AN_65XX_044 **JANUARY 2011**

Meter Design for Power Failure Events

This document describes hardware and firmware methods that can be used in TERIDIAN meter chips to cope with sag and power-failure events. The sample code discussed in this Application Note was written for the Demo Code of the 71M6531, but similar principles may be used with other TERIDIAN meter ICs.

Brief Summary

Below is a brief summary of the most important hardware precautions and a sequence of events that take place when power fails.

Hardware precautions:

1. The meter hardware should be designed to store energy for a few milliseconds after mains power fails.
2. The voltage at V1 must be adjusted so that the meter chip enters reset or low-power mode before the supply voltage becomes too low for stable operation.

Sequence of events when power fails:

1. Mains power begins to fail.
2. The CE detects a sag when an expected peak falls below *SAG_THRESH*. The CE delays *SAG_CNT* samples before reporting the sag in its *STATUS* register.
3. Once, *SAG_CNT* samples are below *SAG_THRESH*, the CE sets the sag bits in its *STATUS* register.
4. The MPU permanently polls the sag bits in the CE *STATUS* register. If the sag bit is not set, nothing more related to sag processing happens.
5. If the "sag timer" is expired, the MPU saves a stable copy of the meter billing registers to EEPROM and starts a 1/3 second "sag timer". The stable copy is copied from working registers for a brief instant while interrupts are disabled. It remains valid and accessible at all other times.
6. If the "sag timer" is not expired this must mean that this is not the first sag event occurring within 1/3 of a second, and that no action should be taken.
7. The meter returns to normal operation, so it can recover if the power disturbance was only temporary.
8. If mains power does not return to normal, the internal power supply of the meter is depleted of energy. The voltage on V1 falls below the threshold.
9. The meter enters brownout mode if a battery is present in the meter. *PLL_OK* falls as the PLL fails, causing an interrupt. The PLL_OK interrupt code copes with the power failure. If a 71M6511 or 71M6513 is used in the meter, it stops operating at this point and transitions to sleep mode if a battery is present.

The Sag Scenario

When the mains voltage or voltages experience a sag event, the meter knows that power will not be available to the meter in the very near future. As opposed to a simple dip, as shown in Figure 1, a sag most likely precedes a power failure (see Figure 2).

When detecting a sag event, the meter firmware must act fast to save crucial billing data reliably before system power disappears. The meter cannot just save the billing data routinely, e.g. once per second, because that would accumulate too many write operations to the available EEPROM and consequently wear out the memory device. This means that feasible strategies are to save billing data either when power begins to fail or at longer periodic intervals.

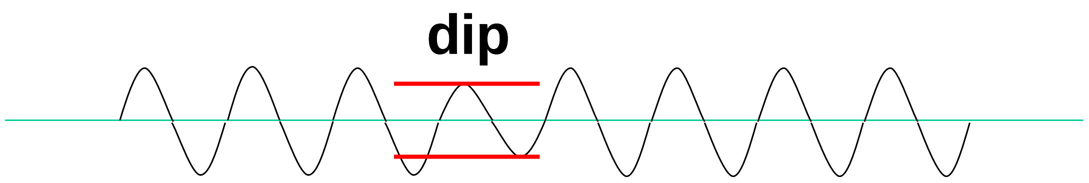

Figure 1: Dip of the Mains Voltage

After storing billing data, the meter cannot just shut down. It must be prepared for the power to return and it must be prepared for several sag events occurring within a short period of time.

What has to be done in the event of a sag event depends on the power grid, the meter power supply, whether there is a battery available in the meter, whether the metering chip has battery (low-power) modes, and, of course, on the specification of the meter.

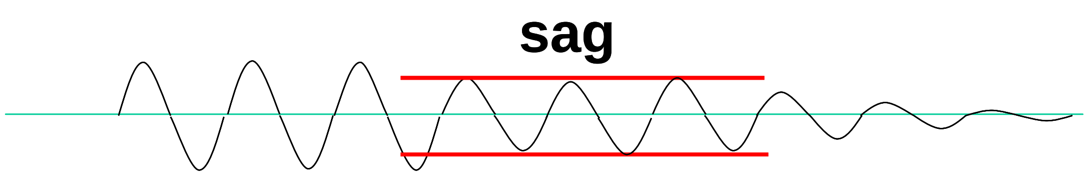

Figure 2: Sag Event of the Mains Voltage

Sag events from reclosers (self closing circuit breakers) pose a challenge to the sag event mechanism: Many meters must operate in power grids that are equipped with reclosers or other grid-switching equipment that can introduce sag-like events. Many reclosers cause five sag events in less than a second. The meter firmware must, of course, tolerate these conditions without saving erroneous data to EEPROM and without entering an unrecoverable state.

Some meters have to detect brief sags and dropouts in order to measure power quality. TERIDIAN's standard method for supplying voltage data by the CE is via the voltage square-sum registers (i.e. *V0SQSUM* and *V1SQSUM*). However, the values of *V0SQSUM* and *V1SQSUM* change only about once per second, at the start of each accumulation interval, which is not fast enough for most applications.

Detecting Sag Events

It is essential to detect a pending brownout or blackout as early as possible. The best way to do this is to monitor the mains voltage. Waiting until the DC supply voltage available on the meter board changes is not recommended. This method would require fine-tuned comparators that can detect the changes in board supply voltages with high accuracy. As opposed to the 71M6513 that has two auxiliary comparators (V2, V3), the 71M6511, 71M6521, and 71M6531 have only one built-in comparator (V1). Since this comparator will invariably shut down the IC once the comparator threshold is reached, it is not an option for the detection of a power failure.

In contrast, monitoring the mains voltage can be done without any hardware overhead using the CE (compute engine). The CE continuously checks the mains voltage against a programmable sag threshold and sets a sag bit in its *CESTATUS* register when the duration of the sag condition exceeds a programmable count (sag count).

Sag Detection for 71M651X and 71M652X

In the 71M651X and 71M652X ICs, the MPU must periodically read the *CESTATUS* register to detect a sag condition, using the CE_BUSY interrupt (`Meter\ce.c ce_busyz_isr()`). This interrupt occurs each time a sequence of ADC samples is taken (396µs), but it is useful to save MPU processing time by only reading the *CESTATUS* register every 8th sample. This introduces a worst-case delay of 7s/2520, or 2.7 milliseconds. 2.7 ms is too short to be important in most sag situations, and the 1/8 decimation reduces the sag-detection overhead to about 4% of CPU time. An example of essential signals measured with this method is shown in Figure 3.

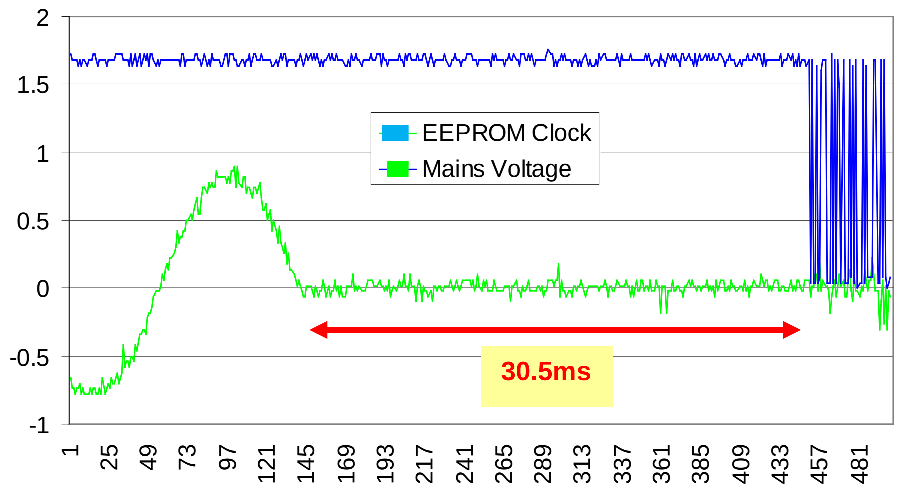

Figure 3: Sag Event at 60Hz Followed by Memory Write Operations

Sag Detection for the 71M653X

In the 71M653x family, most CE codes activate the pulse output YPULSE when a sag event is detected. The MPU can program this pin as an interrupt to detect sag without having to poll the *CESTATUS* register. This method saves MPU overhead substantially.

Other Methods

An alternative method of detecting a pending power failure is to closely monitor the frequency of the mains voltage. A lowered mains frequency is the earliest sign of a brownout from an overloaded grid. When the frequency is 1% below nominal, a power failure is very likely. Meters with ripple-control or load-shedding features can use the mains frequency to detect when to reduce the grid's load, even without a command from a utility's ripple-control transmitter.

TERIDIAN's Demo Code does not demonstrate load shedding controlled by frequency, but the CE code performs the needed frequency sensing at very high resolution (sub-milli Hertz), enabling meter developers to implement the technique.[1]

Hardware Considerations

Meter Hardware Requirements for Functioning with Sag Events and Power Failure

Any meter will be equipped with a power supply that holds charge at least for a few milliseconds after main power fails. Meters with chips that do not support low-power modes (71M6511, 71M6513) or meters with chips that are not equipped with batteries must guarantee that the billing data is reliably transferred to the storage device (usually EEPROM) before the internal DC power of the meter becomes unstable. For the 71M6521 and 71M6531, V3P3SYS must remain above 3.0V if metering is required. The digital circuits will still function at supply voltages as low as 2.6V.

Good meter design ensures that the EEPROM functions reliably during the write operation performed by the IC. Typical EEPROMs functional at supply voltages lower than 3.0VDC. It is important to dimension the resistor divider for the voltage applied to the V1 pin so that the IC is taken to reset mode before the minimum supply voltage for all essential board components is reached. This prevents writing corrupted data to memory devices. It is also essential to account for component tolerances and temperature effects on the generation of V1.

If the meter has an internal battery, the MPU of the 71M6521 or 71M6531 may operate in brownout mode as long as the EEPROM is powered by V3P3D.

The thresholds and delays of the comparators may need to be adjusted for the needs of the power utility. TERIDIAN can give general guidelines, but eventually a meter must meet the specifications of the power utility.

[1] Note: Load shedding by sensed frequency is covered by U.S. Patent No. 7149605, 2003, rights available for public license from the Pacific Northwest National Laboratory.

Cost Reduction for the Meter Power Supply

The earlier the meter detects the power failure, the sooner the billing registers will be saved. This means that the power supply is allowed to store less energy and can be less expensive.

Even if the meter has a battery, this rule still applies. To extend the battery life, the meter should use mains power to accomplish as many operations as possible.

To reduce the size of the power supply, the sag response code can also reduce the power consumption of the meter. The MPU can save several milli-amperes of supply current by disabling the CE and the ADC. After the registers are saved, both CE and ADC should be restored to service. This requires that the metering data of the CE be ignored as if it were just starting up, because the CE operation has been disrupted due to loss of power and the CE-internal PLL has lost its lock on the mains voltage.

Reducing clock rate of the MPU is not recommended. The number of cycles to save the registers will be the same at any clock speed, and thus the total energy to save the registers will be the same. Slowing the clock rate makes the sag response slower with no reduction in energy use, and thus no reduction in cost of power supply components.

A rapid response helps the code deal better with rapid sag events, such as those caused by grid-switching. TERIDIAN's demo code actually increases the MPU clock rate to the maximum to reduce the time needed to save the billing registers.

Meters that perform AMR should not change the clock rate of the MPU. Changing the clock rate unavoidably causes the loss of serial data in the time between adjusting *MPU_DIV*, and resetting the baud rate by writing to the *S0RELL* and *S0RELH* SFRs. TERIDIAN's Demo Code does not perform AMR.

Firmware for Managing Sag Events and Power Failure

Firmware designed to handle sag events should follow the structure outlined below. A flow chart for this process is given in Figure 4.

The Sequence of Events when Power Fails

1. Mains power begins to fail.
2. The CE detects a sag of the mains power when an expected peak falls below *SAG_THRESH*.
3. The CE delays *SAG_CNT* samples before reporting the sag in its *STATUS* register. *SAG_CNT* is a field in the 6521's *CE_STATE* register.
4. Once, *SAG_CNT* samples are below *SAG_THRESH*, the CE sets the sag bits in *STATUS* register.
5. The 6531's MPU is interrupted by the sag pulse on PULSE output Y, or on 6520s and earlier, it must permanently polls the sag bits in the CE *STATUS* register. If the sag bit is not set, nothing more related to sag processing happens.
6. If the "sag timer" is expired, the MPU saves two copies of the meter billing registers to EEPROM and starts a 1/3 second "sag timer".
7. If the "sag timer" is not expired this must mean that this is not the first sag event occurring within 1/3 of a second, and that no action should be taken.
8. The meter returns to normal operation, if the power disturbance was only temporary.
9. The internal power supply of the meter is depleted of energy. The voltage on *V1* falls below the threshold.
10. The 71M6531 or 71M6521 enter brownout mode if a battery is present in the meter. If a 71M6511 or 71M6513 is used in the meter, it stops operating at this point and transitions to sleep mode if a battery is present.
11. For the 71M6531 or 71M6521, *PLL_OK* falls as the PLL fails. This causes an interrupt. The interrupt code deals with the power failure.

Coping with Grid-Switching and Off-Grid Power

A sag event does not necessarily indicate a power failure, since often power returns to normal quickly after sag events, for example during grid switching. Therefore, after saving the data due to a sag event, later versions of

TERIDIAN's demonstration code return to normal operation. Code should not reset or permanently disable metering due to a sag event, because this causes loss of revenue.

In order for the code to recover, all the routines called from the interrupt that detects the sag event have to be reentrant. In particular, the EEPROM routines have to be reentrant. Further, any devices turned off to save power must be correctly turned on and restarted.

Another issue is that if the sag code attempts to save registers for each grid-switch, a partial power failure may occur after several grid switches. A write operation to the EEPROM in this situation can corrupt the data. A related issue is that the sag code may have design defects in its reentrancy, and these could be exposed by rapid reentrant execution of the code.

After a sag event occurs, the second and following sag events (that may occur in the context of grid switches) should be ignored. TERIDIAN's demo code checks the sag bits in `Meter\ce.c: ce_busyz_isr()`. This code has a timer that helps ignore the second and following sag events. The timer counts the times that the code could check the sag bits. If the timer has expired, a sag event is considered the first sag event, and the meter code saves the registers to EEPROM, because it has no way to distinguish a true power outage from the first sag associated with a grid switch.

After checking the timer after a sag event, the code restarts the sag timer. This means that after the first sag, the timer ignores further sags until 1/3 second has elapsed without a sag event. The sag timer is also started after a reset, so grid switching during power-up is also ignored.

In addition, the sag timer permits the meter to operate off the grid from a separate power supply, e.g. an external DC supply, without continually saving registers. This helps preserve the EEPROM during meter development.

Avoiding Transitions to Brownout Mode in 71M652X and 71M653X

TERIDIAN metering ICs of the second and third generation (71M652X, 71M653X) offer operation in brownout mode, i.e. operation with reduced functionality while being powered from a battery or super-capacitor. The purpose of brownout mode is to operate the display of the meter from battery power during a power failure so utility personnel can read the billing information.

TERIDIAN does not recommend saving the register data after the voltage at the V1 pin falls below VBIAS, because this event may happen too late, i.e. the power supply is already depleted of energy causing components to operate near their lower-limits and with lower reliability. It is better to save register data when a sag event on the mains voltage is detected, as shown above. However, this method requires a more robust firmware scheme to cope with transient sags on the grid (e.g. from grid switching).

In brownout mode, the IC's CE, ADC and PLL are inactive to save battery power. The clock source of the 71M6531 and 71M6521 is switched to a 28.6-kHz clock that enables the IC to send and receive serial data via the UART at 300 bd. An interrupt (*PLL_OK*) tells the firmware of the change to brownout mode.

When the battery is not present or the battery charge is too low, a simple scheme to prevent erroneous operation is to place a loop in the *PLL_OK* interrupt. This loop polls *PLL_OK,* and if the PLL has recovered because power has been restored, it performs a soft-reset to start-up the meter. If mission power is not restored (*PLL_OK* false), the code simply runs the loop until the 71M6531 or 71M6521 is forced into an involuntary sleep mode by a low supply voltage.

Below is the demo code that prevents erroneous operation when the voltage at VBAT is too low:

```
EA = 0;
CONFIG0 = 0;    // Make sure we are running as fast as we can.
IFLAGS = (~IE_PLLFALL_) & (~IE_PLLRISE_);  // force an edge to occur
// if entering brownout mode, and there's no battery, hang
while(!BATTERY_MODE_ENABLE && IN_BROWNOUT_MODE)
    RESET_WD(); // wait for the power to die, OK due to slow MPU.
main_soft_reset ();
```

The code first disables interrupts, because no other operation should be permitted. It sets *MPU_DIV* in *CONFIG0* to zero so that the brownout code will run at the full 28-kHz clock rate. It clears the interrupt flags so that the interrupt can run again. In the Demo Code, `BATTERY_MODE_ENABLE` reflects just the status of an external jumper connected to a DIO pin. In a production meter, `BATTERY_MODE_ENABLE` should be false when the battery has failed. `IN_BROWNOUT_MODE` just tests the *PLL_OK* bit to test if the meter chip is in brownout mode.

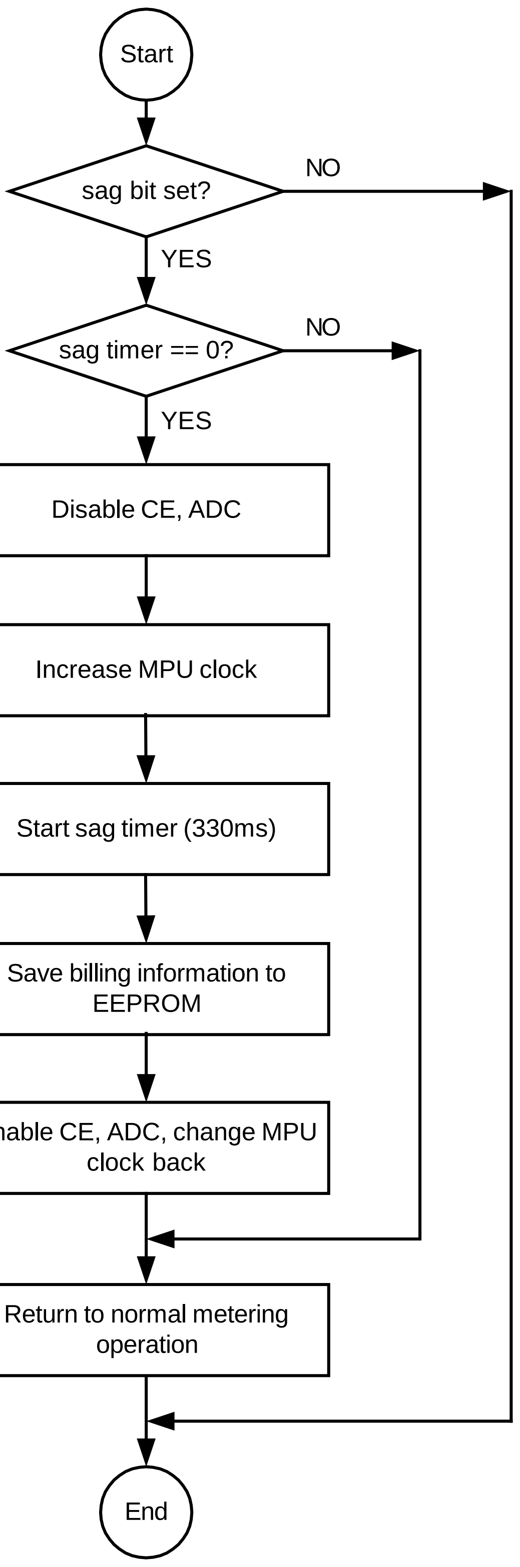
Start
sag bit set?
NO
YES
sag timer == 0?
NO
YES
Disable CE, ADC
Increase MPU clock
Start sag timer (330ms)
Save billing information to EEPROM
Enable CE, ADC, change MPU clock back
Return to normal metering operation
End

Figure 4: Sag Event Flow Chart

While it is waiting for power to fail, the loop resets the watchdog because a watchdog reset would cause the meter to try to start-up again. A start-up usually tries to read the EEPROM, and could cause corruption of data.

`main_soft_reset ()`is the way the demo code starts all of its power mode transitions. The reset path initializes all of the meter's states, and it would be wasteful to duplicate this code for transitions out of sleep and LCD-only mode.

Jumping to the zero code address is not enough code to emulate `main_soft_reset ()`. The soft reset code should also perform `RTI` four times to clear waiting interrupts, and if the start-up code does not initialize every bit, the reset routine should set default values.

Meters that contain batteries should detect battery failure and avoid reading from or writing to the EEPROM in low power modes after the battery has failed. Batteries can be tested in mission mode by setting *BME_ENABLE* and comparing the value for VBAT provided by the CE against a calibrated limit value. The testing uses a small amount of energy, but testing once a day preserves battery life well.

Avoiding Data Loss while Register Data is Computed

On each accumulation interval, TERIDIAN's demonstration code prepares two copies of billing register data. (`Meter\meter.h`: `Totals.Acc, Totals.AccB`). There are two copies, because the firmware has to cope with a power failure immediately. The calculations use copy A. At the end of the calculations, copy A has valid values and a correct check-sum. At this point, in the 71M6531's demo, the code briefly disables interrupts, and quickly copies the values from A to copy B.

The register save operation always uses copy B. Copy B is valid except for the fifty microseconds that the copy takes. This delay does not significantly affect the response time of the sag interrupt.

Avoiding Data Loss from Timing Issues

The firmware should arrange to have a valid, savable set of billing registers in RAM before it begins to test for a sag condition. This data set should have pre-computed checksums, etc. In the above scheme (see Avoiding Data Loss while Register Data is Computed), copy B should be set up from stored register data before sag detection begins.

In addition, the save operation should exit if it is started while another save is in progress. This situation can happen occasionally, for example, if a timed save operation is interrupted by a power-failure detection interrupt.

A basic test for timing-related errors is to cycle the AC power to the meter while sweeping the cycle time from about 0.1 seconds to more than the meter's nominal start-up time. If several thousand cycles complete without noticeable data loss, the designer can conclude that there are no issues with saving and restoring of billing data.

Avoiding Data Loss due to Unplanned Errors

The meter firmware should save the billing registers periodically, just in case some complex series of power-failure events causes the dynamic save to fail. However, saving the registers too often will use up all the write cycles to the EEPROM.

The code below saves the power registers in four different EEPROM pages. The saving scheme is diverse in time and space. Each copy is saved twice as often as the next. Each time the registers are saved, only one copy is saved. The other copies remain intact. In that way, if one register set is corrupted, three other copies remain available.

With an achievable error rate of 0.1% failures per copy ($1x10^{-3}$), four copies increase the theoretical reliability to $1x10^{-12}$, a very practical value for most electronic devices. With a conservative error rate of 1%, six copies can achieve the same reliability.

With a one million write operations to the EEPROM, as most EEPROMS permit, the code below can save the registers every half hour, and the limit for write operations will not be exceeded in 50 years.

The restore operation picks the register set with a valid longitudinal redundancy check (LRC) and the largest Whr. LRCs, also called longitudinal parity, "exclusive-or" all the bytes together. They are superior to checksums because all the bits have equal significance and are also easy and fast to implement.

Almost all of the time, picking the register with the largest Whr value preserves the largest valid revenue. When the Whr register turns over (from 999999 to 000000), the register selection can pick an old Whr value. However, if periodic write operations occur every half hour, all the register copies are rewritten within eight hours, and this problem fixes itself.

```
// Several copies of power registers in EEPROM
#define REG_SET_SIZE \
    (((sizeof(struct Accumulators_t) + PAGE_SIZE - 1) / PAGE_SIZE) * PAGE_SIZE)
int16r_t reg_set_adr[EEPROM_REG_SET_CNT] = {
    (EEPROM_REGISTERS + 0 * REG_SET_SIZE),
    (EEPROM_REGISTERS + 1 * REG_SET_SIZE),
    (EEPROM_REGISTERS + 2 * REG_SET_SIZE),
    (EEPROM_REGISTERS + 3 * REG_SET_SIZE)
};

#pragma save
#pragma NOAREGS
void meter_save (void) small reentrant
{
    int8_t reg_idx;
    IRQ_DEFINES;

    // prevent reentrant calling of this routine
    IRQ_DISABLE();
    if (save_semaphore)
    {
        IRQ_ENABLE();
        return;
    }
    save_semaphore = 1; // prevent saves
    IRQ_ENABLE();

    // select an EEPROM slot for saving.
    // Save at different times, and at different spaces.
    // As save_count counts, the sequence is 010201030102010 repeat
    ++save_cnt; // nonvolatile value in Totals.Acc
    if ((save_cnt & 7) == 7)
        reg_idx = 3; // register set 3
    else if ((save_cnt & 3) == 3)
        reg_idx = 2; // register set 2
    else if ((save_cnt & 1) == 1)
        reg_idx = 1; // register set 1
    else
        reg_idx = 0; // register set 0

    // save the best available data
    memcpy_prx (
            reg_set_adr[reg_idx], // address to store to
            (uint8x_t *)&Totals.AccB, // The start of the data
            sizeof(struct Accumulators_t));

    save_semaphore = 0; // permit saves
}
```

To recover from unexpected, rare corruptions such as strikes from ionizing particles, EFT, etc., meters may also reset themselves periodically. To use this method, a meter should measure the power rate before and after the reset, and interpolate the use between. A commonly used reset period is once per day. A periodic reset also sets a natural limit to the firmware test interval: The firmware never runs longer than the time between resets, so it never needs to be tested for a longer time.

Avoiding Data Loss due to Write Limits of EEPROM or FLASH

Some commercially available EEPROMs fail to accept data after as few as 10,000 write cycles. To simplify firmware, TERIDIAN recommends selecting an EEPROM that can accept 1,000,000 or more write cycles.

TERIDIAN's 71M6521B Demo Code demonstrates the use of the internal flash memory to save register data. The limit for write operations is specified as only 20,000. In order not to exceed this limit over the life time of the meter, the write operations must be distributed over many cells. Every read of the saved area should be of data that is in a known state. The methods shown below may also be applied to EEPROMs with low limits of write operations.

As a first step, the size of the save area should be designed. The number of register save operations per year should be estimated. Most meters require either a 30 or 50-year life. The save area should be large enough so that the required number of write operations can be performed with 20% margin.

The normal scheme is that at power-up, the code searches the save area backwards from a high memory address to a low memory address. It searches for the last empty byte in the save area. The high-address end of the save area will be erased, and therefore have a value of 0xFF.

From the location of the empty byte, the code steps backwards by a fixed length to find and restore the last saved set of registers. If both copies have errors, detected by a checksum scheme, it can step backwards again until it finds a valid register set. This is a simple version of a "journaling flash file system."

If there is room to save the registers again, the last empty address is saved for the next time the code must store the registers.

If the save area is full, the following process is used:

1) The first flash page is erased.
2) The billing registers are written once to the beginning of the first page in order to create one safe copy.
3) The rest of the pages are erased from front to back.
4) After the erasure, the code should test the erase by searching for the last empty address.

With this scheme, most meters maintain an erase counter in the registers. When the meter is within 20% of the maximum write cycles, it reports an error so the utility can predict and avoid write failures of the meter.

Sag Detection Implemented with the TERIDIAN Demo Code

TERIDIAN's Demo Code detects a sag event when the peak voltage falls below 80V, because this number is easy to demonstrate in many test environments. A production meter can often have a higher sag limit. Failing at higher voltages detects the sag event earlier, which permits a less-expensive power supply.

TERIDIAN's Demo Code waits for 80 samples before setting the sag bit. All voltage samples have to be below the sag threshold, otherwise the sag bit is reset. Since each sample is typically 1/2520.6 of a second, the sampling interval amounts to 32 ms, about two cycles of a 60 Hz signal. This relatively short time helps to compensate for the low sag threshold. Many production meters wait up to 100 ms (six cycles of 60 Hz, five cycles of 50Hz). To set the meter to a 100 ms sag response, *SAG_CNT* should be set to 252 decimal. For a production meter, a higher, more realistic sag threshold is recommended, as well.

TERIDIAN's Demo Boards have single-phase power supplies operating from phase A. TERIDIAN's Demo Code saves registers only when phase A sags. However, meters powered from other phases should sense sag events for those phases.

Poly-phase powered meters should sense all the phases from which they receive power, and fail only when all the phases sag.

TERIDIAN's meter Demo Code checks the phases using a bitwise "and" with the CE's *STATUS* register. To change TERIDIAN's Demo Code from single-phase sag detection to multiphase sag detection, `options.h` has a bit-mask constant called `POWERED_PHASE`. Change this constant so that it has a bit set for every phase that can feed the meter's power supply. Then, all the phases covered by the bit-mask will have to sag before a register save will occur.

Revision History

Revision	Date	Description
Rev. 1.0	N/A	Not published.
Rev. 1.1	7/20/2007	First publication.
Rev. 1.2	12/03/2008	Added methods for securing data save and restore operations using multiple copies. Added description of advanced sag detection mechanisms in 653X ICs. Updated Teridian street address, document format and document name.