Implementing Fast Telemetry with Power System Management Controllers

Michael Jones

INTRODUCTION

The second-generation Power System Management (PSM) Controllers, such as the LTC®3887, introduce new features for fast telemetry[1]. Fast telemetry allows firmware to focus on a single measurement, or a fixed set of common voltage/current measurements to improve telemetry bandwidth up to 16 times. Two new commands allow control of the telemetry multiplexer (MFR_ADC_CONTROL) and time alignment (MFR_ADC_TELEMETRY_STATUS). The multiplexer can be set to a single or smaller set of measurements, and the status enables polling for a "measurement complete". When combined, not only is the throughput improved, but there are no duplicate measurements: all obtained values are unique ADC measurements.

FIRST-GENERATION ADC MULTIPLEXER

The LTC38XX family prior to the LTC3887 (LTC3880 and LTC3883) uses a multiplexer and a round-robin approach to measurements. Measurements are always made in the same order, with a total loop time of approximately 100ms.

From a firmware perspective, any given measurement is updated once each 100ms, and there is no way to know how "fresh" the current returned value is. Any given measurement could be up to 100ms old, and sampling faster than every 100ms just returns the same data.

SECOND-GENERATION ADC MULTIPLEXER AND STATUS

The LTC3887 and newer devices upgraded the round-robin approach with behavioral choices. The MFR_ADC_CONTROL command selects 1 of 10 telemetry modes, or selections as shown in Table 1.

Table 1. Telemetry Modes

COMMANDED VALUE	TELEMETRY SELECTED
0x0E – 0xFF	Reserved
0x0D	ADC Short Round-Robin
0x0C	Channel 1 External Temperature
0x0B	Reserved
0x0A	Channel 1 I_{OUT}
0x09	Channel 1 V_{OUT}
0x08	Channel 0 External Temperature
0x07	Reserved
0x06	Channel 0 I_{OUT}
0x05	Channel 0 V_{OUT}
0x04	Internal IC Temperature
0x03	Reserved
0x02	Reserved
0x01	V_{IN}
0x00	Standard ADC Round-Robin Telemetry

Value 0x00 behaves exactly the same as first-generation devices. The value 0x0D enables a round-robin function that includes four measurements (V_{OUT0}, I_{OUT0}, V_{OUT1}, I_{OUT1}). The remaining values enable a single measurement.

The MFR_ADC_CONTROL is not stored in EEPROM, so firmware must send the selection to the device to enable a mode other than the default of round-robin (0x00). It is recommended that controllers run in round-robin ADC mode most of the time to assure that ADC-enabled faults such as temperature and VIN_OV are operational.

Note 1. LTC3884, LTC3886, LTC3887, LTM®4676A, LTM4675, LTM4677 all support fast telemetry.

The second-generation devices also add measurement status capability that enables polling for a change in measurement value. Table 2 shows that there are status bits for the four critical measurements using command MFR_ADC_TELEMETRY_STATUS. Each bit transitions from 0 to 1 immediately after the corresponding measurement is performed. Writing a one to a bit clears it.

Table 2. Telemetry Status

BIT	TELEMETRY DATA AVAILABLE
7	Reserved Returns 0
6	Reserved Returns 0
5	Reserved Returns 0
4	Reserved Returns 0
3	Channel 1 I_{OUT} Readback (I_{OUT1})
2	Channel 1 V_{OUT} Readback (V_{OUT1})
3	Channel 0 I_{OUT} Readback (I_{OUT0})
1	Channel 0 V_{OUT} Readback (V_{OUT0})

When switching from 0x0D, short telemetry loop, to another mode, the firmware must first switch to 0x00, round-robin, for at least 120ms, before switching to the new mode, to ensure well behaved status. Switching between all other modes does not require switching to 0x00 first.

BACKWARDS COMPATIBILITY

If the new commands are not used, the second-generation devices behave like the first-generation devices and no firmware changes are required.

IMPACT ON SUPERVISION AND VOLTAGE SERVO

First-generation devices rely on the multiplexer and ADC for three non-telemetry functions: temperature supervision, V_{IN} overvoltage supervision and voltage servo.

Temperature changes slowly in a system: typical latency in a PSM controller ADC is sufficiently small that temperature supervision is easily managed with the ADC. V_{IN} also moves slowly in most PSM applications allowing the ADC to adequately supervise the input voltage.

All PSM controllers rely on the ADC to achieve 0.5% total DC accuracy. The second-generation devices can only provide this feature for multiplexer selection 0x00.

In both cases, the only way to ensure that all temperature and V_{IN} supervision and all V_{OUT} accuracies are achieved is to use multiplexer selection 0x00, or to design firmware to ensure that selection 0x00 is periodically enabled for long enough time to allow the ADC to make the measurements required. This means selecting 0x00 for at least 120ms to allow a full round-robin telemetry loop. The time between selections that interfere with these measurements will determine the response time of the feature.

FIRMWARE DESIGN

Second-generation application firmware can be evaluated using Linduino® PSM and a demo board.

The Linduino PSM API contains an SMBus and PMBus interface to the LTC38XX, LTM46XX and LTC29XX families.

The best place to start is to look at code for selection 0x0D, the short loop. We will use LTC3887 for the following discussion.

Code Listing 1 Shows a Short Telemetry Loop. Line 1 puts the LTC3887 into the "ADC Short Round-Robin" mode using value 0x0D, and command MFR_ADC_CONTROL. This command is an SMBus Write Byte.

Line 2 clears the status using value 0x0F and command MFR_ADC_TELEMETRY_STATUS.

At this point, the code can monitor the status register using polling until it notices a change to V_{OUT0}. Line 6 sets the PAGE to 0, corresponding to V_{OUT0}. Line 7 enters a polling loop on bit one. When this loop completes, it indicates that V_{OUT0} was just measured. Line 9 reads the value of V_{OUT0} and stores it in an array.

The short telemetry loop measures in this order: V_{OUT0}, I_{OUT0}, V_{OUT1}, I_{OUT1}.

The following lines loop on the status of each of these measurements. At the end, on line 30, the MFR_ADC_TELEMETRY_STATUS is cleared in preparation for the loop to wrap around and continue.

When all loops are complete, line 32 sets the telemetry back to the standard mode, and line 33 delays 120ms to ensure there is at least one full round-robin, thus allowing the temperature supervisor and voltage servo to update.

Other code in this listing measures the time between measurements.

Table 3 shows the results. On the first line, notice the time value of 33296µs. When the telemetry mode is changed to the short round-robin, the current measurement in progress must complete before the first short round-robin measurement can be made. Therefore, this number represents a latency of getting into the short round-robin mode.

The times between telemetry loops are approximately 28ms, a 3× to 4× improvement over the older generation. The time measurement resolution will be the time it takes to query the PMBus for the status, which depends on the clock rate of the PMBus master.

Table 3.

Δt	V_{OUT0}	Δt	I_{OUT0}	Δt	V_{OUT1}	Δt	I_{OUT0}
33296µs	5.00	6704µs	0.21	5816µs	2.00	6812µs	0.08
6164µs	5.00	6712µs	0.21	6268µs	2.00	6808µs	0.08
6156µs	5.00	6704µs	0.21	6264µs	2.00	6808µs	0.08
6164µs	5.00	6712µs	0.21	6256µs	2.00	6372µs	0.08
6156µs	5.00	6712µs	0.20	6268µs	2.00	6808µs	0.09
6156µs	5.00	6708µs	0.21	6260µs	2.00	6808µs	0.08
6164µs	5.00	6700µs	0.21	6264µs	2.00	6812µs	0.08
6164µs	5.00	6712µs	0.21	5824µs	2.00	6808µs	0.08
6156µs	5.00	6704µs	0.21	6264µs	2.00	6804µs	0.09
6168µs	5.00	6700µs	0.21	6264µs	2.00	6820µs	0.08

Listing 1: Short Loop

```
 1  smbus->writeByte(ltc3880_i2c_address, MFR_ADC_CONTROL, 0x0D); // Fast
 2  smbus->writeByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS, 0x0F); // Clear status
 3  start_time = micros();
 4  for (i = 0; i < NUM_MEAS; i++)
 5  {
 6  pmbus->setPage(ltc3880_i2c_address, 0);
 7  while ((smbus->readByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS) & 0x01) == 0);
 8  current_time = micros();
 9  voltage0[i] = pmbus->readVout(ltc3880_i2c_address, false);
10  voltage0_time[i] = current_time > start_time ? current_time - start_time :
        max_time - start_time + current_time;
11  start_time = current_time;
12  smbus->writeByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS, 0x0E); // Clear remain-
ing status
13  while ((smbus->readByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS) & 0x02) == 0);
14  current_time = micros();
15  current0[i] = pmbus->readIout(ltc3880_i2c_address, false);
16  current0_time[i] = current_time > start_time ? current_time - start_time :
        max_time - start_time + current_time;
17  start_time = current_time;
18  pmbus->setPage(ltc3880_i2c_address, 1);
19  while ((smbus->readByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS) & 0x04) == 0);
20  current_time = micros();
21  voltage1[i] = pmbus->readVout(ltc3880_i2c_address, false);
22  voltage1_time[i] = current_time > start_time ? current_time - start_time :
        max_time - start_time + current_time;
23  start_time = current_time;
24  while ((smbus->readByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS) & 0x08) == 0);
25  current_time = micros();
26  current1[i] = pmbus->readIout(ltc3880_i2c_address, false);
27  current1_time[i] = current_time > start_time ? current_time - start_time :
        max_time - start_time + current_time;
28  start_time = current_time;
29
30  smbus->writeByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS, 0x0F); // Clear status
31  }
32  smbus->writeByte(ltc3880_i2c_address, MFR_ADC_CONTROL, 0x00); // Standard
33  delay(120);
```

Listing 2: Single Measurement

```
 1  smbus->writeByte(ltc3880_i2c_address, MFR_ADC_CONTROL, 0x06); // Iout0
 2  pmbus->setPage(ltc3880_i2c_address, 0);
 3
 4  smbus->writeByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS, 0x0F); // Clear status
 5  start_time = micros();
 6  for (i = 0; i < NUM_MEAS; i++)
 7  {
 8  while ((smbus->readByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS) & 0x02) == 0);
 9  current_time = micros();
10  current0[i] = pmbus->readIout(ltc3880_i2c_address, false);
11  current0_time[i] = current_time > start_time ? current_time - start_time :
        max_time - start_time + current_time;
12  start_time = current_time;
13  smbus->writeByte(ltc3880_i2c_address, ADC_MFR_TELEMETRY_STATUS, 0x0F);
14  }
15  smbus->writeByte(ltc3880_i2c_address, MFR_ADC_CONTROL, 0x00); // Standard
16  delay(120);
```

To get another 4× improvement, firmware can concentrate on a single measurement. Listing 2 demonstrates a fast measurement of output current. Table 4 shows the results.

Table 4

Δt	I_{OUT0}
7564µs	0.21
6704µs	0.21
6264µs	0.21
6272µs	0.20
6268µs	0.21
6272µs	0.21
6700µs	0.21
6264µs	0.21
6260µs	0.21
6268µs	0.21

Notice the initial delay of 7564µs. This is the same variable delay as the short round-robin mode. The delay between measurements is 6.2ms, an improvement of 16×!

HOW TO SOLVE PROBLEMS WITH FAST TELEMETRY

The most typical usages of fast telemetry are characterization and system debugging. During system characterization, systems engineers want an accurate record of voltage and current along with other system measurements such as computational load.

During debug, occasionally a system has a start-up issue or other complex interaction and fast telemetry can gather more data.

In the above cases, the firmware, or even an external USB-based PMBus master, may keep the device in short loop mode and neglect the temperature and servo. This might be OK during debug, but is dangerous during system deployment. For deployment, temperature and V_{IN} measurements protect the design, and output voltage measurements ensure 0.5% accuracy.

HANDLING MULTIPLE DEVICES

If data from more than one device is required in fast mode, the firmware or external controller will have to interleave polling of all devices of concern. However, it is possible to use a timer to learn the time delays between different devices and poll them just before the expected measurement. This will make the timing of the measurements a little more deterministic, and in multi threaded applications, allow the microcontroller to service other tasks.

SUMMARY

The second-generation Power System Management Controllers have new telemetry features that allow up to 16× improvement in throughput. By focusing the multiplexer and ADC on a smaller set of measurements, the telemetry loop runs faster. Polling allows firmware or an external PMBus master to determine the time that the ADC makes a measurement assuring captured data has no duplicate values. Most sophisticated polling can capture data from multiple devices in a time-correlated fashion. A little care must be used if fast telemetry is used on production units so that the temperature V_{IN} supervisors and voltage servos have adequate time to protect the system and ensure its accuracy.

LT 0118 • PRINTED IN USA