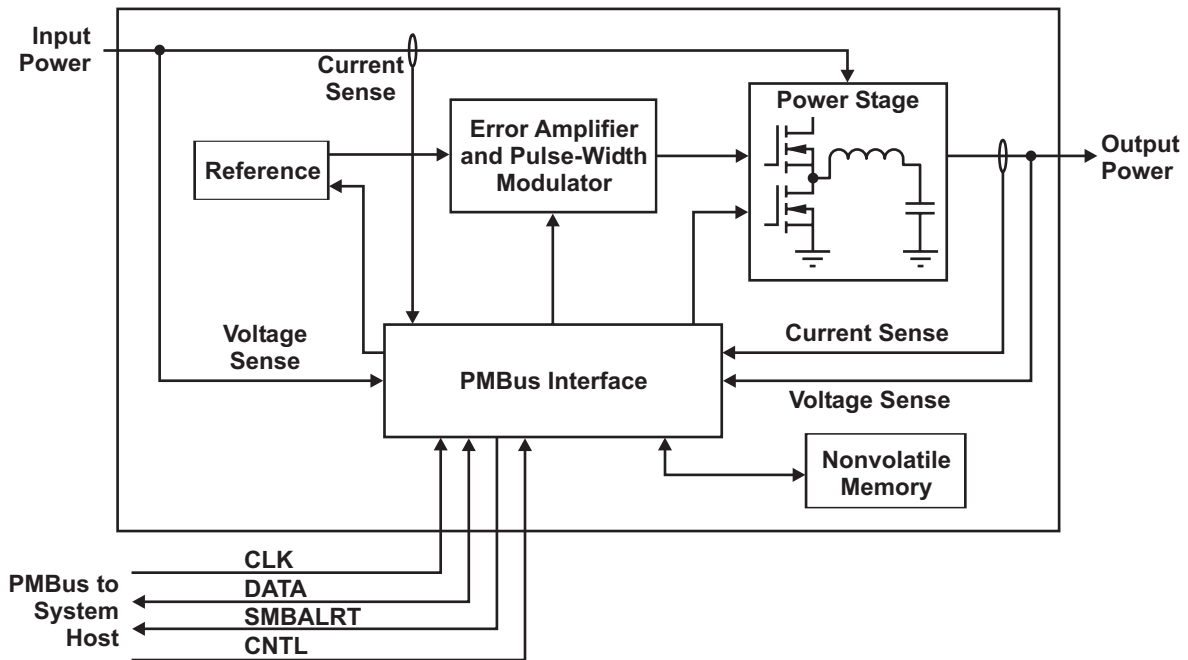

Fig. 1. PMBus conceptual diagram.

Topic 6

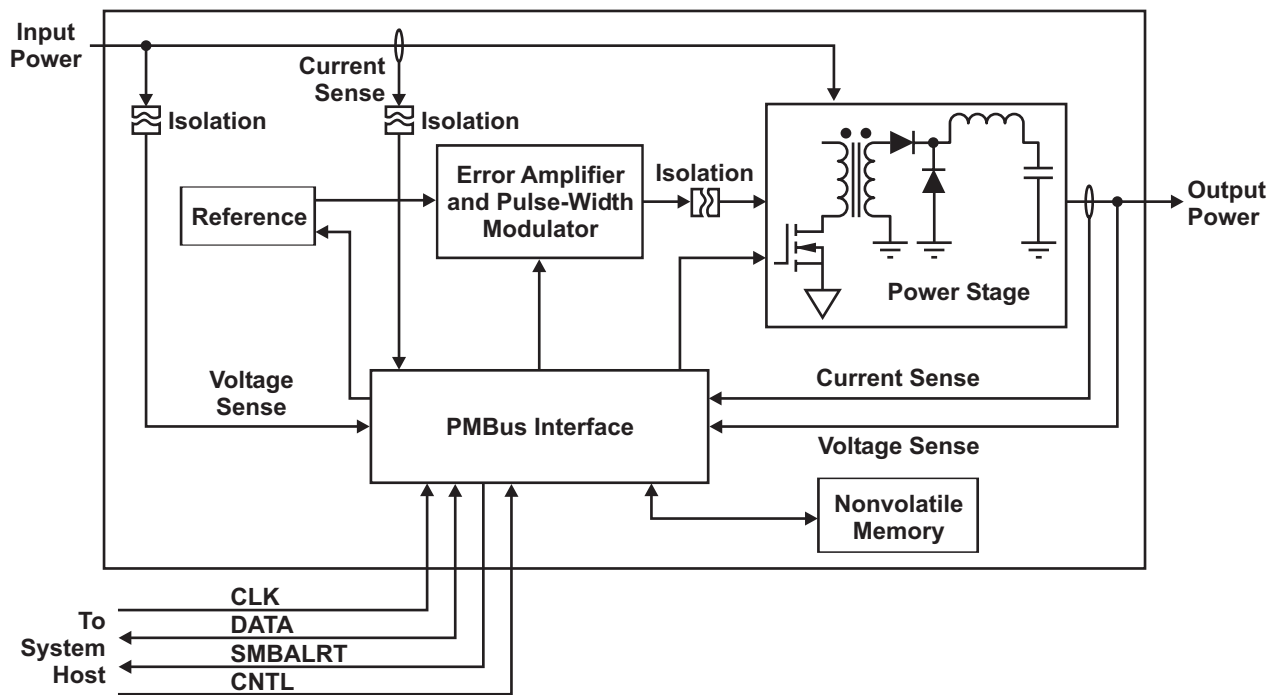*Fig. 2. Block diagram of typical nonisolated PMBus converter.*

Fig. 3. Block diagram of typical isolated PMBus converter.

The typical system employing a PMBus will have a central control unit and at least one PMBus-enabled power supply attached to it. The connected power supplies are always slaves, and the central control unit is always the master. The central control unit initiates all communication on the bus, and the slave power supplies respond to the master when they are addressed.

The PMBus specification governs the way in which the central control unit and the slave power supplies communicate with one another—and that is all. The PMBus specification does not put constraints on power-supply architecture, form factor, pinout, power input, power output, or any other characteristics of the supply. The specification is also divided into two parts. Part I deals with

physical implementation and electrical specifications, and Part II deals with the protocol, communication, and command language.

For a power supply to be PMBus-compliant, it must do several things:

- Meet all requirements of Part I of the PMBus specification.
- Implement at least one of the PMBus commands that is not a manufacturer-specific command.
- If a PMBus command is supported, execute that command as specified in Part II of the PMBus specification.
- If a PMBus command is not supported, respond as described in the "Fault Management and Reporting" section of Part II of the PMBus specification.

Additionally, the device must be capable of starting up unassisted and without any communication with or connection to the PMBus. This behavior may be overridden by programming new defaults for the device, but the capability to start up unassisted must be present. This implies that the PMBus device must be able to store operating defaults for its configurable parameters on the device itself in some form of nonvolatile storage. Doing so can significantly decrease the amount of time required for the system to start up, since no communication is required to configure the device for its operating parameters. If the central control unit gets power from a PMBus device that it is controlling, then that PMBus device must obviously be set to start up automatically, or the central control unit would never start and the system would not function.

II. IMPLEMENTATION SPECIFICS

To get the latest and most complete specification for PMBus, download it from the PMBus Web site at http://pmbus.org/specs.html. The PMBus is derived from the System Management Bus (SMBus) Specification Version 1.1 which is an improvement over the I^{2}C bus. I^{2}C is a simple two-line, synchronous serial communication bus originally designed to allow communication between two or more integrated circuits that are in close proximity to each other [2]. SMBus extensions and improvements over I^{2}C include host notification via the SMBALRT bus line and packet error checking (PEC) to help prevent erroneous operation from noise issues.

There are several differences between the PMBus and SMBus specifications. Those most notable from a system-design perspective are the optional host notify protocol and the group command protocol.

Host notification is required for SMBus compliance, but is optional for PMBus compliance. However, most PMBus devices will support this feature, since it tells the host that a problem exists so it can take appropriate action without having to continually poll each slave device to check for problems. This lightens the load on both the host and the bus itself, providing greater system capability. Host notification is done using a single line (SMBALRT) that is passively pulled high. When a slave has information that the host is likely to need, the slave pulls the SMBALRT line low. The host can then poll each device individually or use the protocol described in the "SMBus Host Notify Protocol" section of the SMBus 2.0 specification (see Reference [3]).

The group command protocol is designed to allow several PMBus-compliant devices to simultaneously execute commands. For more specific information on this feature, refer to Part I, section 5.2 of the PMBus Specification, (see Reference [1]).

III. Types of PMBus Commands

For some power-supply designers, the thought of controlling the supply using a processor may be a foreign concept and may raise questions on the mechanics of doing so. The flow is simple and straightforward. For example, Fig. 4 shows a sequence of commands in pseudocode that would cause a specific converter (converter A) to change its overcurrent limit, undervoltage-lockout (UVLO) value, and start-up time and to store these new values as the default operating parameters. The actual syntax will, of course, change depending on the programming environment and libraries available.
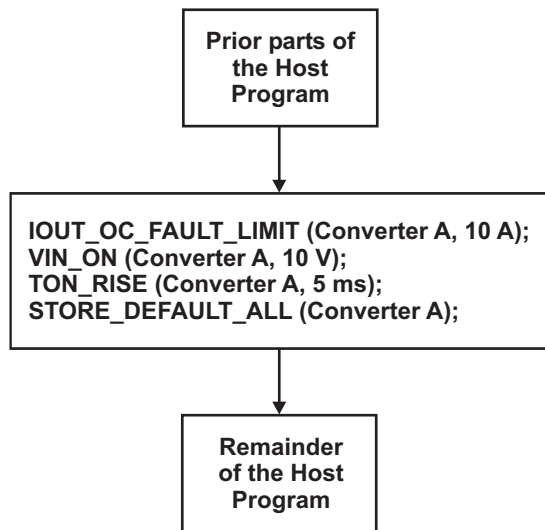
```
┌─────────────────────┐
│   Prior parts of    │
│     the Host        │
│     Program         │
└─────────────────────┘
           │
           ▼
┌──────────────────────────────────────────┐
│ IOUT_OC_FAULT_LIMIT (Converter A, 10 A);  │
│ VIN_ON (Converter A, 10 V);               │
│ TON_RISE (Converter A, 5 ms);             │
│ STORE_DEFAULT_ALL (Converter A);          │
└──────────────────────────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Remainder        │
│   of the Host       │
│    Program          │
└─────────────────────┘
```

Fig. 4. Simple PMBus command sequence.

PMBus commands can generally be grouped into several categories. Some typical categories are:
- On, off, and margin testing
- Output-voltage related
- Addressing, memory, communication, and capability
- Fault management
- Sequencing
- Status
- Telemetry
- Other

Commands for on, off, and margin testing deal with turning the converter on and off as well as setting up parameters for margin testing, and for startup and shutdown timing. The commands in this relatively small group are OPERATION, ON_OFF_CONFIG, VOUT_MARGIN_HIGH, and VOUT_MARGIN_LOW. Voltage margining is the intentional variation of voltages in the system to check for functionality at extremes of operating tolerances to ensure overall system reliability in the field. The VOUT_MARGIN_HIGH and VOUT_MARGIN_LOW commands tell the converter what the margin output voltages will be when output margining is commanded. The OPERATION command is used to command the converter to start, stop, and change the output voltage to one of the previously configured margin voltages. ON_OFF_CONFIG is used to determine what conditions must be met for the converter to start operation and how it shuts down. Some possibilities for start-up include starting as soon as power is applied or waiting for a signal from either the CONTROL pin or the PMBus (or both); for turnoff the controller could use the preprogrammed turn-off delay or just stop as quickly as possible.

Several commands are available that relate to setting the output voltage for the converter. Some of the more useful commands in this group include VOUT_COMMAND, VOUT_TRIM, VOUT_CAL_OFFSET, VOUT_SCALE_LOOP, and VOUT_SCALE_MONITOR. These commands have several functions, including changing and calibrating the converter output voltage and calibrating the output-voltage monitor.

Addressing, memory, communication, and capability commands deal with storing and retrieving default operating parameters in nonvolatile memory on the controller. They can protect the controller from inadvertent changes to operating parameters. For controllers that support multiple rails, these commands also specify which rail (or which phase of a multiphase converter) subsequent commands apply to. There is even a command for querying the converter about what it supports. Some commands that fall into this group include STORE_DEFAULT_ALL, RESTORE_DEFAULT_ALL, STORE_DEFAULT_CODE, RESTORE_DEFAULT_CODE, WRITE_PROTECT, PAGE, PHASE, and QUERY.

Fault-management commands allow the user to determine how the converter responds to fault conditions and to define what those fault conditions are. The PMBus specification allows for very flexible handling of a variety of faults including

both overcurrent and overpower on the input and output, over- and undervoltage on the input and output, and over temperature. The response can generally be programmed to ignore the fault; shut down and try to restart; or latch off and require a restart to be initiated by a PMBus command, a power cycle, or toggling the control (CNTL) pin. Additionally, the action taken on the fault can generally be programmed to be immediate or to take place after the fault condition persists for a period of time. Some of the commands that would fall into this fault-management category are IOUT_OC_FAULT_LIMIT, IOUT_OC_FAULT_RESPONSE, IOUT_OC_WARN_LIMIT, OT_WARN_LIMIT, OT_FAULT_LIMIT, OT_FAULT_RESPONSE, VIN_UV_WARN_LIMIT, VIN_UV_FAULT_LIMIT, and VIN_UV_FAULT_RESPONSE. There are many other commands in this category for the other types of faults that are supported.

For example, the commands IOUT_OC_FAULT_LIMIT and IOUT_OC_FAULT_RESPONSE allow the user to program the overcurrent trip threshold and to specify how the converter should respond when the overcurrent limit has been exceeded. The options for response to an overcurrent event include continuing operation while maintaining the output current set by the IOUT_OC_FAULT_LIMIT command; maintaining the preset current output and shutting down when the output undervoltage threshold has been reached; maintaining the output current and shutting down after a preset timeout; and shutting down immediately. The converter can also be programmed to restart in a variety of ways, including staying shut down until the fault is cleared in the controller; trying to restart a preset number of times and, if that fails, staying off until commanded to restart; and trying to restart indefinitely. Although the response to overcurrent faults is very flexible, not all possibilities can be implemented in all converters.

When a fault occurs, a status register bit is set and remains set so that it can be read by the user. If the fault response is set to require user interaction to restart the converter, these fault indication bits must be reset with a command called CLEAR_FAULTS. This will reset all the fault indicators and allow the controller to restart when commanded to by the OPERATION command or when the control line is toggled. If the fault condition still exists after the CLEAR_FAULTS command is given, the status bit for that fault will immediately reset, and the converter will respond as programmed by the appropriate fault-response command.

Sequencing commands control the timing of converter start-up and shutdown. These commands set how long the converter waits after receiving a command to start before it begins ramping up its output voltage; how long the ramp-up time itself is; and possibly even how long to wait before beginning to shut down after receiving a command to do so. There are also commands that can specify how long a converter is allowed to try to start up and that can throw a fault condition if it takes too long, as well as commands that set a time limit for the output voltage to fall once the command to shut down has been given. Commands in this group include TON_DELAY and TON_RISE, which affect the start-up of the converter. With the TON_DELAY command, a converter can be set to start up at a preset time after it is enabled, which can be useful for power-supply sequencing. TON_RISE controls the start-up ramp rate of the converter after it begins operation. The effect of these two commands is shown conceptually in Fig. 5. The enable signal can come from several sources: meeting the input UVLO threshold (also programmable with VIN_ON), using the CONTROL pin, or controlling the PMBus with the OPERATION command.
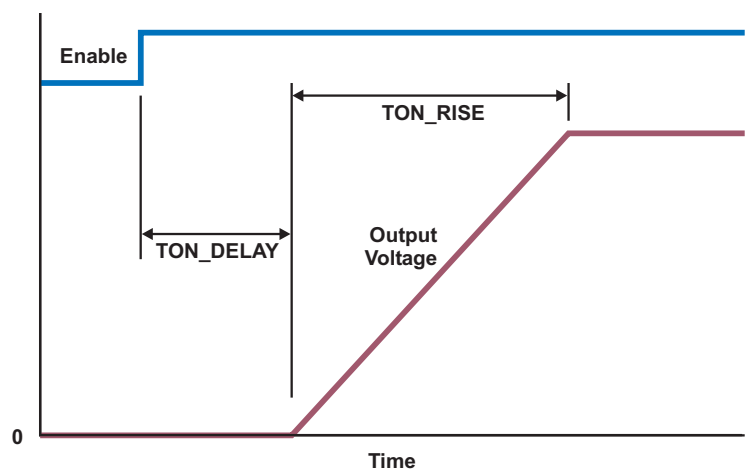
Fig. 5. Relationship of TON_DELAY and TON_RISE.

Similarly, TOFF_DELAY and TOFF_FALL affect the time delay from the removal or de-assertion of the enable signal until the output begins to fall, and the amount of time it takes for the output to fall to zero. The effect of these two commands is shown graphically in Fig. 6. The source of the enable signal is the same as before.
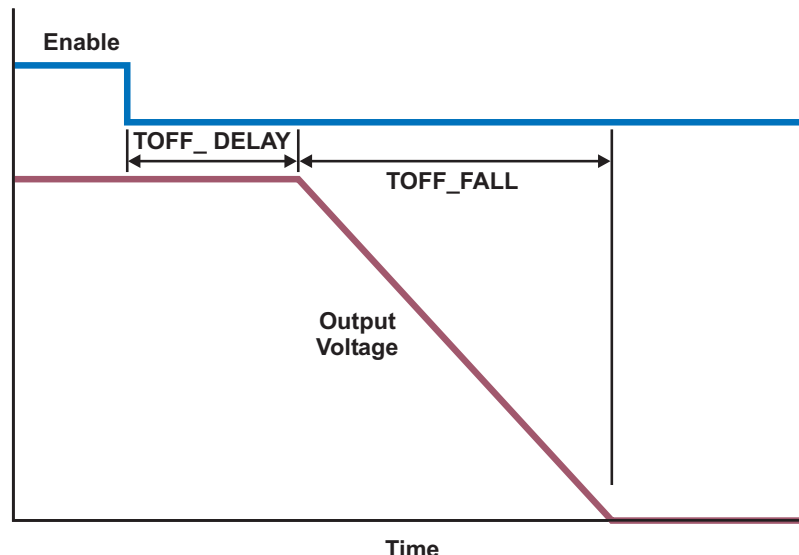
Status commands allow the user to query the converter and receive information about its state. Some of these commands include STATUS_BYTE, STATUS_WORD, STATUS_VOUT, STATUS_IOUT, and STATUS_CML (communication, memory and logic), among others. STATUS_BYTE and STATUS_WORD return a single byte and two bytes, respectively, that give a snapshot of the state of the converter as shown in Fig. 7. Note that STATUS_BYTE is simply the low-order byte in STATUS_WORD and contains the flags for the most common problems that are likely to occur. If the system host is busy or the PMBus has a lot of traffic, using STATUS_BYTE instead of STATUS_WORD when querying controllers could prove beneficial. Depending on the results of the STATUS_WORD or STATUS_BYTE command, the user can obtain further detail by using one of the other status commands. For example, if the STATUS_BYTE has bit 4 (the overcurrent flag) set, issuing the STATUS_IOUT command will return a byte describing the indication in more detail.

Telemetry commands allow the user to query the device about various operating parameters of the converter. Commands in this group include READ_VIN, READ_VOUT, READ_IIN, READ_IOUT, READ_TEMPERATURE, READ_DUTY_CYCLE, READ_PIN, and READ_POUT, among a few others. These commands are useful for monitoring the converter and the load operation over time for data logging, system health monitoring or the prediction of failure of either the load or the converter, and general troubleshooting.

Commands that fall into the "other" group category have varied functions. A few examples are FREQUENCY_SWITCH, VIN_ON, VIN_OFF, POUT_MAX, and the fan-control commands. FREQUENCY_SWITCH is used to alter the

Fig. 6. Relationship of TOFF_DELAY and TOFF_FALL.

STATUS_WORD
High Byte
7– VOUT
6– IOUT/POUT
5– INPUT
4– MFR
3– POWER_GOOD
2– FANS
1– OTHER
0– UNKNOWN
Low Byte
7– BUSY
6– OFF
5– VOUT_OV
4– IOUT_OC
3– VIN_UV
2– TEMPERATURE
1– CML
0– None of the above

STATUS_BYTE
7– BUSY
6– OFF
5– VOUT_OV
4– IOUT_OC
3– VIN_UV
2– TEMPERATURE
1– CML
0– None of the above

Fig. 7. STATUS_WORD and STATUS_BYTE.

switching frequency of the converter, while VIN_ON and VIN_OFF set the input UVLO turn-on and turn-off thresholds, allowing complete flexibility for power and other variations in any given system. POUT_MAX is used to define the maximum power that the converter is allowed to provide. Exceeding the limit set here will trigger a fault with a programmable response similar to those described earlier.

IV. USING PMBUS IN THE SYSTEM

The preceding brief introduction to PMBus describes only a portion of the commands available to the user. It is worth noting that any individual PMBus-enabled device will probably not support all of the commands previously listed. The command set is diverse; some commands are targeted toward off-line power conversion, others toward point-of-load conversion, and still others toward both. When choosing a converter for use in any given application, it is best to first determine what tasks the device will need to perform and what it will require to perform them. Then those requirements can be balanced against what the available devices offer and at what cost—just like any other engineering decision.

A. Sequencing

A very common requirement of power supplies is that they sequence, or start up and shut down, in a particular order. If the converter supports the required commands, this task is easy. There are several ways in which PMBus converters may be sequenced:

1. Remote or system host enable
   a. With the CNTL pins
   b. With the OPERATION command
2. Time-delay sequencing with TON_DELAY and TOFF_DELAY
3. Open-loop tracking by manipulation of TON_RISE, TOFF_DELAY, and TOFF_FALL

Case 1a is illustrated in Fig. 8 and really does not require a PMBus; all that is required is a converter with a hardware pin for enabling and shutting down. This will control the power-up sequence and to some extent the power-down sequence. Power-up starts when the host enables converter A. Once converter A has started and is operating, it will assert its Power Good signal, alerting the host that it can enable converter B. Converter B will assert its Power Good signal after it has successfully started. Power-down sequencing (if a reverse sequence is assumed) can be accomplished by the host disabling converter B. After some time the host can disable converter A. The amount of time it needs to wait between disabling each converter is dependent on the system and must be known by the host. This
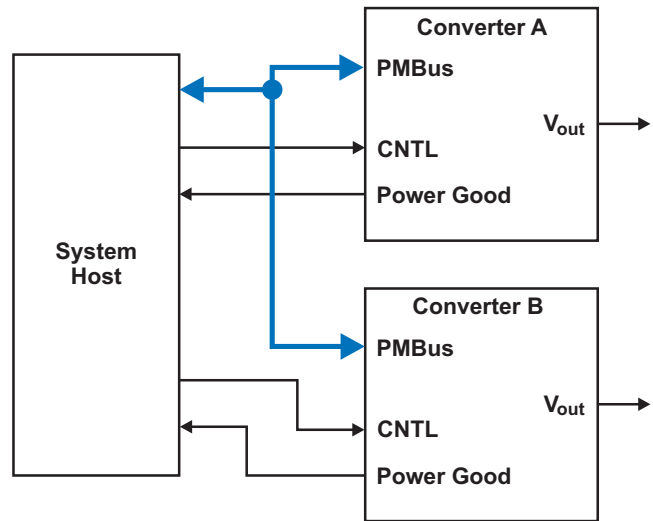

Fig. 8. Hardware sequencing.

places a burden on the host to time the converter shutdowns.

Case 1b is a similar scenario. Here the host enables and disables the converters by sending the appropriate data byte to each converter with the OPERATION command. This method of control is also illustrated in Fig. 8, except that the CNTL lines are not required to be routed on the printed circuit board (PCB). To accomplish this method of sequencing, the host sends converter A the OPERATION command with the bits in the data-byte set to cause the converter to start up. After a sufficient amount of time has passed, the host sends the same OPERATION command to controller B. It is incumbent upon the host to know how long to wait and to keep track of the time between commands, just as in the shutdown phase of Case 1a. The trade-off in using the OPERATION command rather than the CNTL pins is a little less PCB routing for a little more overhead in the system's host software.

Case 2, which accomplishes time-delay sequencing using the TON_DELAY and TOFF_DELAY commands, gives results similar to those in Case 1 but eliminates the need for the host to keep track of the time between issuing commands to the converters. Instead, the converters are initialized with preset turn-on and turn-off delays that they use to determine how long to wait after receiving an enable or disable command before actually turning on or off. The host sends only one command sequence to enable/disable both (or more) converters.

Topic 6

To sequence converters in this manner, each converter must be initialized correctly. First, the ON_OFF_CONFIG command must be issued to each converter to cause it to use the delays set by TOFF_DELAY. Next, for the converter that is to start first, TON_DELAY should be set to zero or some small value. For the converter that is to start next, TON_DELAY is set to some time longer than the start-up time for the first converter. This start-up time is typically set with the TON_RISE command or by hardware programming via some external resistor or capacitor. Once these values are set in the converter, they can be commanded to start and will sequence their power up properly. This is illustrated in Fig. 9. Note that the enable signal can come either from the PMBus or from a hardware-enable pin.

The setup and execution for the power-down sequence is similar. The TOFF_DELAY settings determine which converter will shut down first and which will shut down later. Fig. 10 illustrates the timing relationships.

Here the fall time of each power supply is determined by the parameter TOFF_FALL. Setting TOFF_FALL properly depends on the characteristics of the load the converters are supplying. If the converter is not allowed to sink current from the load at any time, the TOFF_FALL times must be set longer than the natural decay times that the converters have when operating at the minimum expected load. If the converter is allowed to sink current, the TOFF_FALL time may be set as the user desires. Of course, the converters may be set to ignore the fall-time parameters and allowed to decay naturally as in the previous examples. If supported by the converters, and if required, the previous examples may incorporate the TOFF_FALL parameter for more deterministic control of the power-down sequence.

Tracking is a feature designed to keep multiple voltage rails close together during start-up and shutdown. The purpose is to prevent unwanted current that may be destructive to the load from flowing from one rail to the other during power-up and power-down. Tracking start-up can be defined as two or more rails following each other closely as they rise, with the lowest voltage rail stopping its rise as it reaches its final regulation value. The other rails continue to rise together, and each one stops as it reaches its final regulation value. Tracking shutdown is similar. The highest voltage rail is brought down at a controlled rate. As it reaches the value of the next lower rail, both of
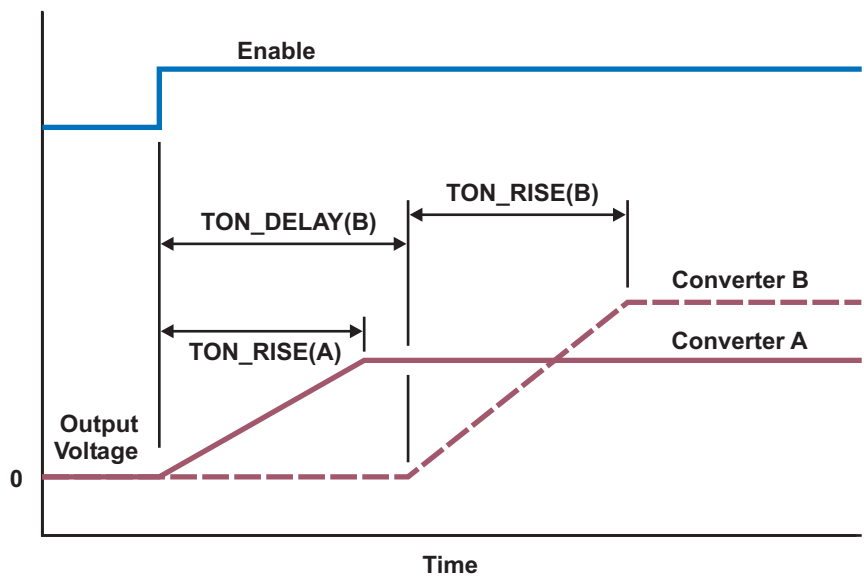

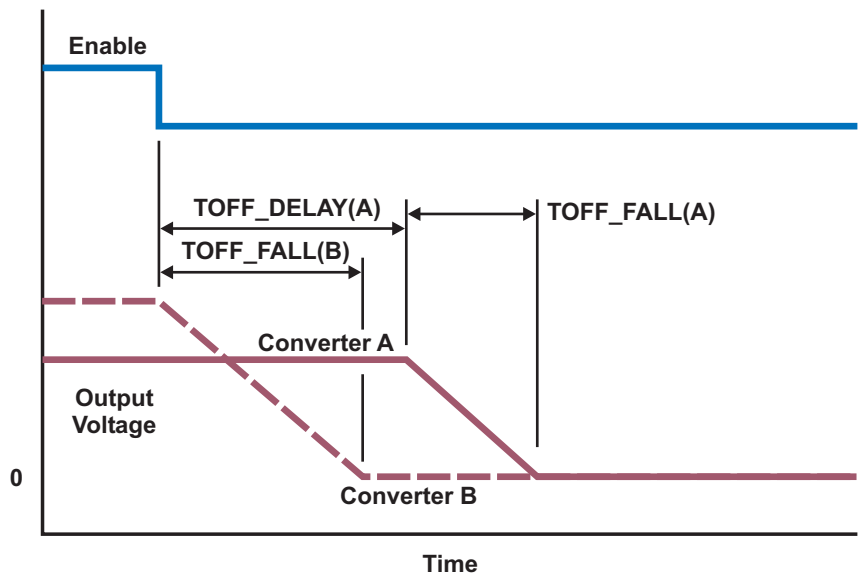Fig. 9. Time-delay start-up sequencing.

Fig. 10. Time-delay power-down sequencing.

these rails continue down together and maintain a negligible voltage difference between them. The process can be extended to any number of voltage rails.

Tracking can be done with PMBus converters that support the TON_RISE command for start-up and the TOFF_DELAY and TOFF_FALL commands for power-down. Fig. 11 illustrates this and the required settings for start-up. The TON_RISE setting for the two (or more) converters is set so that the slope of the output-voltage-rise waveforms is the same. If both converters receive the enable signal at the same time, the output voltages will track one another until converter A reaches its steady-state voltage. The output of converter B will then continue on to its steady-state output voltage at the same slope. This will require converters that allow fairly fine control of the TON_RISE parameter. The setting for TON_RISE can be calculated by making the slopes of the two start-up waveforms equal:

$$\frac{V_{OUT(A)}}{TON\_RISE(A)} = \frac{V_{OUT(B)}}{TON\_RISE(B)} \quad (1)$$

The converters can be set up to track on a power-down as well. Adjusting the TOFF_DELAY parameter of the lower-output-voltage supply and the TOFF_FALL parameters of both converters causes the output voltages to track each other when the converters are shut down. The waveform slopes of the TOFF_FALL parameters must be set to match each other like the TON_RISE parameters in tracking start-up. The TOFF_DELAY parameter of the converter with the lower output voltage must be set so that that voltage starts to fall as soon as the higher output voltage of the other converter falls enough to match it. This is illustrated in Fig. 12. Equations (2) and (3) show the calculations to set the parameters for power-down tracking, given the conditions in Fig. 12.

$$\frac{V_{OUT(A)}}{TOFF\_FALL(A)} = \frac{V_{OUT(B)}}{TOFF\_FALL(B)} \quad (2)$$

$$TOFF\_DELAY(A) = \left(V_{OUT(B)} - V_{OUT(A)}\right) \times \left(\frac{TOFF\_FALL(B)}{V_{OUT(B)}}\right) \quad (3)$$

The tolerances required for a converter's timing and output voltage to meet specific targets for tracking accuracy are beyond this topic's scope.
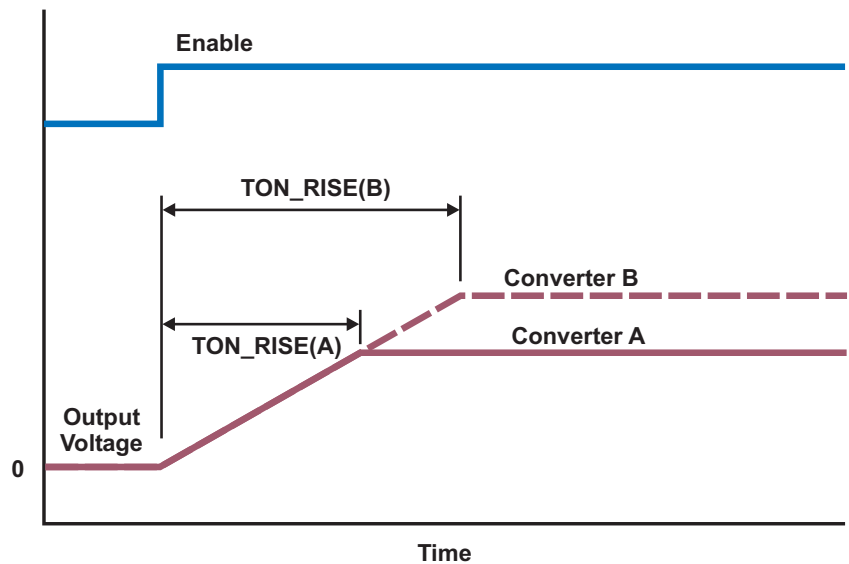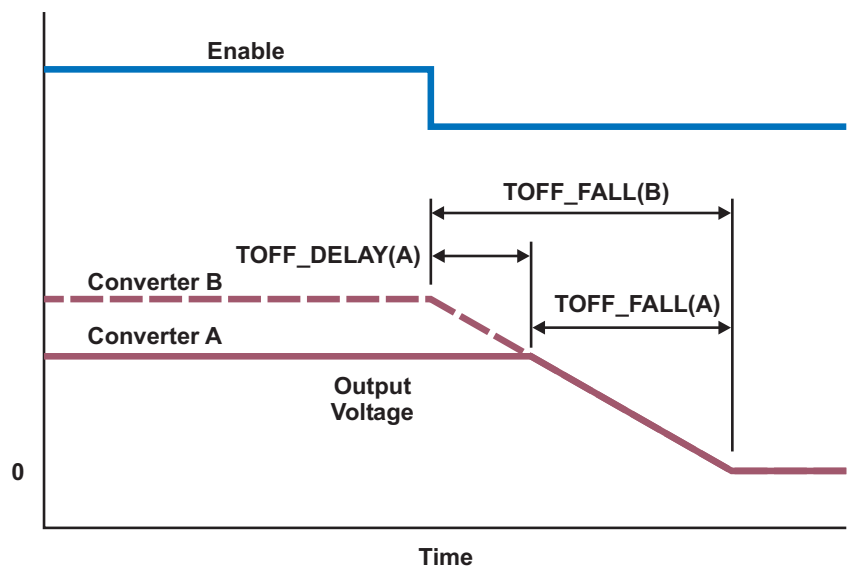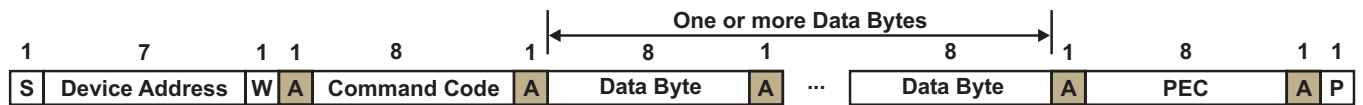

Fig. 11. PMBus tracking start-up.

Fig. 12. PMBus tracking shutdown.

The astute observer will notice that the nature of the PMBus as a serial communication bus seems to preclude sending these startup and shutdown commands simultaneously to both converters. This would introduce some unknown variability in the timing that would have to be accounted for when the delays for start-up and shutdown are set. PMBus has a mechanism for overcoming this limitation called the group command protocol. With this protocol, it is possible to issue commands to several converters in sequence and then have the converters execute the commands simultaneously.
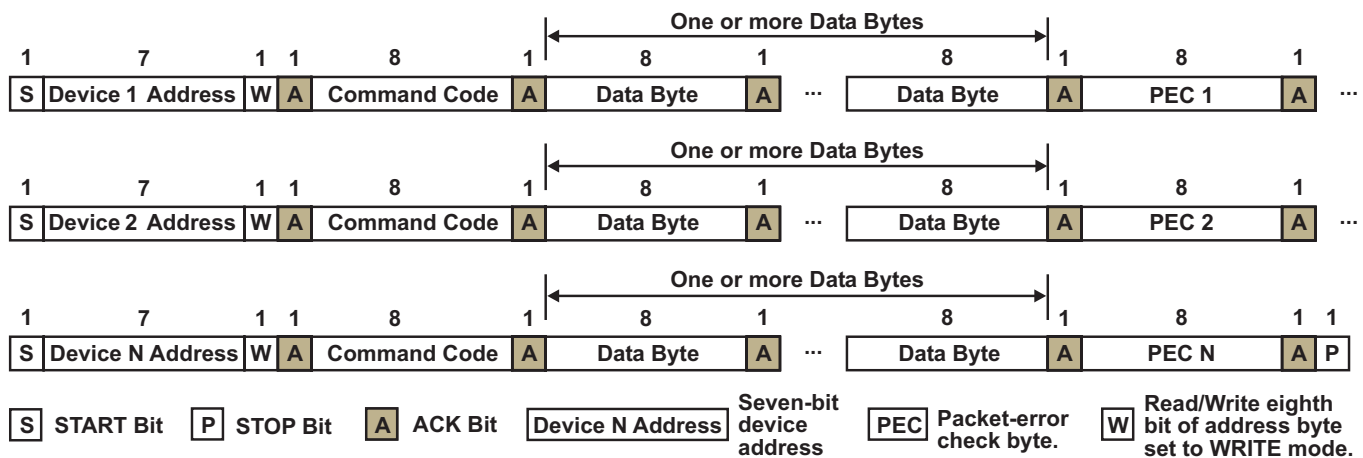
The normal way a command is written to a PMBus device is shown in Fig. 13a. The host puts a START on the bus, followed by the address of the device with which it wishes to communicate. The device then signals an acknowledgement (ACK) of the address. The host responds by sending the command, after which the device signals another ACK. The host then sends the data for the command, followed by the slave-device's ACK. The packet error check byte (PEC) is then sent and an ACK is received. The host then puts a STOP on the bus, causing the target device to execute the command.

Slightly modifying this procedure makes the first PMBus converter(s) delay execution of the command so that all can execute a command simultaneously. This is done as follows. The sequence is the same as before, except that instead of putting a STOP on the bus when writing the command to the first target is completed, the host puts another START on the bus followed by the address of another target. The new target acknowledges the address, and the sequence proceeds as before until writing the command is completed. The host can then put a STOP on the bus or continue writing commands to other targets by placing another START on the bus and beginning the sequence for another target. When the host has finished writing all commands to all targets that are to be executed simultaneously, it puts a STOP on the bus. At this point, all converters will execute the commands that they were given (see Fig. 13b). The key point is that the STOP on the PMBus is the signal for a target converter to execute the command it was given. There are a couple of key points to remember when using the group command protocol. First, only one command can be given to any one target device in any one group command. Second, the commands given must not require the target to transmit data back to the host.

a. Single command to a PMBus target.

b. Group command to multiple PMBus targets.

Fig. 13. Single and group commands on PMBus.

Fig. 14 shows the typical CLK/DATA timing relationships required to assure that the DATA signals are valid during CLK-signal transitions.

Fig. 15 shows two examples of a command-byte exchange between the transmitter and receiver. Each byte has 9 bits, the 8 data bits and an ACK/NACK bit. The transmitting device places the 8 data bits on the bus, then stops. Before the 9th CLK pulse, the receiving device pulls the DATA line low to signal an acknowledgment (ACK) of the byte or leaves it high to signal a negative acknowledgment (NACK) of the byte.

In the group-command protocol, the system host will be the only transmitter. In some other command transactions, the converter or slave can transmit data to the host. Therefore, the role of transmitter and receiver in Fig. 15 depends on the type of command. In all cases, the host will control the CLK line.

B. Voltage Margining

Another task that is made easier with a PMBus converter is voltage margining. As stated earlier, voltage margining is the intentional variation of voltages in the system to check for functionality at extremes of operating tolerances to ensure overall system reliability in the field. Normally, implementing margining requires a converter that specifically offers this feature. However, a

secondary integrated device can be used instead, or a few discrete components that can control or offset the output voltage of the converter via a bias injected into its feedback loop or a change of components on the test board. Each of these alternatives has advantages and disadvantages in terms of size, cost, and ease of implementation.
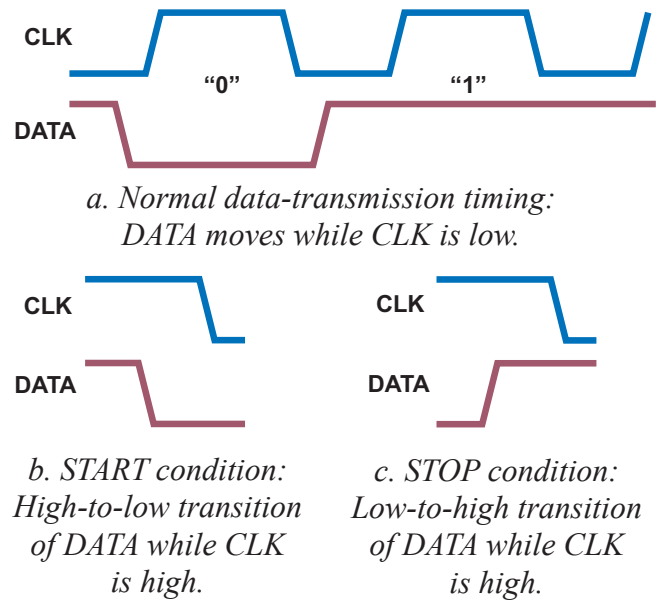

a. Normal data-transmission timing: DATA moves while CLK is low.

b. START condition: High-to-low transition of DATA while CLK is high.

c. STOP condition: Low-to-high transition of DATA while CLK is high.

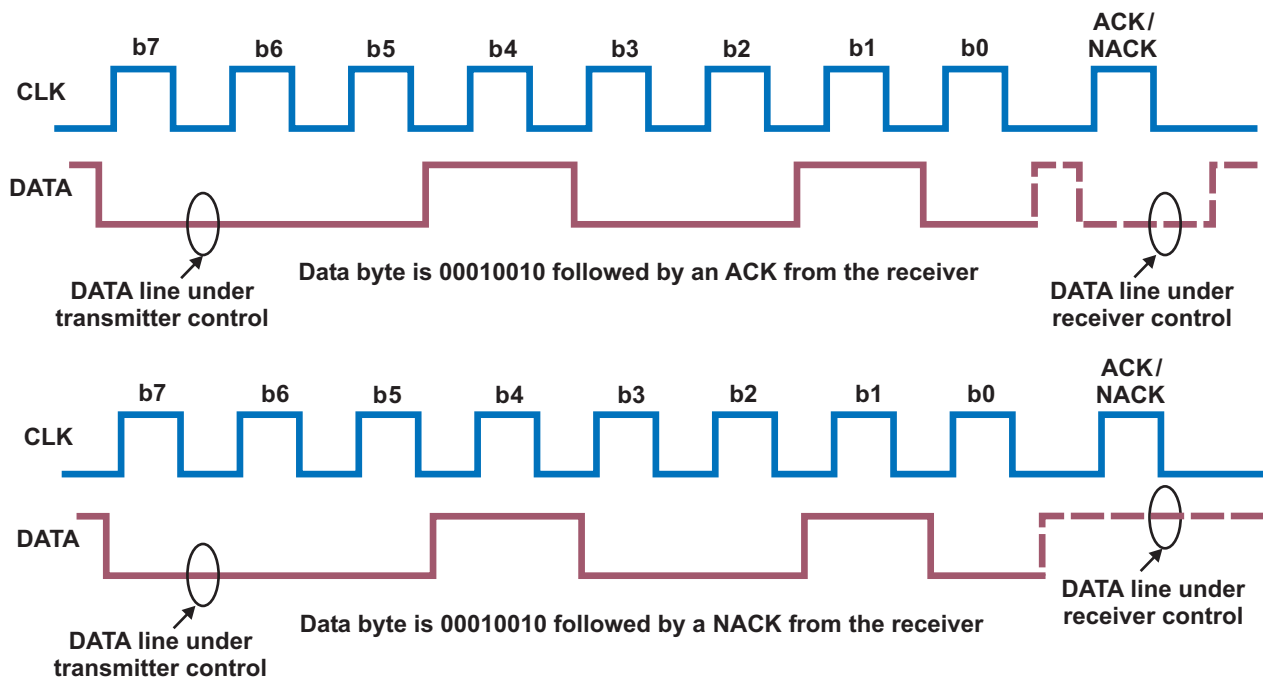*Fig. 14. PMBus data bits during normal transmission and under START and STOP conditions.*

Fig. 15. PMBus data byte with ACK and NACK acknowledgements.

Margining can be implemented with a PMBus converter that supports VOUT_MARGIN_HIGH, VOUT_MARGIN_LOW, and the margin-related bits in the OPERATION command data byte. The desired margin voltage for the target device must first be set with a VOUT_MARGIN_HIGH or VOUT_MARGIN_LOW command; then the OPERATION command is given with the bits set to tell the converter to change to the margin output voltage.

There are a couple of unique features in the PMBus implementation of margining that are noteworthy to the system designer. First, the margin transition rate can be adjusted. The VOUT_TRANSITION_RATE command can be used to set the rate of change of the output voltage when the margin command is issued. This can be helpful in some circumstances to avoid fault conditions in the system. Second, the converter can be set to ignore faults while the margin is in effect. The margin bits in the operation data byte allow four settings for the margin command:
1. Margin high, acting on faults
2. Margin high, ignoring faults
3. Margin low, acting on faults
4. Margin low, ignoring faults

These commands allow system testing beyond normal operating parameters without the controller generating a fault signal that would otherwise cause the system to halt operation or reset. Testing is therefore faster, and tests that otherwise might not be conducted are possible. An example of such testing would be to set the margin's high voltage above the operating limit for output overvoltage. Multiple converters can obviously participate in margin testing, and the group command protocol allows several converters to adjust their output voltages at the same time.

C. Programmable Parameters

The PMBus also allows a system's power parameters to be changed either during factory testing or in the field at any time without the need for hardware modification. Most changes can be accomplished with a simple software update or a one-time parameter update of the converter during testing. For some applications powering power-hungry, low-voltage ASICs, the output voltage of the converter can be changed to match the specific requirements of the ASIC. This allows use of a standard converter circuit to power ASICs whose input-voltage requirements to meet performance specifications may vary from ASIC to ASIC. A feature of the PMBus converter that is particularly beneficial here is its ability to store user settings as the default that will be used every subsequent time the converter is powered up. For example, during system testing, if it is known that a particular ASIC needs a certain voltage to perform to specifications, the controller's output voltage can be set to that value and written to nonvolatile memory for later use as the default setting.

Another use of this feature would be to trim the converter's operating characteristics at test time to allow tighter specification of overall tolerances for the converter. Specific parameters that could be trimmed to offer a more competitive advantage would include overcurrent threshold, output voltage, current-sense gain and offset for the READ_IOUT command to be more accurate, etc.

When adjusting the output voltage of a converter, especially when it is in a system, the user must take care to observe the voltage limitations of any devices connected to the output of the converter. Accidents do happen, especially during the development phases of a project, and a transposed number or other error could cause the converter's output voltage to be greater than the load can withstand. A command designed to deal with this issue, VOUT_MAX, is supported by some converters. This command places a strict upper limit on the output voltage that the converter can command, irrespective of what the user tries to have the converter do. This can save valuable system prototypes and repair time, speeding the development process.

D. System Monitoring

System monitoring is another common PMBus task, but maybe not one usually associated with the power converter in the system. Depending on what the converter supports, there are several important system parameters that can be obtained from it using PMBus. The READ_TEMPERATURE_X commands can be used to flag a rise in the temperature of either the converter itself or some other point in the system designated

by the system designer and converter manufacturer. This information, along with other information available via the PMBus, can help pinpoint and diagnose common system issues that might occur in the field, adding value to the overall system. For instance, if a converter has fans that are under PMBus control, a high temperature reading combined with a READ_FAN_SPEED_N that is too low could indicate a failed fan or possibly a blocked air filter if the fan-speed reading is too high. An alert could be sent to maintenance personnel by the system host if desired. If temperatures are high and the fan speeds are normal, a high ambient temperature could be the cause. One of the readings from the READ_TEMPERATURE_X commands could report this directly if the ambient temperature was one of the temperatures being sensed. For converters that support reading the input and output power (or reading the input and output voltages and currents and letting the system host do more of the calculating), converter efficiency can be computed. Over time, a drop in converter efficiency can signal a converter with problems that might need to be addressed, such as corroded or loose connectors, mechanical damage, or dust buildup that might cause ineffective cooling of the power semiconductors, or some other condition. This trend could be analyzed and preventative maintenance scheduled before a system failure occurs.

The PMBus specification has defined specific commands to deal with controlling and monitoring four cooling fans. Conventional thinking would have the user place these fans in the converter itself, but there is no reason why they could not be placed outside its physical boundaries. If the overall configuration of the system is amenable, the PMBus power converter can be used as a source of information to drive cooling fans for the whole system. The converter could be made to accommodate fans that are primarily for system cooling as well as fans that are primarily for cooling the converter. Along with the ability to read temperatures outside the converter itself (depending on what temperatures the READ_TEMPERATURE_X commands are monitoring), the system host has a single place to control up to four system cooling fans. If supported by the hardware, PMBus allows for complete variable-speed fan control, allowing the user to optimize the system power used for cooling versus the amount of cooling required. This could provide benefits such as reduced power consumption and lower acoustic noise output from the system.

E. Other PMBus Features

A significant problem for the manufacturer of power modules is accommodating the wide variation possible in external capacitive loading. The control loop of the module will work optimally only for a relatively small range of external output capacitance. Performance will degrade and could become unstable for capacitive loadings beyond this range. PMBus, along with a digital control scheme, can be used to overcome these limitations and optimize the module for the particular load conditions. The converter manufacturer can use some of the undefined manufacturer-specific commands to set the feedback-loop parameters. Or, if the controller has some processing capability, as would be likely in a digitally controlled loop application, a manufacturer-specific command could be set up to calculate the optimal feedback-loop parameters from the amount of external load capacitance. Either case would allow optimization of the loop response for the particular application at hand, maximizing system performance.

PMBus-enabled converters could also be useful for tracking faults and correlating failures to specific manufacturing runs, certain suppliers, particular revisions of converters or silicon, or even a particular silicon lot. This can be done through two groups of commands, the manufacturer-information commands and the user-data commands.

The manufacture-information commands allow the user to retrieve identifying information stored by the converter manufacturer in nonvolatile memory. These commands—MFR_ID, MFR_MODEL, MFR_REVISION, MFR_LOCATION, MFR_DATE, and MFR_SERIAL—provide a complete electronic description of the converter that can be used to trace it as far back as desired in the manufacturing process. For those applications that have an embedded converter built from discrete components, the manufacturer-information commands could pertain to silicon and not to the entire converter.

User-data commands are 16 general-purpose commands that enable the user to store arbitrary data in nonvolatile memory in the converter. This data can be anything from a copy of the operating parameters as a backup in case of a programming accident, to identifying system information as an aid in tracking performance and failures, or anything else that the user desires. The implementation of these commands is not defined in the PMBus specification and is left to device manufacturers to decide. It is therefore paramount that the user understand the limitations and capabilities of a particular device if this functionality is needed.

This topic is not by any means a complete treatment of the PMBus or the capabilities and application possibilities of a PMBus-enabled converter. It is intended to illustrate only some of the more common and potentially useful application aids that PMBus supports. In the interest of saving space, no mention has been made of multiple-phase capabilities or of converters with multiple output rails and the specific advantages the PMBus provides for them. Also, as mentioned earlier, not all converters will implement all commands, and the commands that are implemented are dependent on the target application. With that said, a look at an application and the selection of a converter or controller for a converter would be in order.

V. EXAMPLE APPLICATION AND CONVERTER SELECTION

A hypothetical application might be for a converter to power two processors on a board in a communications rack. The processors both require a 1.2-V nominal core voltage and separate 3.3-V I/O voltage rails so each processor can be shut down independently. Output-current requirements are 10 A per processor core and 3 A per I/O rail. The core voltages must be present before the I/O voltages are applied. A processor's specified optimum operating voltage can vary from 1.1 to 1.2 V on a device-by-device basis to balance performance and power dissipation. It is desirable to use this core-voltage range to lower overall system power requirements and avoid unnecessary heat loading in the rack. In addition, the testing and qualification requirements require that power supply rails in the system be varied over a specified

operating range during factory testing and during field testing by service personnel. To monitor power consumption, it is desirable to monitor output voltage and current, however, cost is always an important concern. With the given objectives, the converter should support the following:

1. Nonvolatile memory. All PMBus devices should have this feature, however, some low-cost offerings may not. This memory feature allows device settings to remain as defaults so the device does not need to be configured each time the device is powered.

2. Sequencing. There are four rails (two sets of two, a core and an I/O rail in each set) that have to be supplied with the stipulation that in the two sets of rails, the I/O voltage must not start until the core voltage has been established.

3. VOUT_TRIM command. This will allow the user to set up a default operating voltage that matches the requirements of the particular processor on the system board.

4. Margining commands—VOUT_MARGIN_HIGH, VOUT_MARGIN_LOW, and the support of the margin bits in the OPERATION command (which would be implied by the implementation of the first two commands). Support for the VOUT_TRANSITION_RATE command could be useful but is not absolutely necessary.

5. READ_VOUT and READ_IOUT commands that let the host calculate power output. If the host is severely limited in processing power, READ_POUT is a nice option to place some of the processing load on the converter.

With these requirements, a full blown implementation of PMBus is not necessary for the converter. With the low-cost objective, a multiple-rail controller can provide a more compact, cost-effective total-system solution. One controller that can support all the requirements of this application is the UCD9240. Additional features in the UCD9240 that could benefit the overall system and off-load tasks from the main system host include temperature measurement and cooling fan control.

The UCD9240 is a digital synchronous-buck PWM controller that can control up to four power rails at a time. It also has PWM outputs to drive a

total of eight power stages. The power stages can be configured as combinations of single- and multiple-phase power rails for up to the four supported rails. Each of the four high-speed digital-control loops have dedicated 3-pole/ 3-zero compensators and pulse-width modulators with 250-ps resolution. This architecture enables wide input-to-output voltage-conversion ratios at a switching frequency of up to 2 MHz. The UCD9240 is also able to monitor and manage power-supply operating conditions and report the status to the host system through the PMBus interface. The device operating parameters are configurable using the Power+ Designer tool available from Texas Instruments. This PC-based design tool allows the power-supply designer to easily configure the control-loop characteristics and generate the expected performance by displaying Bode plots for each controlled power stage.

Figures 16 and 17 show the block diagram of the UCD9240 and a schematic that would satisfy the previously specified requirements.
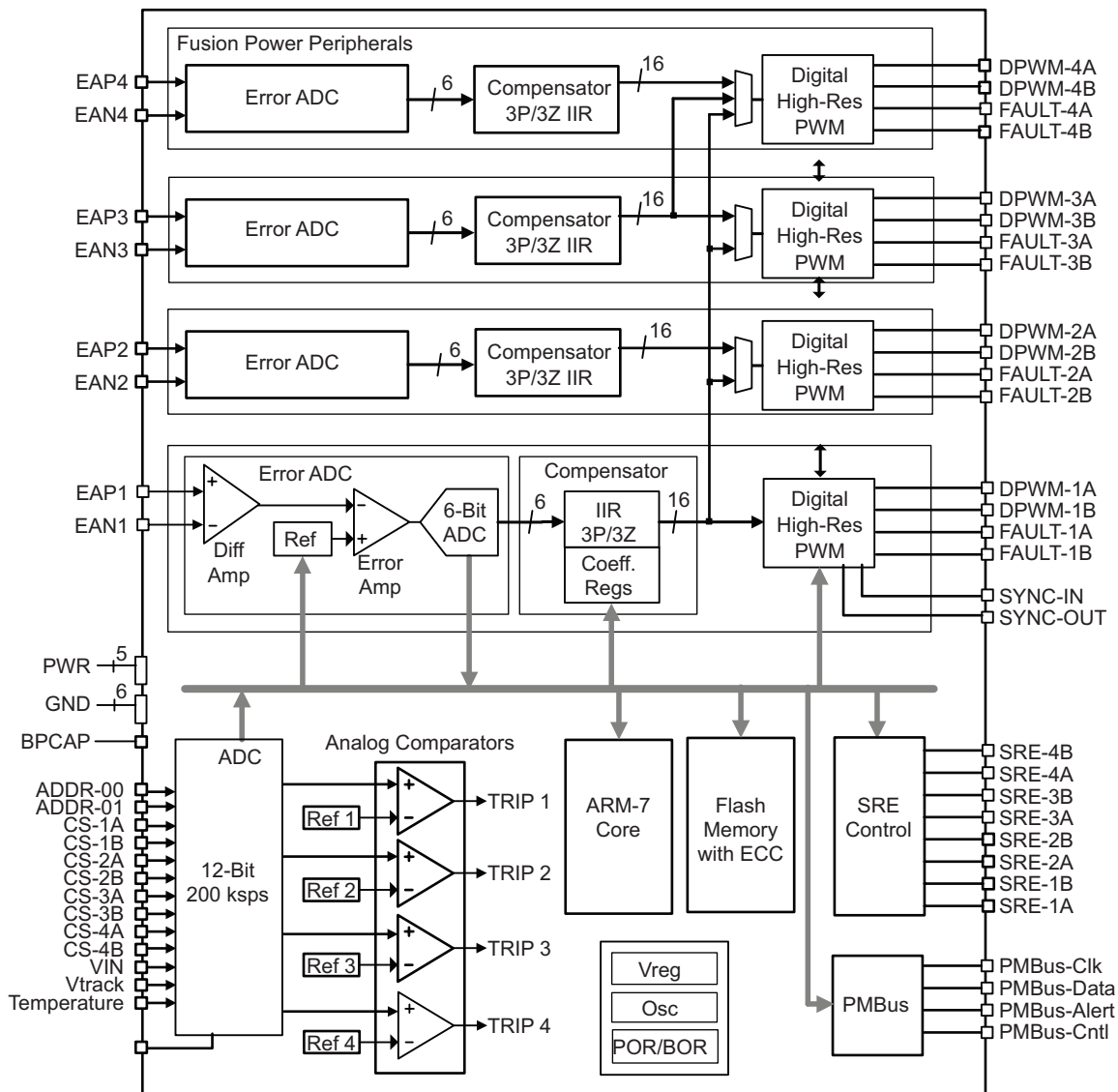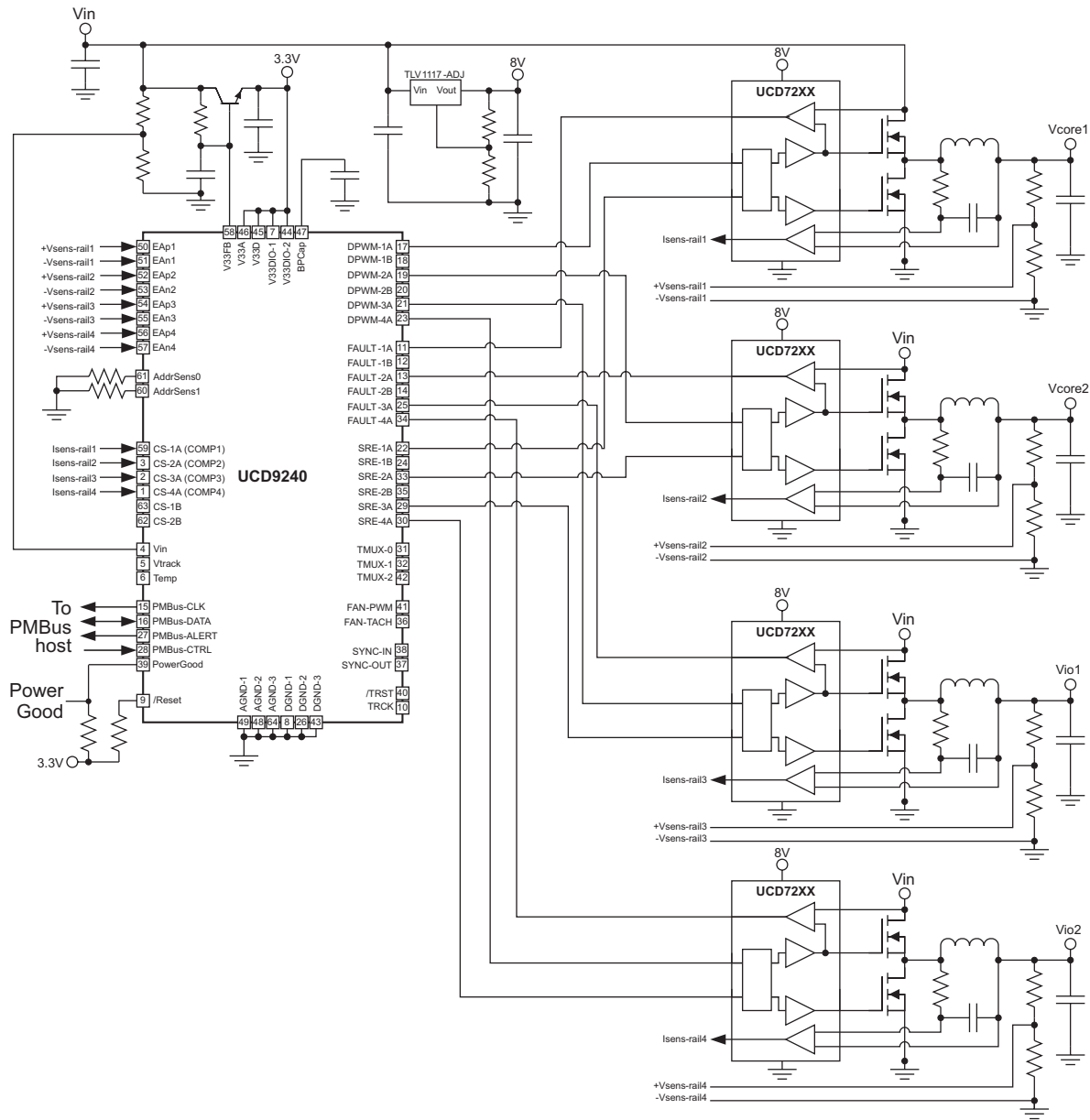

Fig. 16. UCD9240 block diagram.

Fig. 17. Schematic of UCD9240-based solution.

Topic 6

VI. REFERENCES

[1] PMBus Specification Version 1.1, Feb. 5, 2007. [Online.] Available: http://pmbus.org/specs.html

[2] I^{2}C-Bus Specification, Version 2.1, Jan. 2000. [Online.] Available: http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf

[3] SMBus Specification Version 2.0, Aug. 3, 2000. [Online.] Available: http://smbus.org/specs/

[4] Bob White, "PMBus Offers Open-Standard Power Management," *Power Electron. Technology*, Sept. 2005. [Online.] Available: http://powerelectronics.com/mag/power_pmbus_offers_openstandard/

[5] Dave Freeman, "Power System Communications Support Digital-Power Growth," *Electron. Design News*, March 26, 2005. [Online.] Available: http://www.ednasia.com/article-3232-powersystemscommunicationssupportdigitalpowergrowth-Asia.html

[6] UCD9240 datasheet, April 2007. [Online.] Available: http://www.ti.com/lit/gpn/ucd9240