MSP430 Advanced Technical Conference 2006


# Implementing a Temperature Compensated RTC


Steve Underwood
MSP430 FAE Asia
Texas Instruments

# Agenda

- **Building a simple RTC with the MSP430**
- **What limits the accuracy of the simple RTC?**
- **What can we do to improve accuracy?**
- **An example of using correction techniques**

# A simple real time clock

- **Simple MSP430 software RTCs are popular**
- **The 32kHz crystal osc is a suitable source of timing**
- **Several timers can generate 1 interrupt/second from the 32kHz oscillator**
  - Timer A, Timer B, Basic Timer, Watchdog timer
- **Each second, the CPU can wake very quickly (<6us) from LPM3, run at full speed to update time and date variables in RAM, and go back to sleep**
- **Typically an application would also do some housekeeping in the per second interrupt routine**
  - Total system consumption can be <1.2uA
  - A 3022 coin cell battery can run many products for >10 years
- **This is a classic MSP430 application**

# A simple real time clock - tradeoffs

- **MSP430x4xx users usually use the basic timer**
  - Part of LCD controller support
  - A very simple timer for just this type of application
- **Users of other devices use Timer A/B or WDT**
  - The watchdog timer can be used, if watchdog functionality is not required
- **Care is needed, if WDT is performing its watchdog function**
  - The longest watchdog timeout is 1s, but the interrupt period is also 1s
  - Kicking the WDT at both the start and the end of the interrupt service can avoid false expiry of the WDT

# Real time clocks – the problem

- **Many appliances, such as water and electricity meters, operate over a wide temperature range**
- **Watch crystal frequencies vary significantly over this range**
  - You don't cook or freeze your wrist, so the good timekeeping of a watch is no indicator of performance
- **Utilities are demanding precision real time clocks**
  - This demands temperature compensation
- **Precision real time clocks usually need to be ultra low power**
  - They need to run for years from a small battery, for cost or size reasons

# Agenda

- Building a simple RTC with the MSP430
- What limits the accuracy of the simple RTC?
- What can we do to improve accuracy?
- An example of using correction techniques

# Factors affecting clock accuracy

- **Manufacturing tolerance of the crystal**
  - Crystals better than 10ppm at room temp. can be expensive
  - 100ppm crystals are cheaper, if we can tolerate them
- **Incorrect loading of the crystal**
  - Crystals can be pulled off frequency by the loading applied to them (think VCXO)
  - A very poorly loaded crystal might be unstable, especially in the presence of high EMI
- **Handling can change the crystal**
  - Thermal shock during soldering can alter a crystal's frequency
- **Temperature changes the crystal**
  - Over a wide temperature range, this is the biggest factor affecting clock accuracy
- **Crystals change with age**
  - Several ppm per year for typical 32kHz crystals

# Real time clocks – the solution

- **Every ADC used in the MSP430 family has an internal temperature sensor we can use for RTC compensation purposes**
  - Requires calibration as the product is tested
  - Quick, simple calibration is adequate for most uses
  - Devices without a true ADC might use Comp_A to form a slope ADC, and sense temperature with an external device
- **Low precision crystals are cheaper than high precision, and may have just as good an aging characteristic**
  - Manufacturing tolerance error can be calibrated away as the product is tested
  - Calibration can be fast, cheap and simple
  - Calibration is best delayed as long as possible, so stresses relax

# A typical crystal (Microtune)

| Package Size | | DS26 | DS15 | DS10 | |
|---|---|---|---|---|---|
| Nominal frequency | $F_L$ | 32.768 | 32.768 | 32.768 | kHz |
| Load capacitance [1)] | $C_L$ | 8.2 | 8.2 | 8.2 | pF |
| Frequency tolerance [2)] | $\Delta F/F$ | +/-20 | +/-20 | +/-20 | ppm |
| | $\Delta F/F$ | +/-30 | +/-30 | +/-30 | ppm |
| | $\Delta F/F$ | +/-100 | +/-100 | +/-100 | ppm |
| Series resistance typ./max. | $R_S$ | 30 / 42 | 35 / 50 | 45 / 60 | kΩ |
| Motional capacitance typ. | $C_1$ | 2.1 | 2.1 | 2.1 | fF |
| Static capacitance typ. | $C_0$ | 0.9 | 0.9 | 0.9 | pF |
| Drive level max. | P | 1.0 | 1.0 | 1.0 | µW |
| Quality factor min. | Q | 55'000 | 45'000 | 38'000 | |
| Insulation resistance min. | $R_i$ | 500 | 500 | 500 | MΩ |
| Aging first year max. | $\Delta F/F$ | +/-3 | +/-3 | +/-3 | ppm |
| Turnover temperature | $T_0$ | 25 +/-5 | 25 +/-5 | 25 +/-5 | °C |
| Frequency vs. temperature | $\Delta F/F_0$ | $-0.035\ ^{ppm}/_{°C^2}\,(T - T_0)^2$ +/-10% | | | ppm |

**Data sheet for a typical 32kHz crystal. Note:**

- The crystal frequency peaks close to 25C
- Frequency falls in a parabolic manner above and below 25C
- The parabolic curve varies little between samples of crystal
- Aging is the same for low and high precision versions

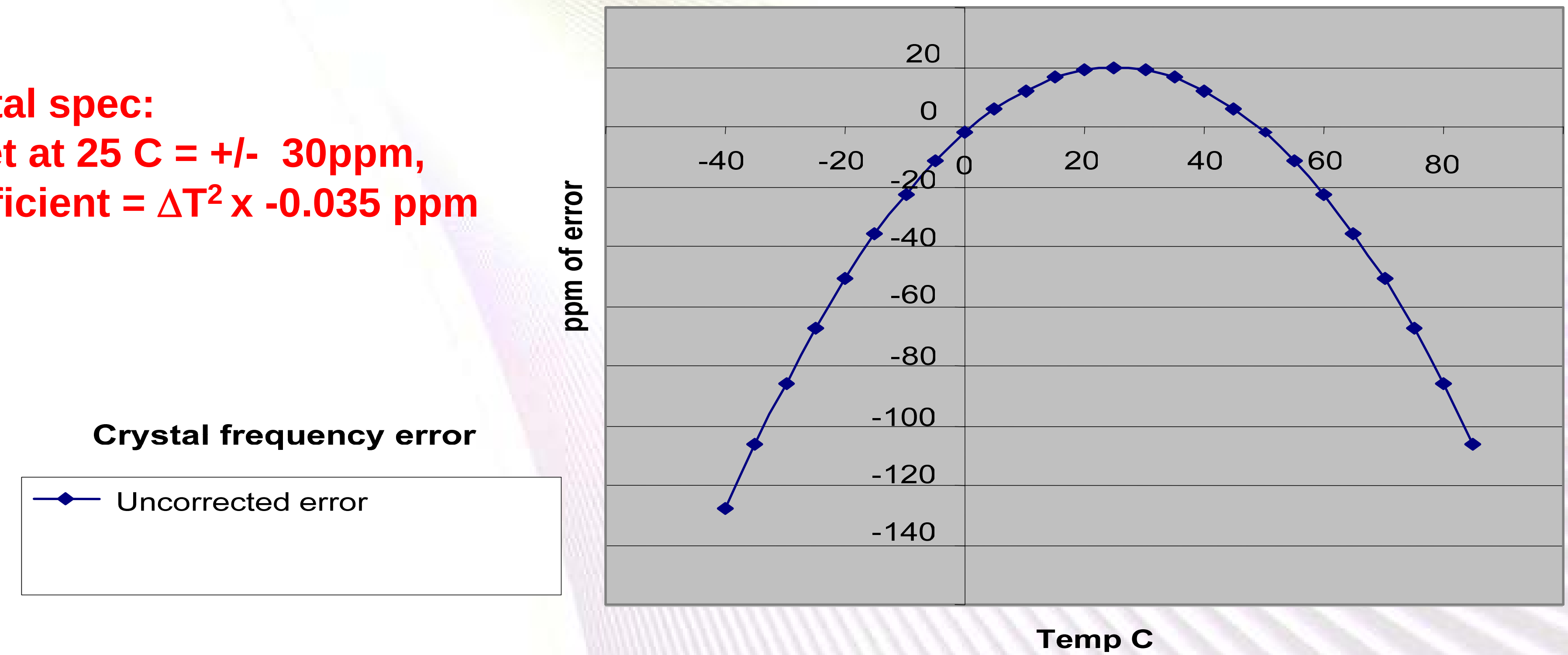
# Agenda

- Building a simple RTC with the MSP430
- What limits the accuracy of the simple RTC?
- What can we do to improve accuracy?
- An example of using correction techniques

# Crystal frequency with no correction

- **Without correction, the crystal frequency changes parabolically with temperature**
- **At high and low temperatures, realistic for severe climates and self-heating, the error is quite large**

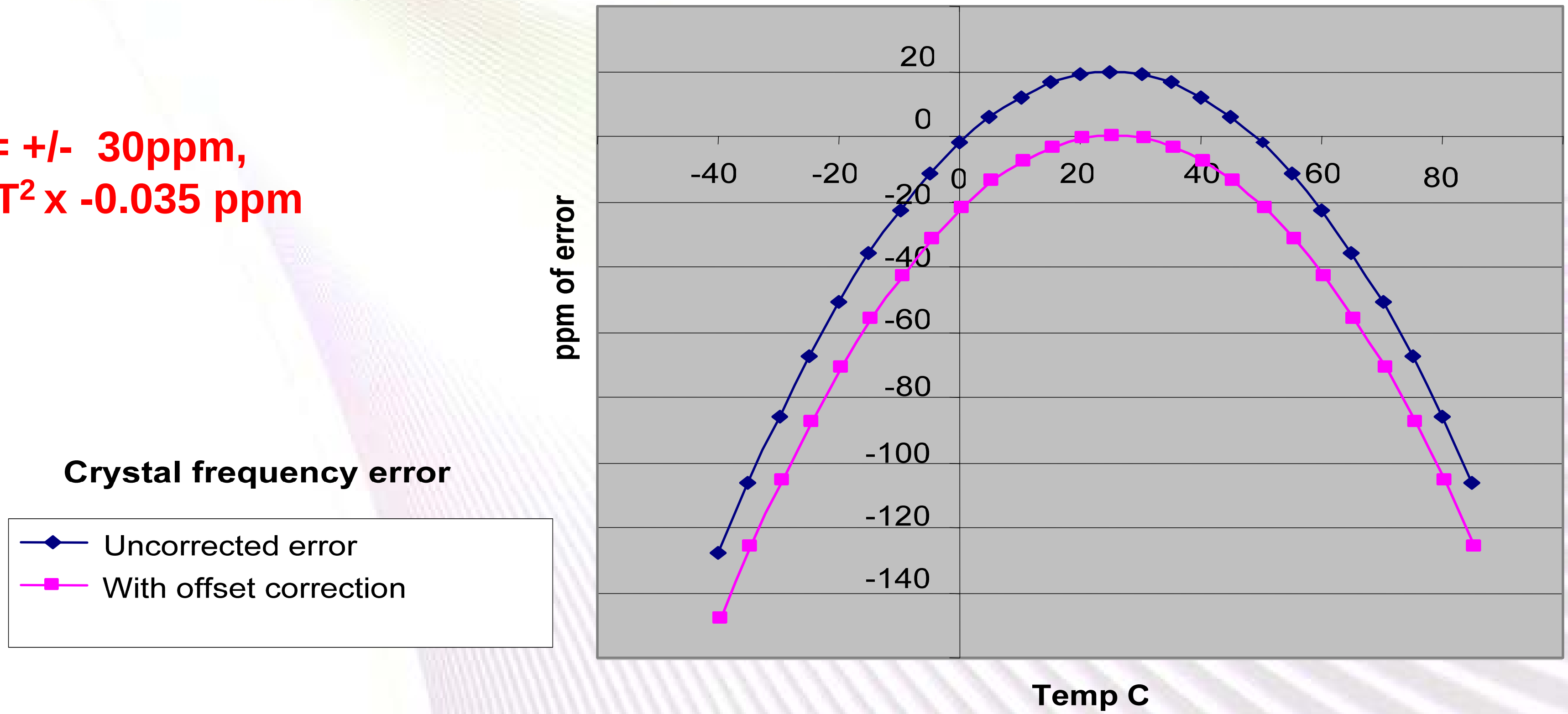
**Crystal spec:**
**Offset at 25 C = +/- 30ppm,**
**Coefficient = $\Delta T^2$ x -0.035 ppm**



# Correcting for manufacturing tolerance

- With compensation for crystal manufacturing tolerance results are better at room temperature
- There is no improvement elsewhere

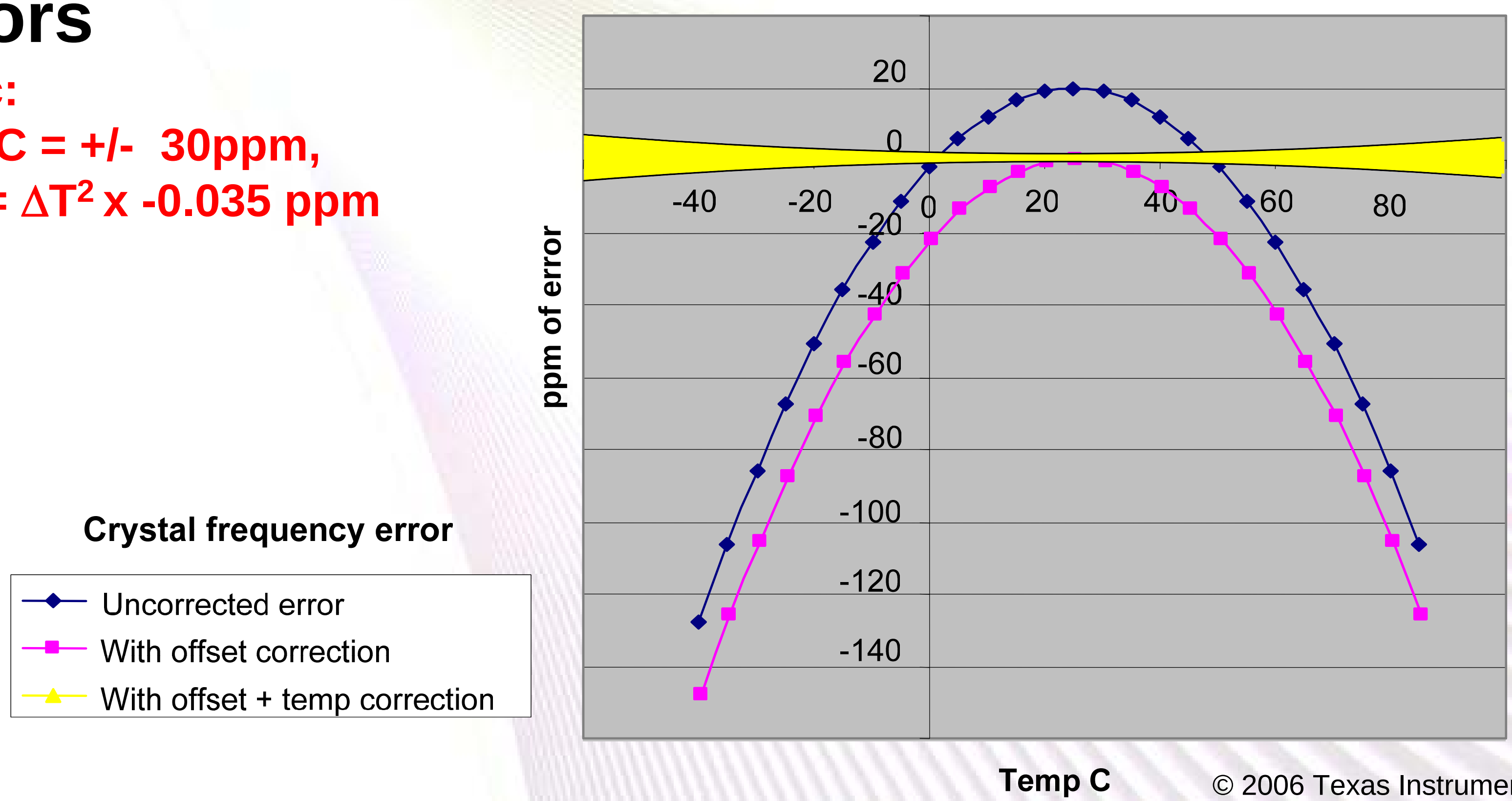
**Crystal spec:**
**Offset at 25 C = +/-  30ppm,**
**Coefficient = $\Delta T^2$ x -0.035 ppm**



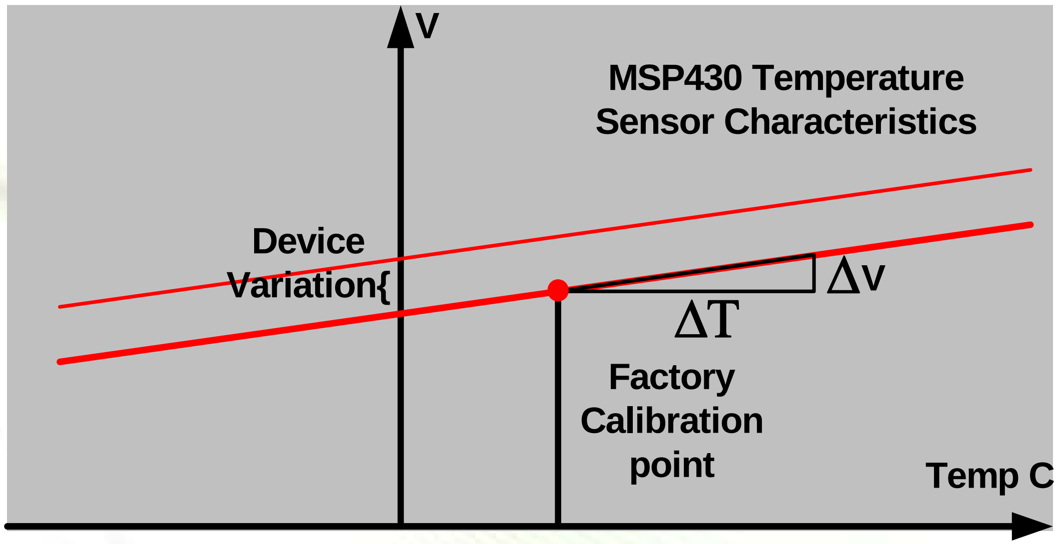
# Fully correcting the crystal frequency

- With compensation for manufacturing tolerance and temperature a spread of errors something like the yellow band is achievable
- Actual results will depend on layout and other factors

**Crystal spec:**
**Offset at 25 C = +/-  30ppm,**
**Coefficient = $\Delta T^2$ x -0.035 ppm**



# Internal temperature sensors




- Each ADC in the MSP430 range – ADC10, ADC12, SD16 and SD16A has a temperature sensor with a near to straight line characteristic
- The slope and the y-axis intercept point varies from sample to sample
- Precise calibration requires measurement at two temperatures
- Using a compromise value for the slope, and finding the y-axis intercept through a measurement at room temperature is good enough for most purposes

# External temperature sensors

- **Some small MSP430 device have no true ADC, but they do have comparator A**
  - Can be used to make a low power slope ADC
- **Used with an external temperature sensor, this could be the basis for a temperature compensated RTC design**

# A ULP temp corrected RTC

- **We can compensate the clock with very little power increase over the simple RTC, if we:**
  - Run the ADC for a very short time, at well spaced intervals, and measure the temperature
  - Calculate the current error in the crystal frequency, and integrate this over time, until we are one second fast or slow
  - Make the time hop by one second, when the integration reaches its threshold
- **Current consumption still extremely low**
  - A 3022 coin cell typically still runs products for 10 years
- **Time hops can be mitigated if necessary**
  - Clock updates can be increased around the hop time, to allow them to be smoothed out

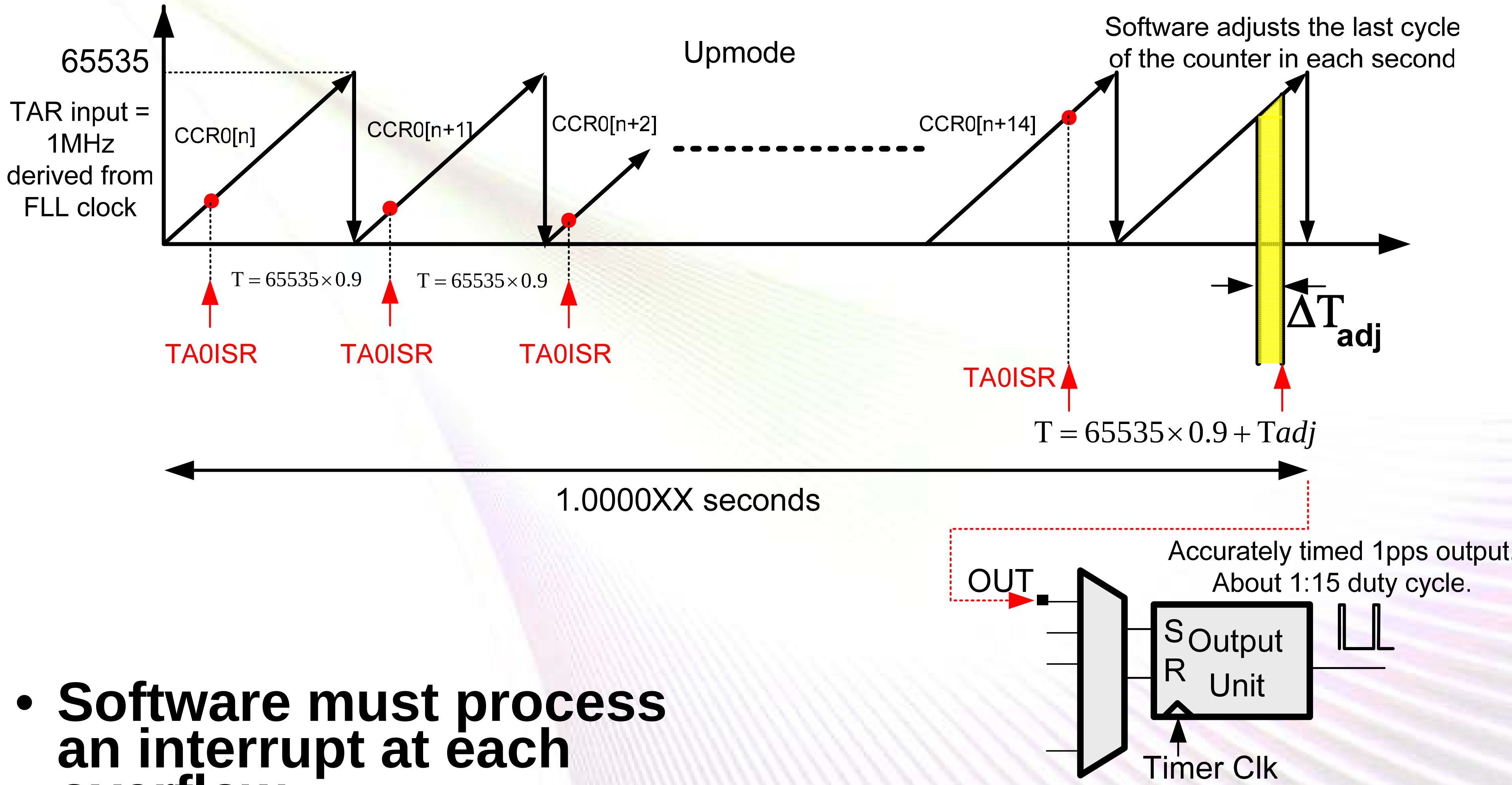
# A low power temp corrected 1pps out

- **Often, approvals, or other tests, require a 1pps output**
- **We can generate this very well on devices with the FLL clock module**
- **We use the fast CPU clock locked to the 32kHz clock**
  - A special feature of Timer A lets us time the output of the 1pps output to within one CPU clock cycle
  - Most of the time the device can be in the LPM0 state
- **Power consumption can still be quite low (~30uA)**
  - The additional consumption is usually not too important. The 1pps output does not need to be enabled at all times

# What about devices without an FLL?

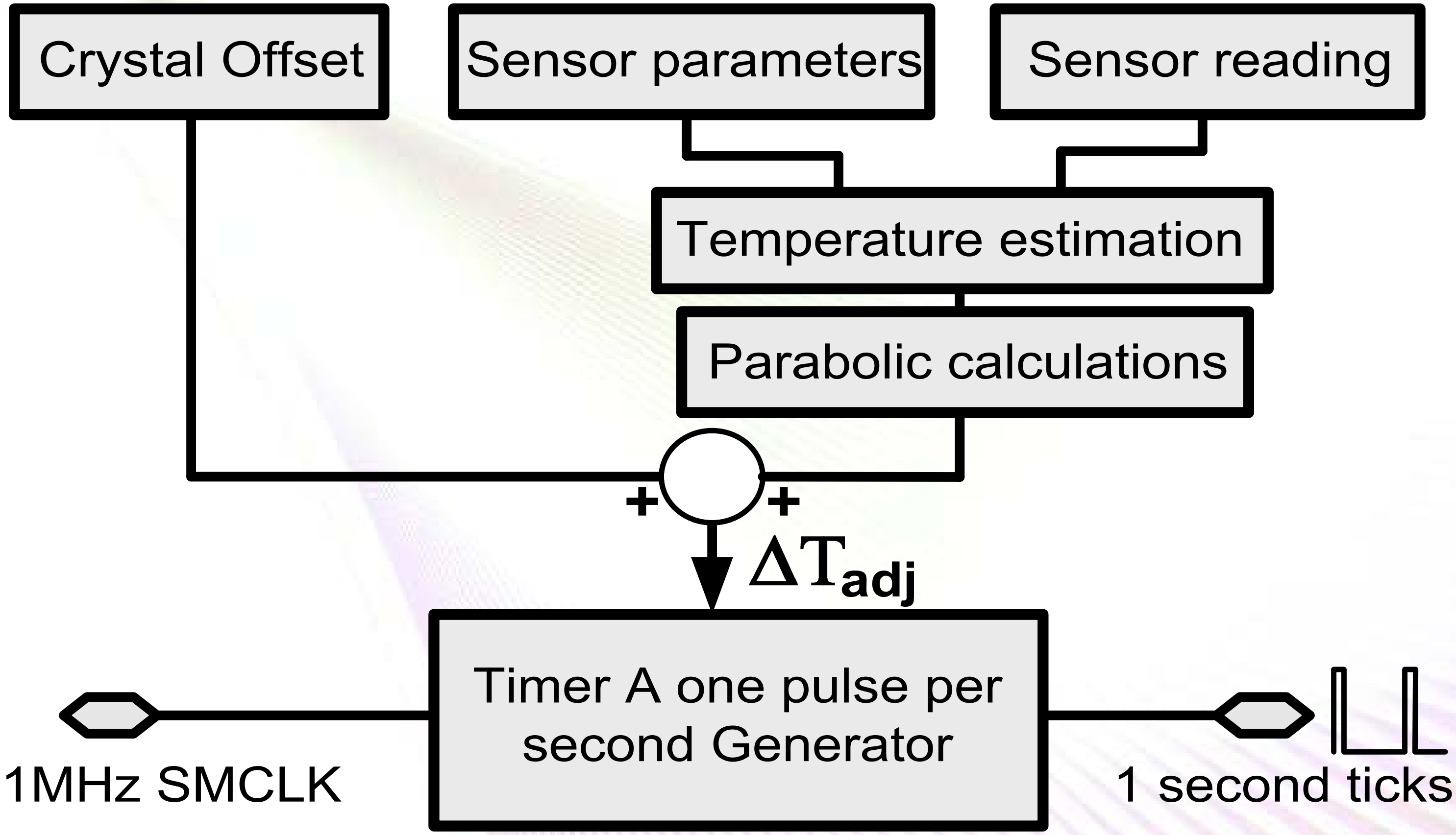
- **We have used the hardware module so far**
  - The simple factory calibration used it
  - The 1pps output used it
  - The corrected RTC only used the 32kHz crystal
- **We can calibrate the basic error by using an accurate external timer**
- **We can build a corrected RTC without the FLL**

# Timer A generating an accurate 1pps




- **Software must process an interrupt at each overflow**
- **Interrupt rate is low**

# Crystal frequency error estimation



# Factory calibration

- **The basic error at the centre of the parabola can be measured quite simply at production time**
  - A precision 32kHz clock is connected to a Timer A input pin on the MSP430
  - Timer A, a CPU clock frequency locked to the MCU's 32kHz crystal, and a little software do the work
  - We tell the meter the calibration room temperature, so the software can allow for the current position on the crystal's parabola
  - About 30 seconds of self calibration is all that is needed
- **The MSP430's internal temperature sensor can be calibrated at the same time**
- **Calibration parameters can be stored in the 430's information memory**

# Agenda

- Building a simple RTC with the MSP430
- What limits the accuracy of the simple RTC?
- What can we do to improve accuracy?
- An example of using correction techniques

# Example: putting it all together in an energy meter application

- The real time clock is updated only once a second
- Correction by adding or subtracting a whole second is generally acceptable in a meter
- We could measure the temperature once a second in normal operation, without affecting energy measurement
- When the meter is in power off/RTC only mode we can measure the temperature once every few minutes, without high overall battery consumption
- We integrate error each second, at the current rate, until we are either:
  - a whole second ahead – we increment an extra second, or…
  - a whole second behind – we skip an increment of one second

# Some Code & Demonstration

- **We will look through key elements of some demonstration code, in C, to see in detail how it works**
  - This code implements both the ultra low power, and 1pps forms of compensated clock
  - This code has been provided, and runs on the ATC board
- **We will show it working in the 1pps mode**
  - Updates occur too infrequently in the ultra low power mode to provide a workable demonstration
- **We will hand calibrate for the sample of MCU and crystal we have, and show the results we can achieve**

# Set things up, Interrupts do the rest

```
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   /* Stop watchdog timer */
    init_fll();                 /* Lock the FLL at 4MHz */
    init_lcd();                 /* LCD to show the time */
    init_basic_timer();         /* Provides 1s kicks */
    set_rtc_sumcheck();         /* Init. the RTC */
    if (one_pps_active)
        init_pulse_stuffer();   /* Init. 1pps generation */
    init_adc();                 /* Init. temp. sensing */
    _EINT();
    /* Sleep, and let interrupts do the work */
    if (one_pps_active)
        LPM0;
    else
        LPM3;
}
```

# Estimating the crystal clock speed...

```
void estimate_current_cycles_per_second(int16_t temperature)
{
    int32_t temp;

    /* A simple IIR filter smoothes noisy sensor readings */
    raw_temperature_from_adc +=
        temperature – (raw_temperature_from_adc >> 3);
    /* Find the temp., in Celsius, based on the sensor
       characteristics found at calibration time. */
    temp = raw_temperature_from_adc
         - temperature_sensor_intercept;
    temp *= temperature_sensor_slope;
    temp >>= 16;
    temperature_in_celsius = temp
```

# ...Estimating crystal clock speed...

```
/* Now we need to calculate the ppm of clock error due to
   the current temperature. */
/* Subtract the centre point of the crystal curve. */
temp -= CRYSTAL_QUADRATIC_CENTRE_TEMPERATURE;
/* Do the parabolic curve calculation, to find the
   current ppm of error due to temperature. */
temp = temp*temp;
temp = (temp*CRYSTAL_QUADRATIC_COEFF) >> 16;
```

# …Estimating crystal clock speed…

```
switch (correction_components)
{
case RTC_CORRECTION_NONE:
    current_rtc_correction = 0;
    break;
case RTC_CORRECTION_CRYSTAL_ERROR:
    current_rtc_correction = crystal_base_error;
    break;
case RTC_CORRECTION_TEMPERATURE:
    current_rtc_correction = -temp;
    break;
case RTC_CORRECTION_CRYSTAL_ERROR_AND_TEMPERATURE:
    current_rtc_correction = crystal_base_error - temp;
    break;
}
current_estimated_cycles_per_second =
      CRYSTAL_BASE_CYCLES_PER_SECOND
    + current_rtc_correction;
```

# Prepare to generate the 1pps output

```
void init_pulse_stuffer(void)
{
    TAR = 0;
    TACCR0 = 0;
    TACTL = TASSEL_2 | MC_2 | ID_2;
    P1SEL |= BIT0;
    P1DIR |= BIT0;
    TACCTL0 = OUTMOD0 | CCIE;
    /* Initialize the cycles per second with an
       approximation */
    current_estimated_cycles_per_second =
        CRYSTAL_BASE_CPS + crystal_base_error;
    cycles_left_this_second =
        current_estimated_cycles_per_second;
}
```

# Generate a precise 1pps output

```
void update_pulse_stuffer(void)
{
    uint16_t step;
    step = 65536U - 655U;
    cycles_left_this_second -= step;
    if (cycles_left_this_second <= 655)
    {
        step += cycles_left_this_second;
        TACCTL0 &= ~OUTMOD2;
        cycles_left_this_second =
             current_estimated_cycles_per_second;
    }
    else
    {
        TACCTL0 |= OUTMOD2;
    }
    TACCR0 += step;
}
```

# Update the RTC

```
int update_rtc(void)
{
    integrated_rtc_correction += current_rtc_correction;
    if (integrated_rtc_correction >= CRYSTAL_BASE_CPS)
    {
        integrated_rtc_correction -= CRYSTAL_BASE_CPS;
        /* We need to add an extra second to the RTC */
        bump_rtc();
    }
    else if (integrated_rtc_correction <= -CRYSTAL_BASE_CPS)
    {
        integrated_rtc_correction += CRYSTAL_BASE_CPS;
        /* We need to drop a second from the RTC */
        return;
    }
    bump_rtc();
}
```

# A Real-Time Demonstration

- We will see the software’s 1pps mode in operation
- We will hand calibrate for the sample of MCU and crystal we have, using an accurate timer-counter
- We will show the results we can achieve with an accurate timer-counter, and…… a hairdryer!
- We cannot run this as a hands on lab session, as each person would need an accurate timer-counter…
- All the materials are provided, so you can try this for yourselves later

# IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| RFID | www.ti-rfid.com | Telephony | www.ti.com/telephony |
| Low Power Wireless | www.ti.com/lpw | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265