APRIL 2005

Controlling the CHOP_EN Signal

TERIDIAN Semiconductor's 71M651x Energy Metering family of chips offers a chopper-stabilized ADC reference that effectively eliminates drift over temperature and time. This application note describes proper control of the chopping signal through the MPU firmware.

Chopping Basics

In order to eliminate offset voltages that could affect measurement accuracy, the amplifier within the reference is auto-zeroed by means of an internal signal that is controlled by the *CHOP_EN* bits. When this signal is HIGH, the connection of the amplifier inputs is reversed. This preserves the overall polarity of the amplifier gain but inverts the input offset. By alternately reversing the connection, the offset of the amplifier is averaged to zero. The two bits of the *CHOP_EN* register have the function specified in Table 1.

CHOP_EN[1]	CHOP_EN[0]	Function
0	0	Toggle chop signal
0	1	Reference connection positive
1	0	Reference connection reversed
1	1	Toggle chop signal

Table 1: *CHOP_EN* Bits

Operation with Automatic Chopping

For automatic chopping, the *CHOP_EN* bits are set to either 00 or 11. In this mode, the polarity of the signals feeding the reference amplifier will be automatically toggled for each multiplexer cycle as shown in Figure 1. With an even number of multiplexer cycles in each accumulation interval, the number of cycles with positive reference connection will equal the number of cycles with reversed connection, and the offset for each sampled signal will be averaged to zero. This sequence is acceptable when only the primary signals (meter voltage, meter current) are of interest.

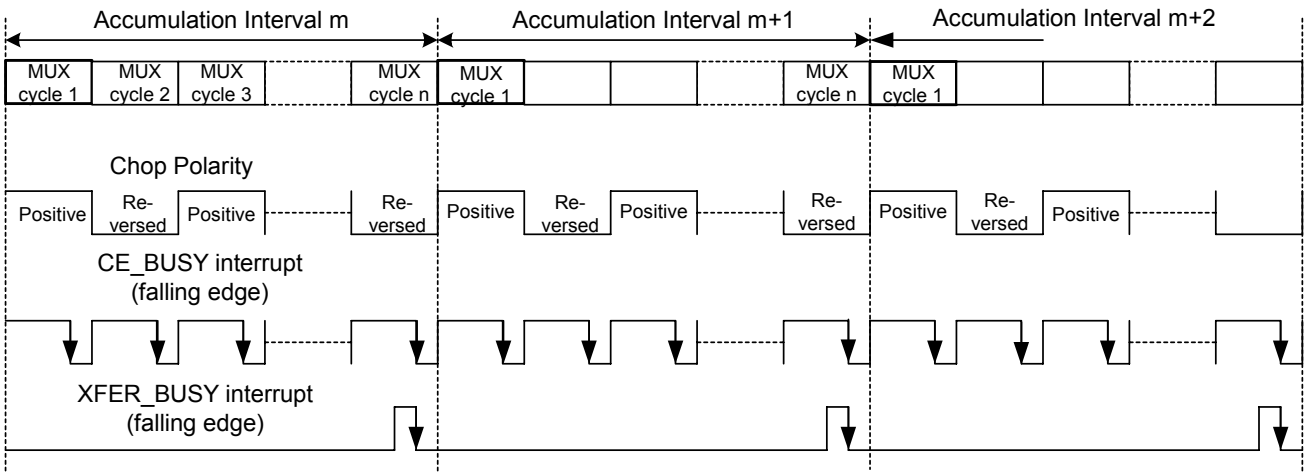

Figure 1: Chop Polarity w/ Automatic Chopping

If temperature compensation or accurate reading of the die temperature is required, alternate multiplexer cycles have to be inserted in between the regular cycles. This is done under MPU firmware control by asserting the *MUX_ALT* bit whenever necessary. Since die temperature usually changes very slowly, alternate multiplexer cycles have to be inserted very infrequently. Usually, an alternate multiplexer cycle is inserted once for every accumulation period, i.e. after each XFER_BUSY interrupt. This sequence is shown in Figure 2.

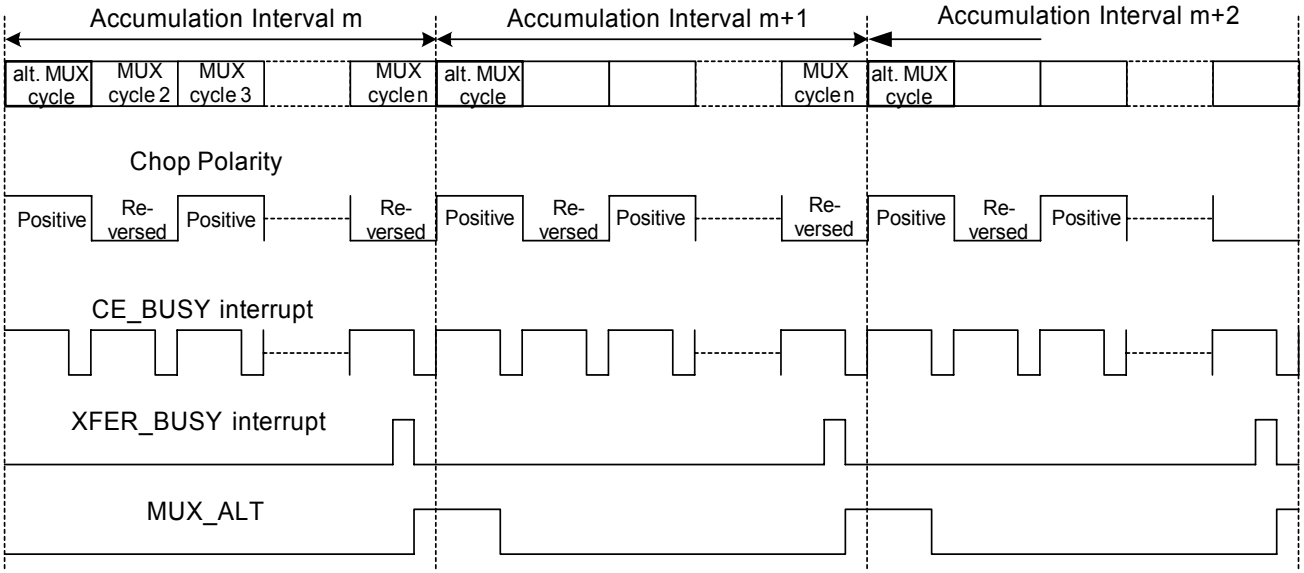

Figure 2: Sequence with Alternative Multiplexer Cycles

This sequence has one major disadvantage: The alternative multiplexer cycle is always operated with positive connection. Consequently, any offset will appear on the temperature measurement, which will decrease the accuracy of this measurement and thus cause temperature reading and compensation to be less accurate.

Operation with Controlled Chopping

The sequence in Figure 3 uses the *CHOP_EN* bits to control the chopper polarity after each XFER_BUSY interrupt. *CHOP_EN* is controlled to alternate between 10 (positive) or 01 (reversed) for the first multiplexer cycle following each XFER_BUSY interrupt. After these first two cycles *CHOP_EN* returns to 11 (automatic toggle). The value of *CHOP_EN*, when set after the XFER_BUSY interrupt, is in force for the entire following multiplexer cycle.

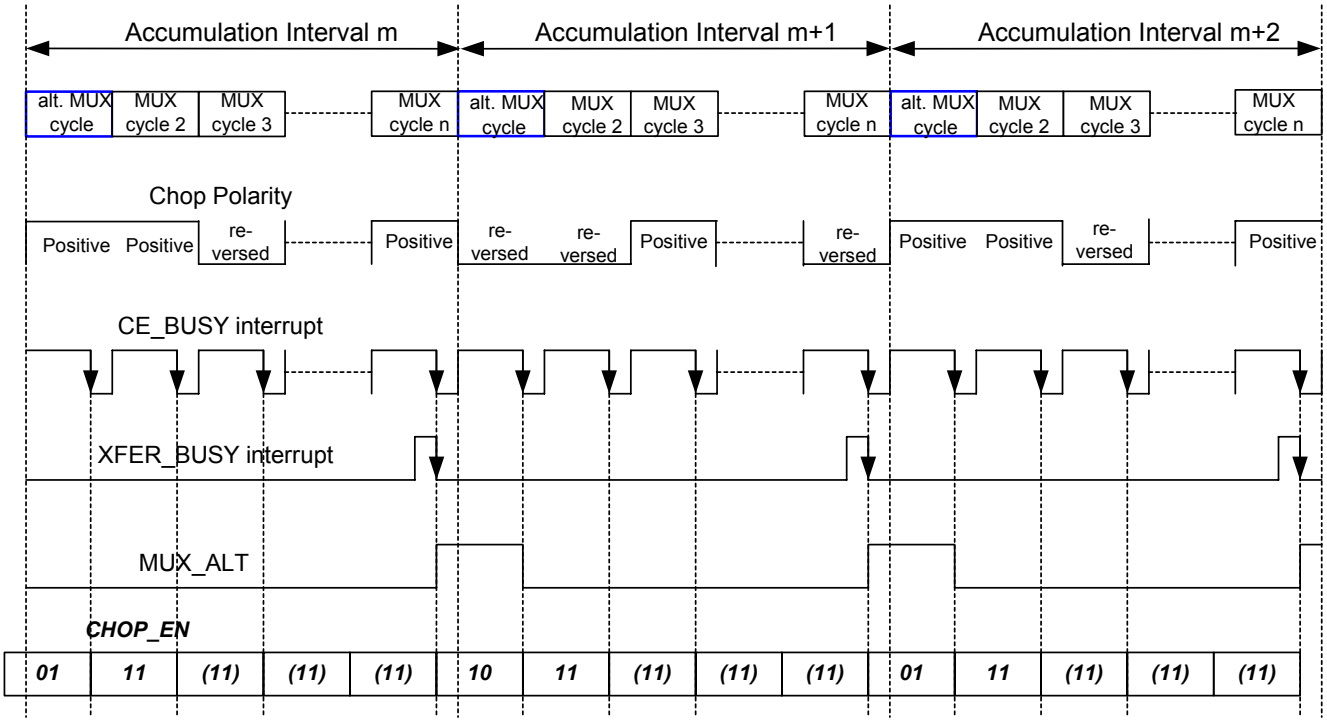

Figure 3: Sequence with Alternative Multiplexer Cycles and Controlled Chopping

When using this sequence, the alternative multiplexer cycle is toggled between positive and reversed connection resulting in accurate temperature measurement.

An example for proper application of the *CHOP_EN* bits can be found in the Demo Code shipped with the 6511 and 6513 Demo Kits. Firmware implementations should closely follow this example.

Firmware Supporting Controlled Chopping

The following code sections extracted from the Demo Code are interrupt service routines. They demonstrate how the *CHOP_EN* bits are controlled.

Definitions:

```
#define  CHOP_EN 0x30       // CHOP_EN is in bits 5-4 at I/O address 0x2002
#define  CE2 IO.Configuration.I_CE2
static U08 chop = _POSITIVE;
enum CHOP { _POSITIVE = 0x10, _REVERSED = 0x20,  _ENABLED = 0x30};
```

On XFER_BUSY Interrupt:

This interrupt occurs after each accumulation cycle, e.g. once per second:

```
void xfer_busy_isr (void) small reentrant
{
    EA = 0; // disable other interrupts for the following critical section

    // alternate polarity to chop temperature, which is read once per sumcycle
    CE2 = (CE2 & ~CHOP_EN) | chop; // Use 01 or 10 state of chop configuration
                                   // for the next 397 microseconds
    chop ^= CHOP_EN;               // Toggle chop between 1 (01b) and 2 (10b) to
                                   // alter the CE2 configuration on the next
                                   // XFER_BUSY interrupt.

    // enable next CE busy to restore hardware chopping
    IEX_CE_BUSY = 0;
    EX_CE_BUSY = 1;

    if (meter_alt)
       mux_alt ();     // Do Alternate MUX cycle.

    EA = 1;  // enable other interrupts, the critical section is ended
    ……..

       }
```

On CE_BUSY Interrupt:

This interrupt occurs after each multiplexer cycle, i.e. once every 397µs. It occurs synchronous with the XFER_BUSY interrupt:

```
void ce_busy_isr (void) small reentrant
{
    CE2 = (CE2 & ~CHOP_EN) | CHOP_EN;   //Switch to the Automatic mode for Chop
                                        // configuration

……..
}
```

TERIDIAN Semiconductor Corp., 6440 Oak Canyon Rd., Irvine, CA 92618
TEL (714) 508-8800, FAX (714) 508-8877, http://www.teridian.com

