TI Technology Days 2010

MSP430™ Ultra-Low Power MCUs
The next generation

Yunus Karaborek

MSP430 Agenda

- **Introduction**
  - MSP430-Enabled applications
  - MSP430 Core Values
- **MSP430F5xx Architecture**
  - **MSP430F5xx core and key modules**
  - **New connectivity solutions**
    - USB Peripheral Overview including resources, support & tools
    - RF Peripheral Overview including resources, support & tools
- **MSP430 Value Line G2xx**

Technology Roadmap Overview

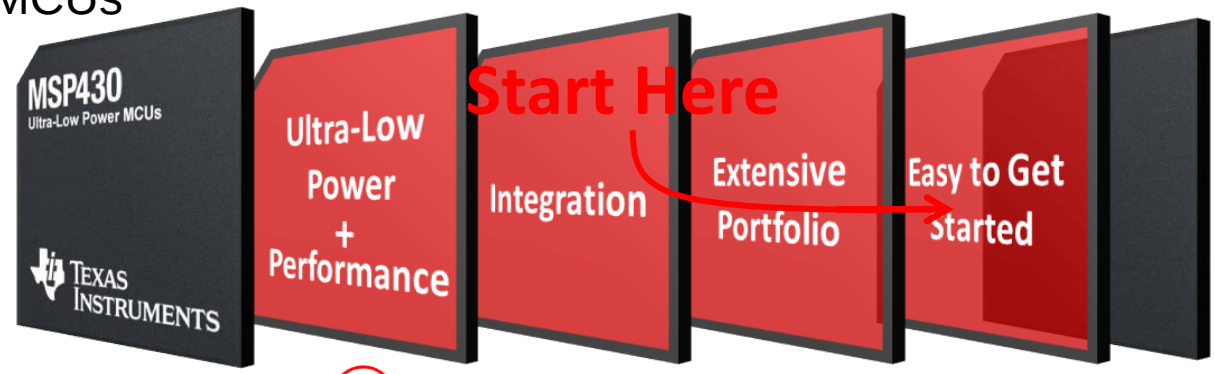
  - FRAM
  - Native 0.9V technology

MSP430 Ultra Low Power MCU

MSP430-Enabled Solutions at a Glance

MSP430 Ultra-Low Power MCUs

Think MCU
Think MSP430



Ultra-Low Power + Performance

World's Lowest Power MCU
- 6 Low Power Modes & Instant Wake-up
- 110µA/MIPS | 0.3 µA Standby | 0.1 µA RAM

Do More with MSP430
- Leading code density, autonomous peripherals, at low cost and the lowest power
- CPU Speed: 8MHz – 25MHz

Integration

Intelligent Analog & Digital Peripherals
- Do more without the CPU | Faster, Lower power
- Featuring 16-bit ADCs, 12-bit DACs, DMA, MPY, Comparators, LCD, USB, RF, and more

COMING SOON
- Embedded memory technology breakthrough: FRAM

Extensive Portfolio, Low Cost Options

Find the right MCU for you
- 200+ Devices with one consistent architecture

Flash: 0.5 kB – 256 kB **RAM**: 128 B – 16 kB
Pins: 14 - 113 **Package Types**: 22

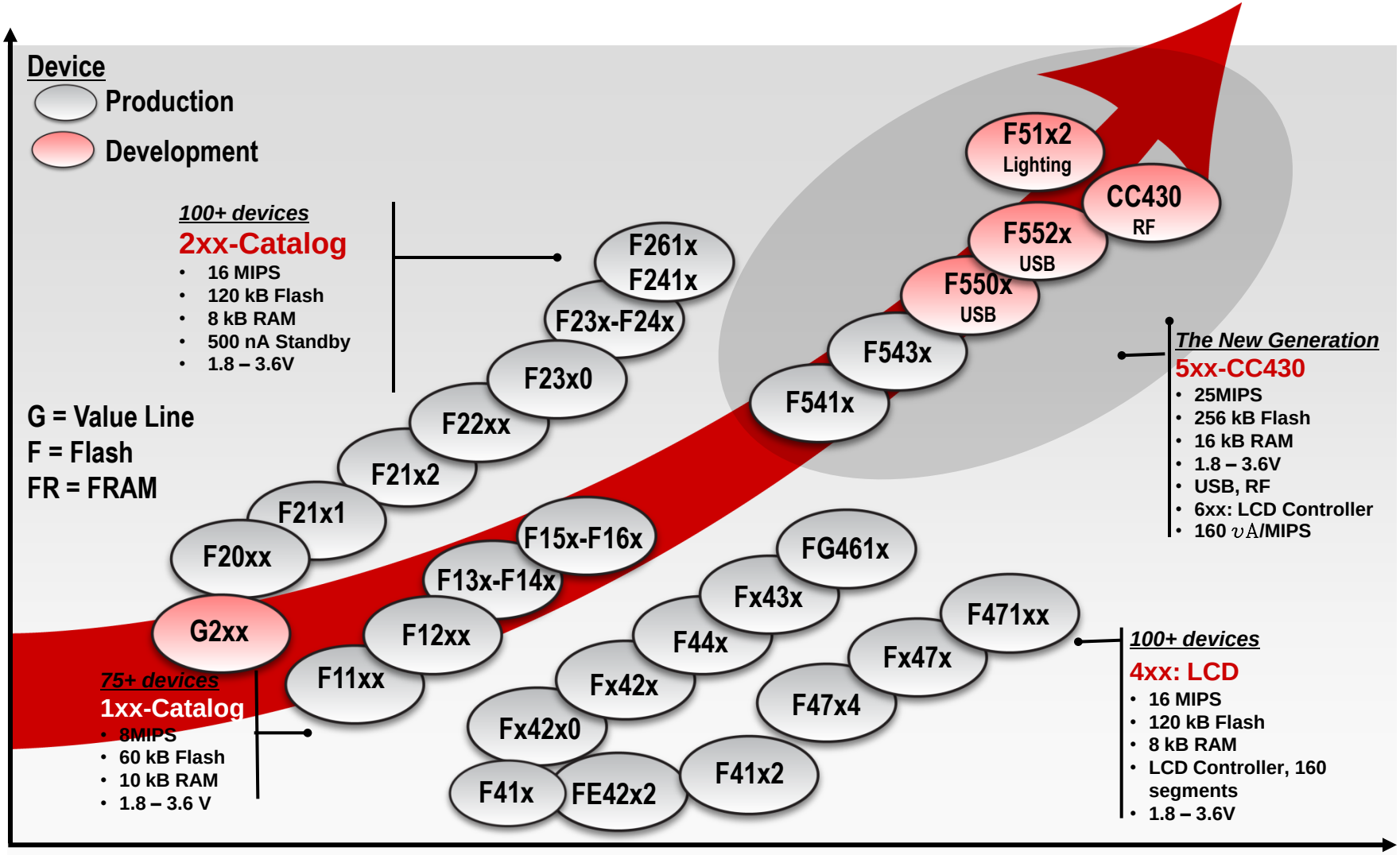
NEW MSP430 Value Line
- Don't Settle | 16-bit Performance, 8-bit Price
- Starting @ $0.25 USD

Easy to Get Started

Consistent Development Environment
- One development tool for all MSP430 devices
- Code compatibility throughout portfolio

Get started now!
- eZ430 development kits @ $20 USD & Free IDEs
- eZ430-Chronos development watch @ $49 USD
- 1000s of starter code examples & App notes

MSP430 Portfolio + Roadmap



Introduction to the MSP430F5xx Architecture + an overview of the new USB and RF blocks

=

Ultra-Low Power Is In Our DNA

- MSP430 designed for ULP from ground up
- Peripherals optimized to reduce power and minimize CPU usage
- Intelligent, low power peripherals can operate independently of CPU and let the system stay in a lower power mode longer www.ti.com/ulp

✓ Multiple operating modes
  - 100 nA power down (RAM retained)
  - 0.3 µA standby
  - 110 µA / MIPS from RAM
  - 220 µA / MIPS from Flash

✓ Instant-on stable high-speed clock

✓ 1.8 - 3.6V single-supply operation

✓ Zero-power, always-on BOR

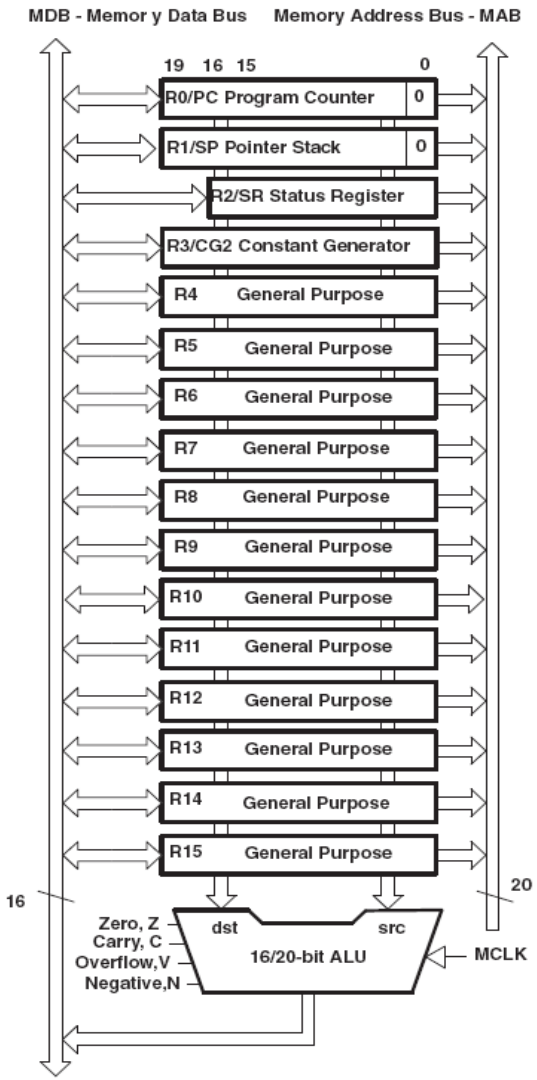
✓ <50nA pin leakage

✓ CPU that minimizes cycles per task

✓ Low-power intelligent peripherals
  - ADC that automatically transfers data
  - Timers that consume negligible power
  - 100 nA analog comparators

✓ Performance over required operating conditions

5xx MSP430 CPUXv2



- Efficient, ultra-low power CPU
- C-compiler friendly
- CPU registers expanded to 20-bits
- RISC architecture
  - 27 core instructions
  - 24 emulated instructions
  - 7 addressing modes
  - Constant generator
- Address instructions
  - Direct 20-bit CPU register access
  - Atomic instructions
- Instruction compatible with all MSP430's
- Extension word allows all instructions
  - Direct access to 1MB address space
  - Bit, byte, word and address-word data

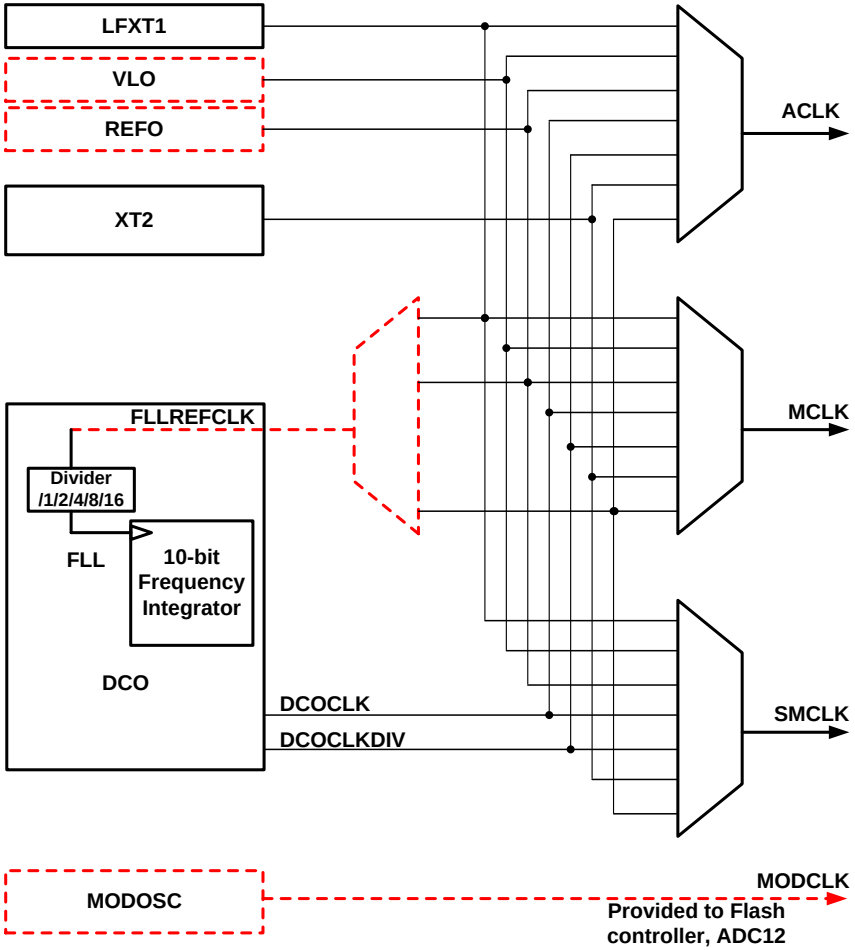
Unified Clock System | Speed & Flexibility

FEATURES

- Orthogonal clock system
  - Any source can drive any clock signal
- 2 Integrated clock sources:
  - REFO: 32kHz, trimmed osc.
  - VLO: 12kHz, ultra-low power
- DCO & FLL provide high frequency accurate timing
- MODOSC provides bullet proof timing for Flash
- Crystal pins muxed with I/O function

BENEFITS

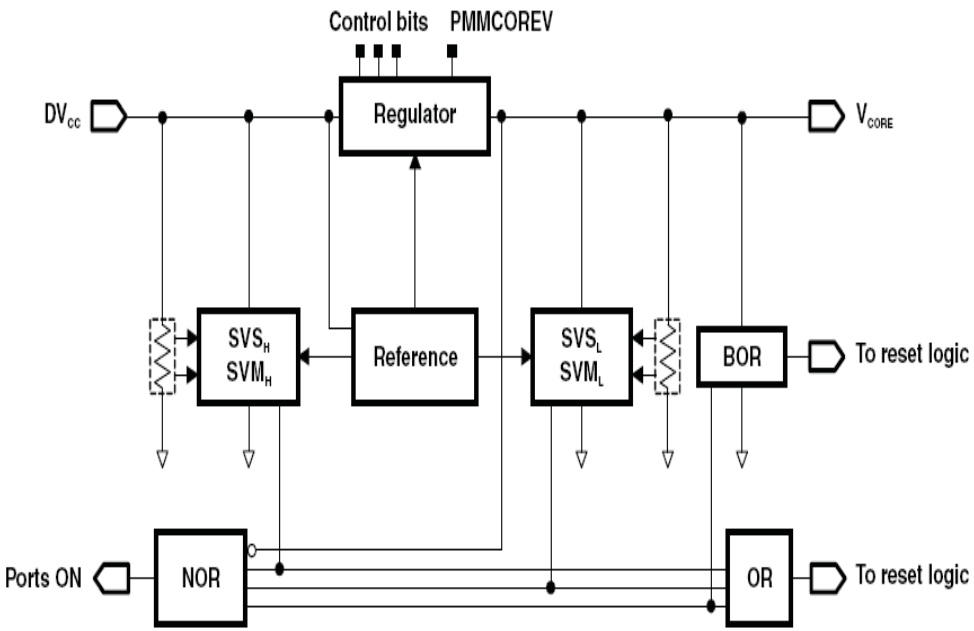
- Ultra-Low Power functionality
- Best balance of performance and low power
- Enables flexible and capable Low Power Modes
- Instant on with fast oscillator wakeup



Power Management Module

FEATURES

- Integrated LDO
- V_{CORE} level programmable
- Flexibility in processing performance vs. power
- Integrated *supervision & monitoring*
- *Zero-power* BOR
- Five integrated supervisors
  - SVSH, SVSL, SVMH, SVML & BOR

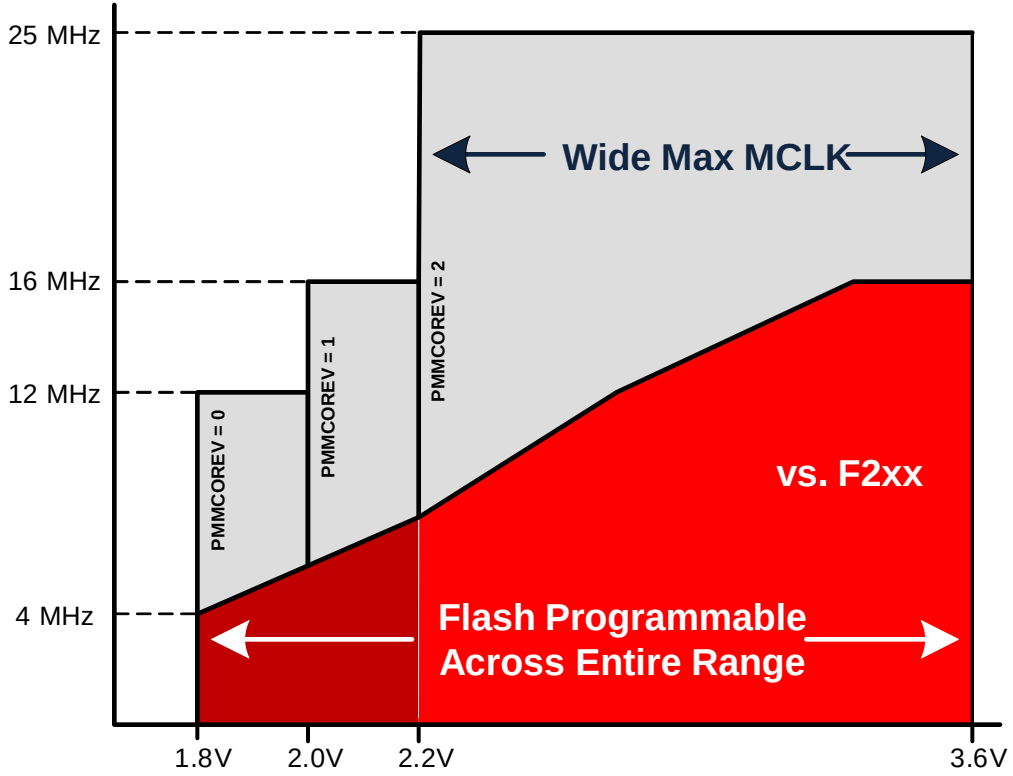


BENEFITS

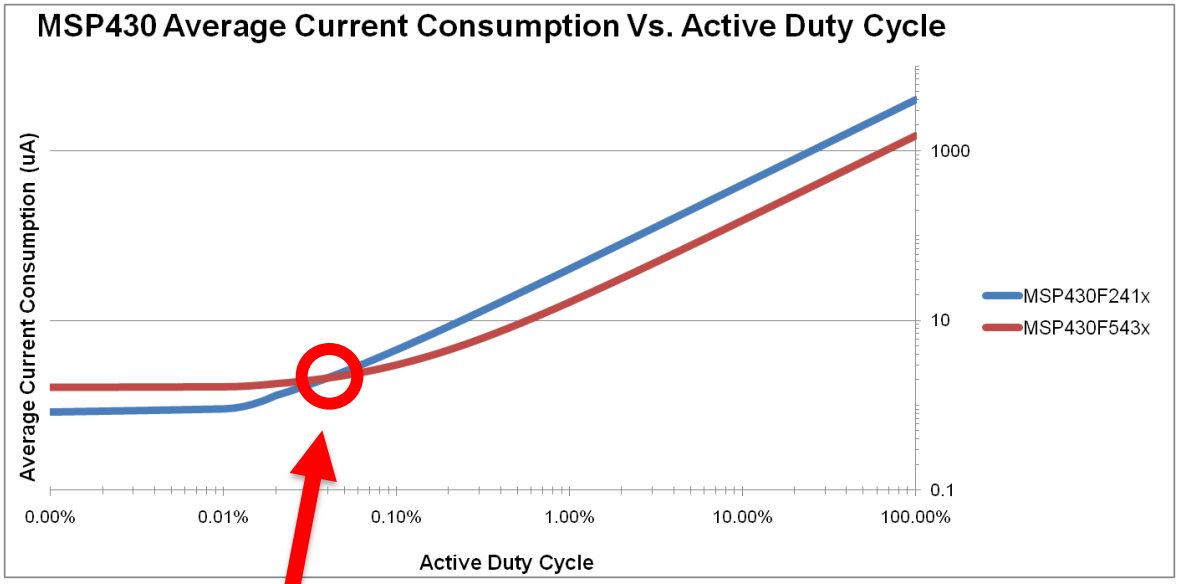
- Ultra-Low Power Functionality
- Ensure proper operation in power on and off sequences

MSP430F5xx Speed and Flexible Operation

- 25MHz peak performance
- More performance across V_{CC} range
  - Flash ISP @ 1.8V
  - 12MHz @ 1.8V
  - 25MHz @ 2.4V-3.6V



Which MCU? MSP430x2xx vs MSP430F5xx



	F241x	F543x
Active @ 8MHz	4.25 mA	1.32 mA
LPM3	0.8 uA	1.1 uA

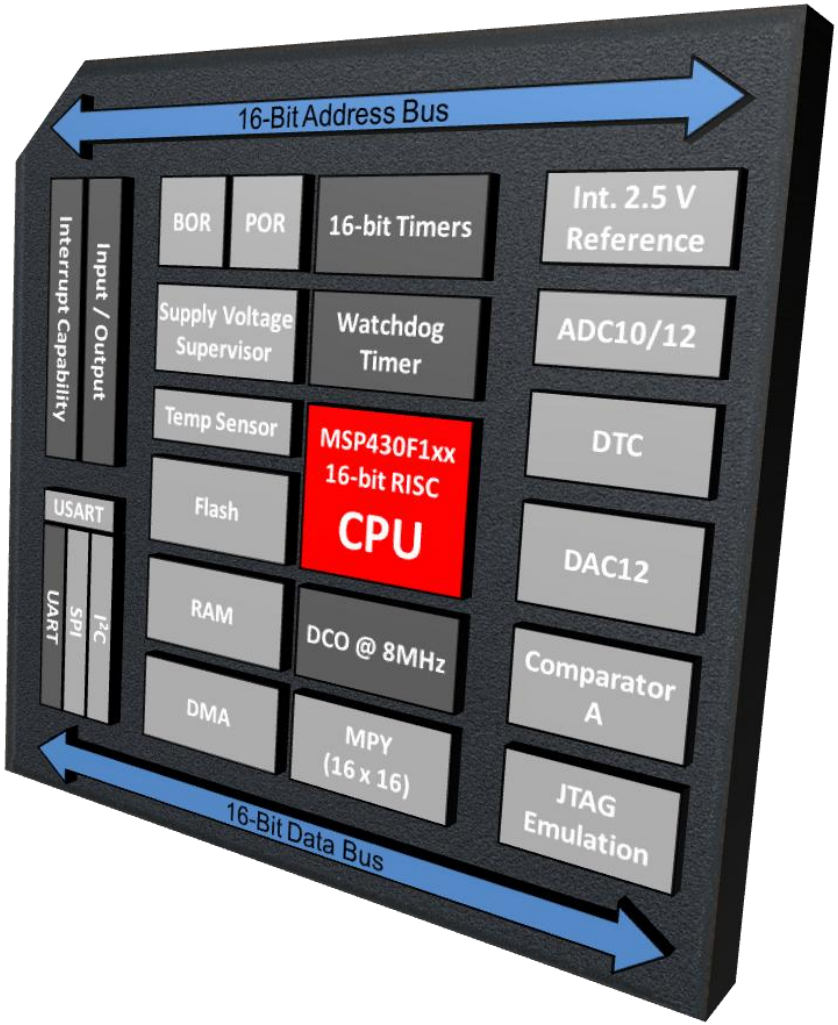
- MSP430x2xx has lower LPM3 consumption
- MSP430F5xx thas lower active power consumption

The crossover is at ~0.04% Active

Or about 35 seconds/day

This means that if the CPU is Active >0.04% of the time (or >35 seconds per day), the lower Active Mode power of the F5xx outweighs the lower LPM3 current advantage of the F2xx device.

Performance through Integration



- The 200+ MSP430 devices offer high-performance integration

Snapshot of Integrated Peripherals

• ADC10, ADC12	• WDT	• AES
• SD16	• Basic Timer	• USB
• Comparator	• RTC	• SPI
• DAC12	• PMM	• I²C
• DMA	• BOR	• LIN/IrDA
• Multiplier	• SVS	• SCAN_IF
• OpAmp	• EDI	• ESP430
• Timer	• RF Front End	• LCD

- Various levels of integration allows designers to find the right MSP430
- Integration enables smaller physical footprints and minimizes system costs

Integrated USB Connectivity with MSP430 MCUs

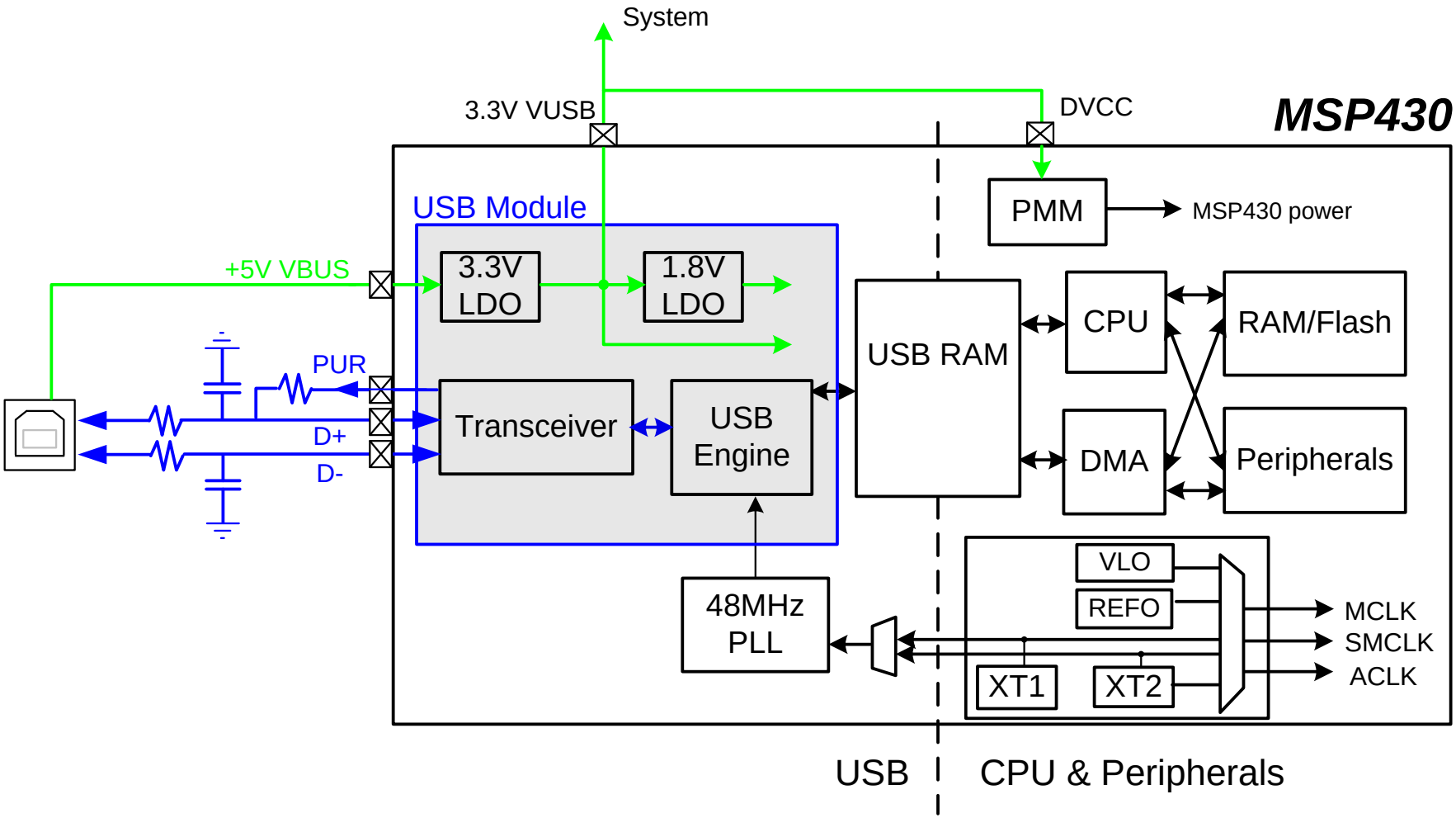
Popular USB Terms

- **Host Vs Device: USB is a master-slave network with a single master or 'host' and multiple slaves or 'devices'**
  - **A single USB host can support up to 127 devices**
- **OTG: is a Dual mode implementation, can act as either host or device**
- **VID: Vendor ID uniquely identifies the manufacturer e.g. Apple**
- **PID: Product ID distinguishes between products for a given vendor e.g. iPhone, iPod Nano etc**
- **USB Descriptors: give the host information about the device such as USB version supported, PID and VID**
- **Enumeration: process of obtaining descriptors to determine what device is connected to the bus and load drivers associated with its device class**
- **Device Class: defines a protocol for a particular kind of device (e.g. mass storage, human interface, communications etc)**

MSP430 Bus Speed/Role

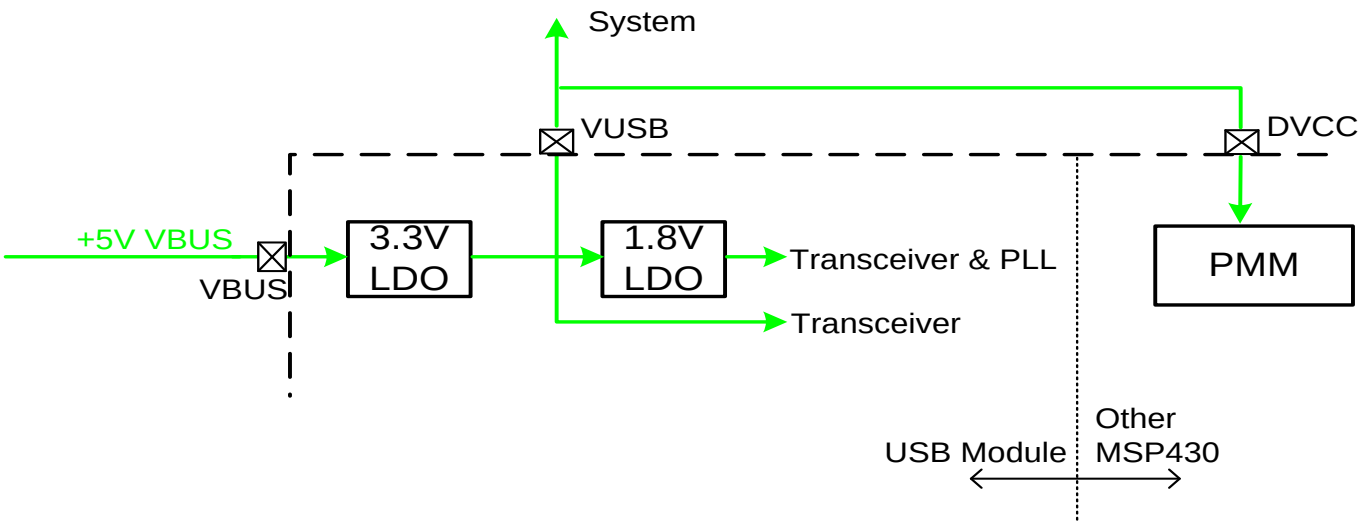
- **USB speed classes:**
  - **Low Speed: 1,5Mbit / sec**
  - **Full Speed: 12Mbit / sec**
  - **High Speed: 480Mbit / sec**
- **MSP430's USB is a full-speed *device*, for applications such as:**
  - **Uploading data from a mobile device to a host (PC, cell phone, PDA, etc.)**
  - **Downloading a firmware update from a host**
  - **Controlling/configuring the device from a host**

MSP430 USB Block Diagram & Data Flow



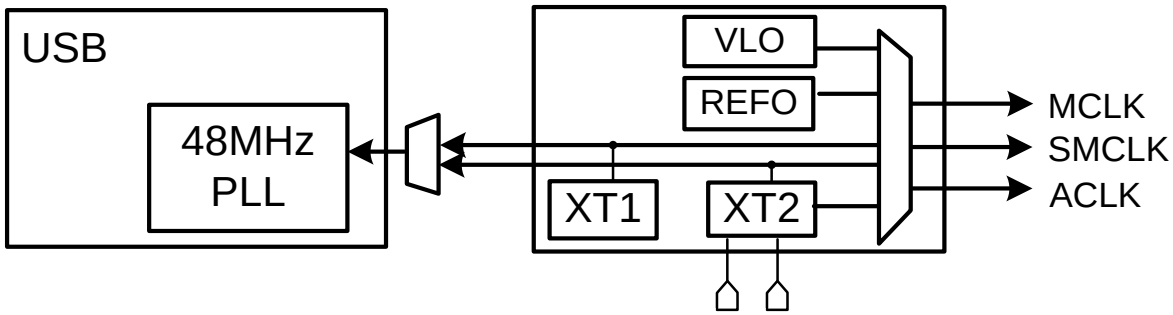
USB Power System (UPS)

- **Integrated LDO**
  - **Operates directly from 5V V_{BUS}**
- **LDO output is provided externally**
  - **Can be used to power device DV_{CC}**
  - **Can be used to power non-MSP430 components as well (<12mA)**
- **DV_{CC} can also be provided through other means**
  - **Easy flexibility for bus/self-powered operation, USB battery charging, etc.**



USB Clocking

- **USB clocked from HF crystal**
  - **Or, input a precise clock from another device**
- **Integrated PLL revs it to 48MHz**
  - **Usually powered down during USB suspend, & re-activated when resumed**
- **Can also use clock for other system functions**
- **PLL is programmable – can use many reference freq's**
  - **Choose one used elsewhere in the system**
  - **Or choose the cheapest crystal you can find**
  - **>4MHz (crystal) or >1.5MHz (bypass mode)**



MSP430 USB Descriptor Tool

- **USB Descriptors: give the host information about the device such as USB version supported, PID and VID**
- **Often the first step towards developing a USB application**

- **Developing structures is tedious & error-prone**
  - **One error & enumeration fails; or worse, appears to succeed but doesn't**
  - **Debugging is time consuming**
  - **Which settings to use? Compatibility with hosts?**

- **TI provides a Windows utility to make generating descriptors easier**
  - **Saves debugging effort and development time**

- **Descriptor files are integrated into the TI USB API stacks (default descriptors are provided with the stack)**

MSP430 USB BSL

- **On-chip bootstrap loader (BSL) can communicate via USB**
  - **Based on Human Interface device class**
  - **No installations needed**
- **Invoking the USB BSL**
  - **For a blank device: Supply V_{BUS} (plug in USB cable)**
  - **For a programmed device: Hold switch S3 down (PUR pin) on target board when supplying V_{BUS}**
  - **Call from user application**
- **Application report for creating a custom BSL is available:**
  - ***[SLAA450] Creating a Custom Flash-Based BSL***
    Related code examples can be downloaded from www.ti.com/lit/zip/slaa450

Easy to use Online Resources

MSP430™16-bit Ultra-Low Power MCUs - 5xx 25MHz Series - MSP430F5529 - TI.com - Windows Internet Explorer provided by Texas Inst

http://focus.ti.com/docs/prod/folders/print/msp430f5529.html#toolssoftware

Tools & Software

Name	Part #	Company	Tool / Sof
Code Composer Studio (CCStudio) Integrated Development Environment (IDE) - v4.x	CCSTUDIO	Texas Instruments	Code Comp
MSP430 USB Debugging Interface	MSP-FET430UIF	Texas Instruments	Developmen
MSP430F55xx 64-Pin Target board only	MSP-TS430RGC64USB	Texas Instruments	Developmen
MSP430F55xx USB 80-Pin Target board	MSP-TS430PN80USB	Texas Instruments	Developmen
IAR Embedded Workbench Kickstart - Free 4KB IDE	IAR-KICKSTART	Texas Instruments	Developmen
MSP430F55xx 80-Pin FET tool and target board combination	MSP-FET430U80USB	Texas Instruments	Starter Kits

MSP430F552x C Examples (IAR and CCSv4) (Rev. A) (ZIP 121 KB)
03 Feb 2010 2,454 views

MSP430F5529 USB Sample Kit - Getting Started and Support Files (Rev. B) (ZIP 1895 KB)
03 Feb 2010 2,027 views

MSP430 USB CDC+HID API Stacks (Rev. A) (ZIP 7006 KB)
03 Feb 2010 2,306 views

MSP430 Field Firmware Update Demo (ZIP 410 KB)
20 Jul 2009 1,165 views

MSP430 USB Descriptor Tool (ZIP 10042 KB)
20 Jul 2009 1,117 views

Development Tools

C Code Examples

USB Stacks

USB Field Update tool

USB Descriptor tool

Support Collateral

- **F5529 USB Sample Kit – Getting Started & Support Files**
  - ***Starting a USB Design with MSP430* → 20-page overview of MSP430 USB**
  - ***Getting Started* – detailed walk-through for using FET target board demo**
- **MSP430 Field Firmware Updater Demo**
  - ***.exe demo for firmware update**
  - **Visual Studio project coming soon**
  - ***USB Field Firmware Updates on MSP430 MCUs* → 20-page overview**
- **MSP430 USB Descriptor Tool**
  - **Help pane is integrated; readme file in the zip**
  - **If you have trouble running it – install latest Java Virtual Machine**
- **MSP430 USB CDC+HID API Stacks**
  - **API stack for CDC & HID**
  - **Programmer's Guide for each**

Complete support packages for USB development

Tools & Software

MSP430F5x USB Sample Kit

USB BSL
- Supporting device programming
- Field Firmware updates

Descriptor tool
- Configures stack functions

Software tool chain (CCE & IAR)

Support

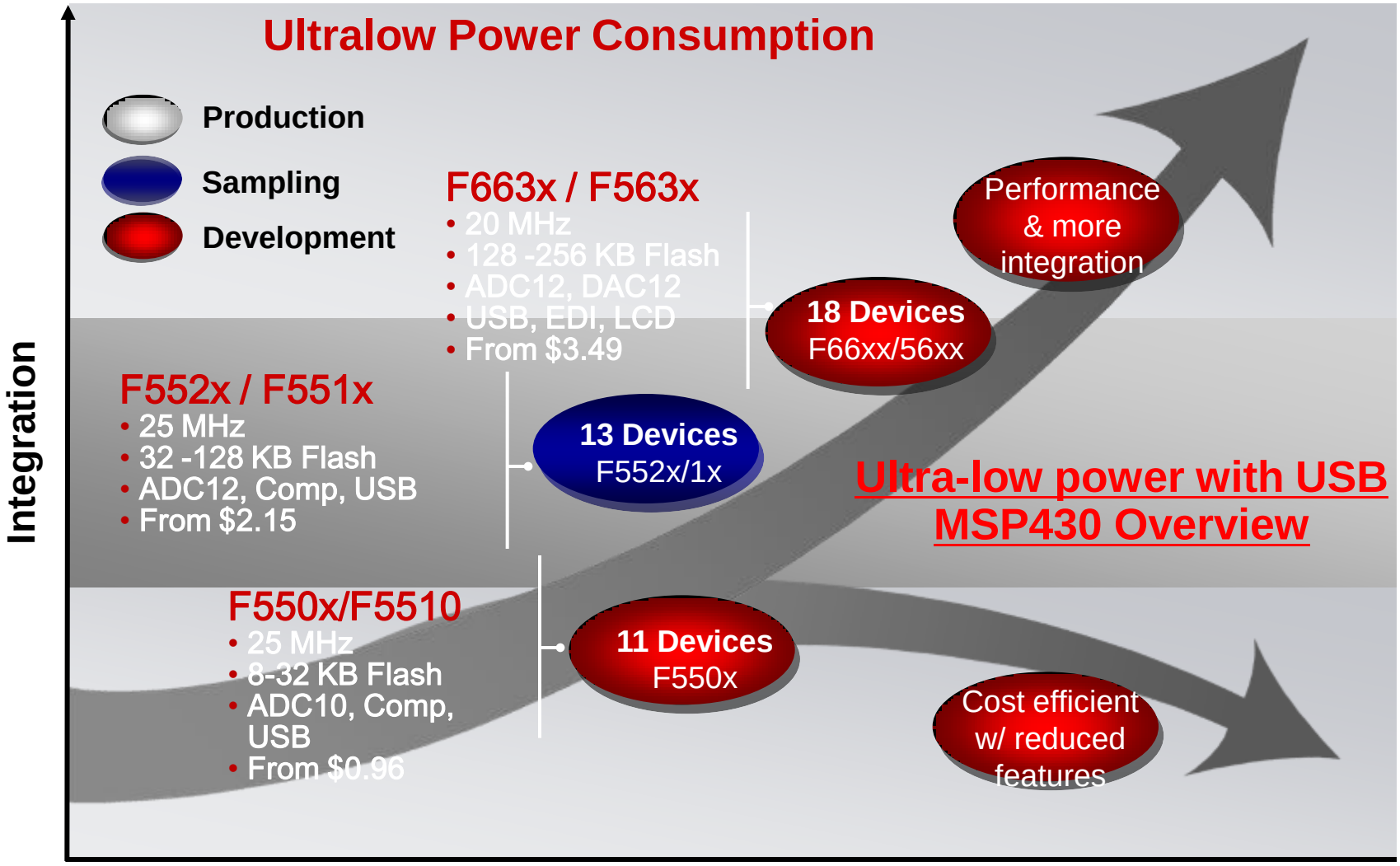
TI supports multiple software stacks:
- Communication Device Class (CDC)
- Human Interface Device (HID)
- Mass Storage Class (MSC)

Additional third party software support

VID Request for embedded USB products

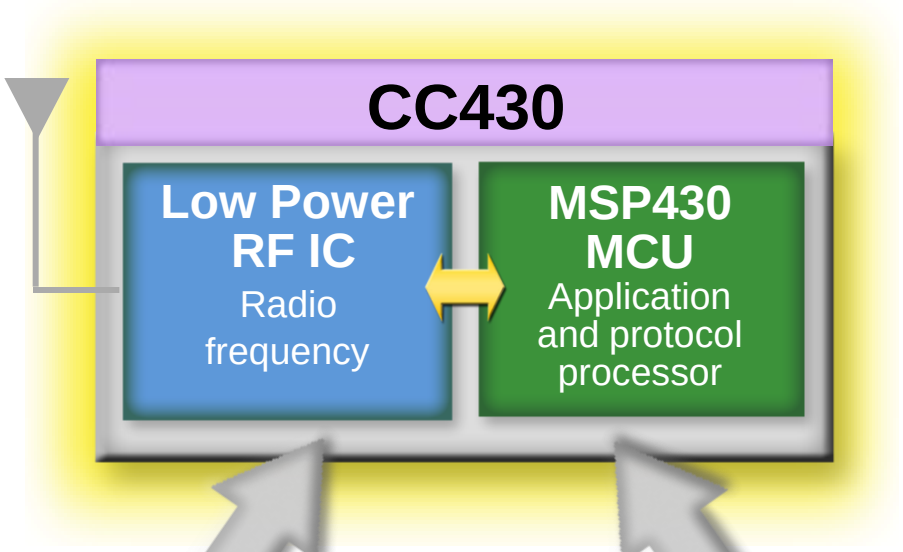
FREE Vendor ID/ Product ID sharing program

MSP430 w/USB Roadmap



Single-chip Low-Power Wireless System using the CC430

What is the CC430?



Low-power RF SoC

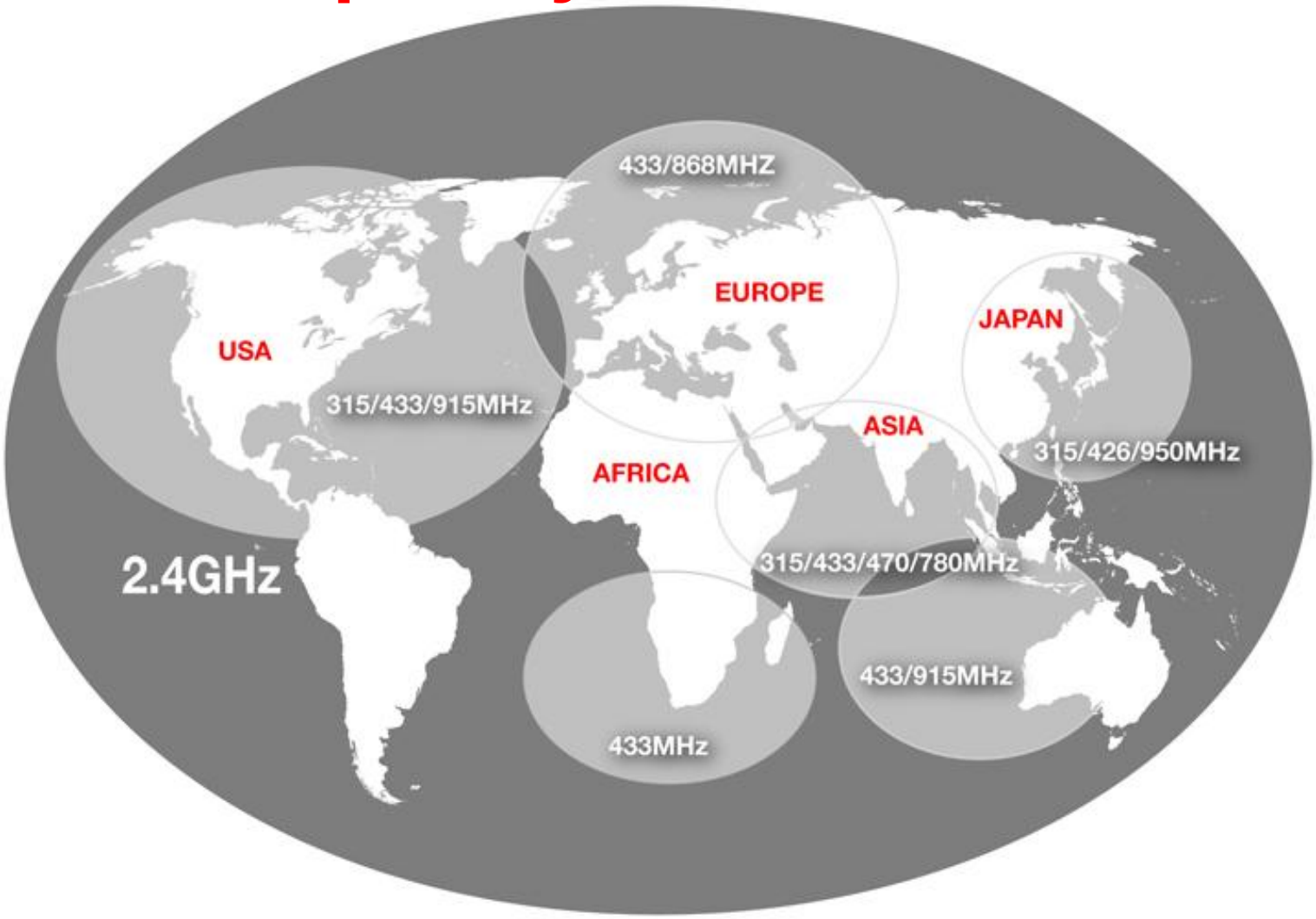
Low Power < 1 GHz RF Transceiver

- High sensitivity
- Low current consumption
- Power output from -30 dBm to 10 dBm
- Flexible data rate & modulation format
- 300-348 MHz, 389-464 MHz, 779-928

MSP430 5xx MCU

- Ultra-low power
- High analog performance
- High level of integration
- Ease of development

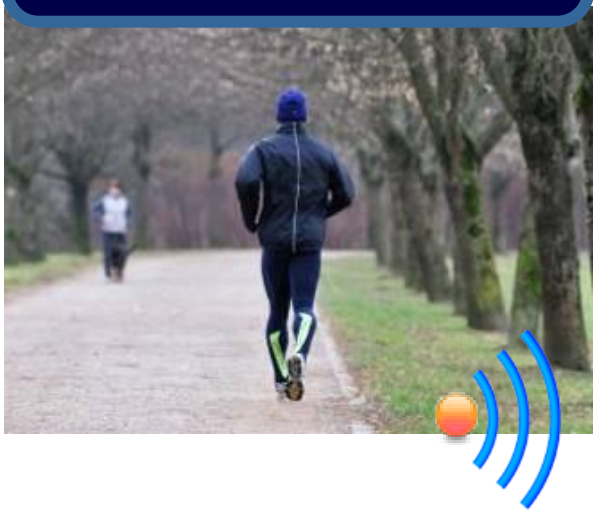
Global frequency bands



CC430 Supports: 300-348MHz, 387- 464MHz and 779-928MHz

Broad range of applications benefit from CC430's low power networking feature

Consumer / personal networking



- Watch/shoe combination for monitoring of miles and calories
- Enough processing for wireless networking and batteries that 10+ years

Industrial remote monitoring

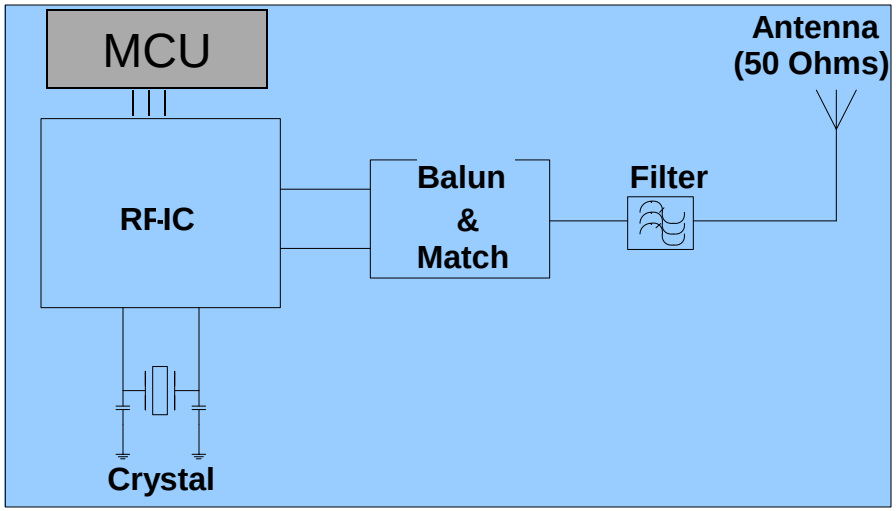
- Low power sensor networks for innovative applications like remote monitoring for stress cracks
- Harvest energy from motion, vibration and heat

Home and Industrial Automation

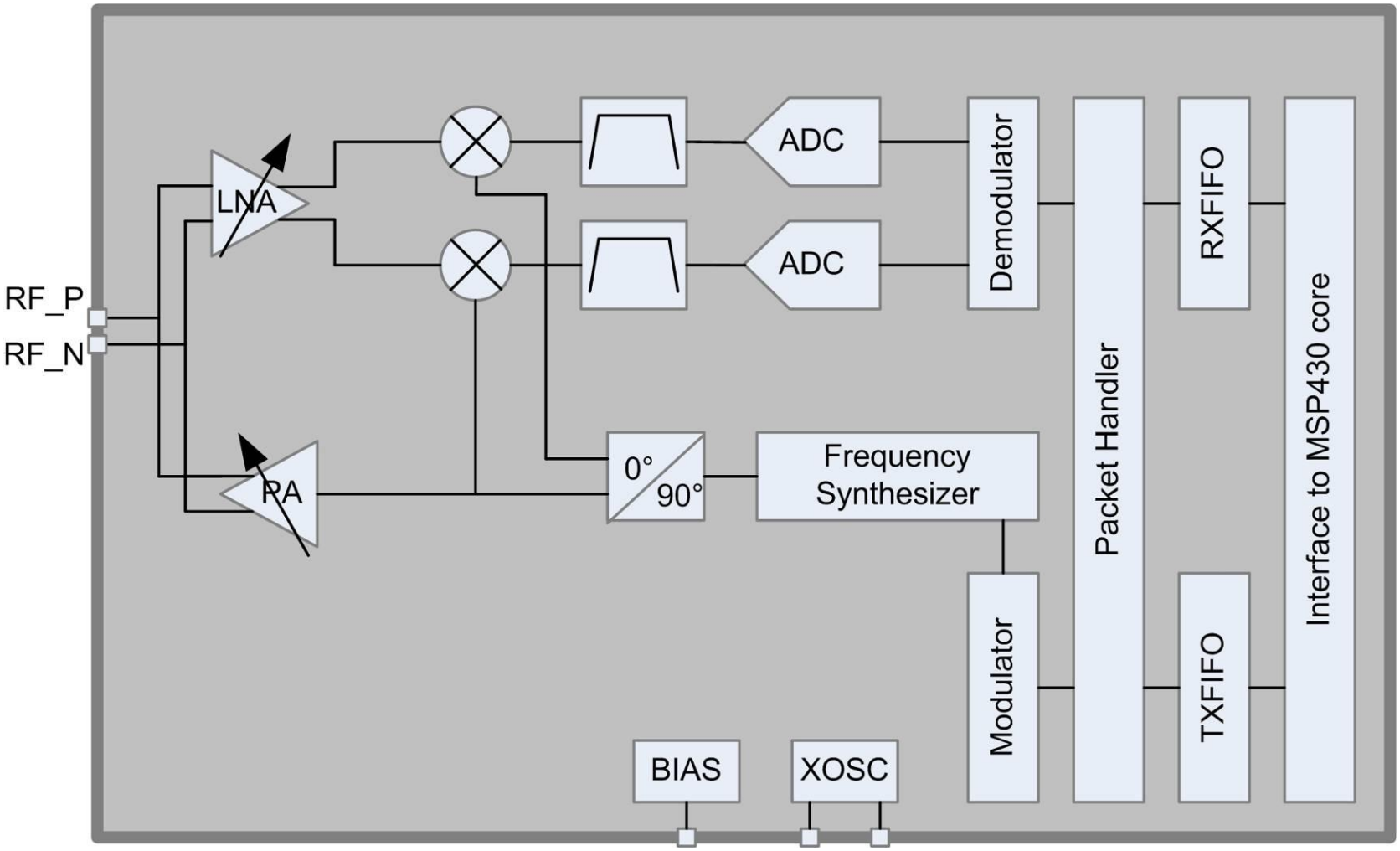
- Wireless Alarm and security Systems
- Automated Meter Reading (AMR)
- Industrial Monitoring and control

Basic Building Blocks of an RF System

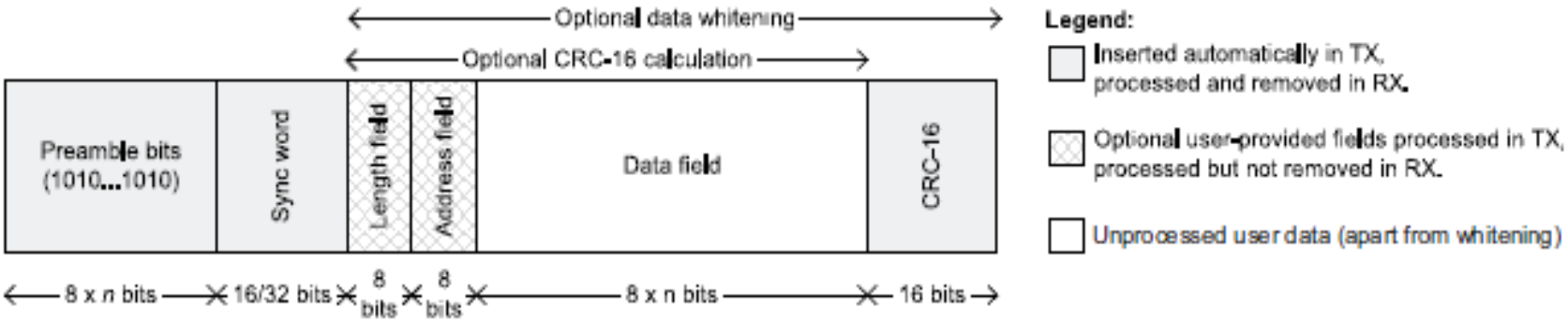
- RF-IC
  - **Transmitter / Receiver**
  - **Transceiver**
- Microcontroller
  - **For protocol and applicaiton processing**
  - **System-on-Chip (SoC): typically transceiver with integrated uC**
- Crystal
  - **Reference frequency for the LO and the carrier frequency**
- Balun
  - **Balanced to unbalanced**
  - **Converts a differential signal to a single-ended signal or vice versa**
  - Impedence Matching
- Filter
  - **Used if needed to pass regulatory requirements / improve selectivity**
- Antenna



CC430 Radio


RF_P
RF_N
LNA
PA
ADC
ADC
0°
90°
Frequency Synthesizer
Demodulator
Modulator
Packet Handler
RXFIFO
TXFIFO
Interface to MSP430 core
BIAS
XOSC

CC430 – Packet Handling Support



- Built-in support for packet handling protocol:
  - Programmable # of preamble bytes
  - 8, 16, or 32 sync word bits (programmable quality thresholds)
- Flexible packet length (Variable, fixed, infinite)
- Optional address byte and CRC-filtering

CC430 Radio – TX path

- **Power Amplifier (PA)**
  - -30 dBm to +10dBm output power to 50 Ohm antenna
  - Due to built-in LDO, output power is stable across operating range (1.8V – 3.6V)
  - Output is balanced >> requires a balun at a single ended antenna
- **PA Table**
  - Used for PA power ramp up and down (ASK/OOK)
  - Cleaner RF environment when co-existing with other networks
  - PA power can be limited to the necessary minimum
    - Useful in achieving the best power consumption budgets

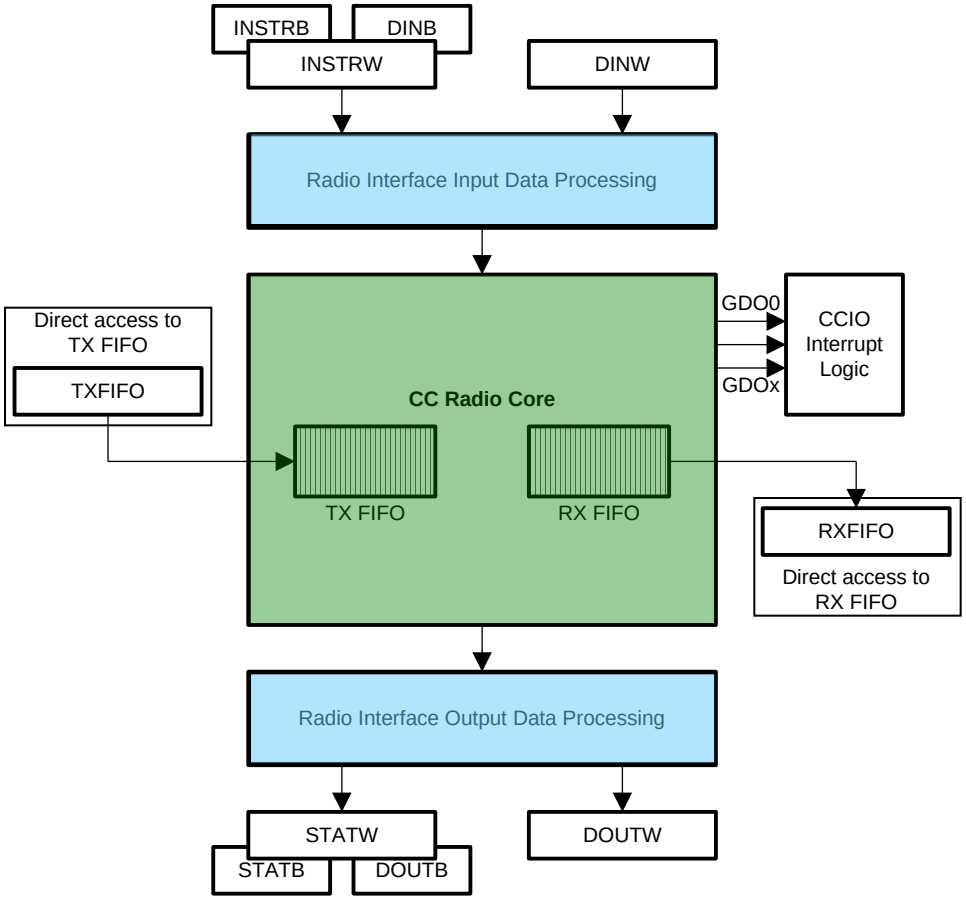
CC430 Radio – TX path

- **Synthesizer**
  - Generates the required output frequencies
  - Reference crystal frequency range 26MHz – 27MHz
  - Very high frequency resolution (397Hz – 412Hz)
- **Modulator**
  - Supports frequency modulation schemes:
    - 2-FSK, 2-GFSK and MSK
  - Supports amplitude modulation schemes:
    - ASK, OOK
  - Programmable data rates: 1.2kBaud – 500kBaud
  - Supports automatic Manchester encoding, when enabled

CC430 Radio – RX Path

- Received Signal Strength (RSSI) can be read continuously from RSSI status register until sync word is detected
  - Programmable RSSI update rate ... OR ...
  - If sync word is detected RSSI value is locked until radio re-enters RX state
- Carrier Sense asserted on one of two conditions when used as sync word qualifier
  - RSSI above a programmable absolute threshold (hysteresis included for de-assertion)
  - RSSI increase in programmable number of dB from one RSSI sample to the next
    - Useful for detecting signals with time-varying noise floor
- Carrier Sense also used for Clear Channel Assessment (CCA)

MSP430 Radio Interface Logic



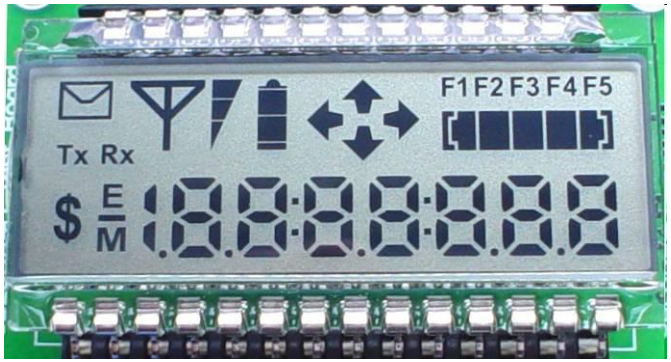
Instruction based radio controller

- **Instructions are written byte- or word-wise into INSTR register**
- **Appropriate number of data (instruction parameters) are written into DIN register**
- **Writing INSTR register initiates status update at STAT register**
- **Writing DIN register initiates output data at DOUT register**

Auto-Read feature simplifies the read accesses

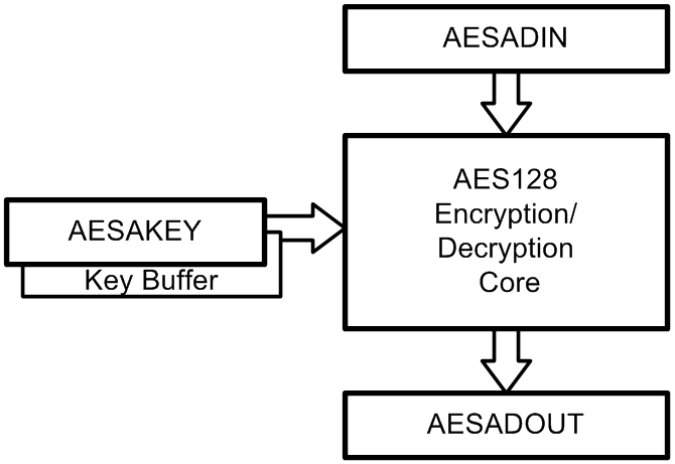
- **Saves dummy-writes**
- **Speeds up processing**
- **Simplifies usage**

LCD_B & AES128 encryption / decryption

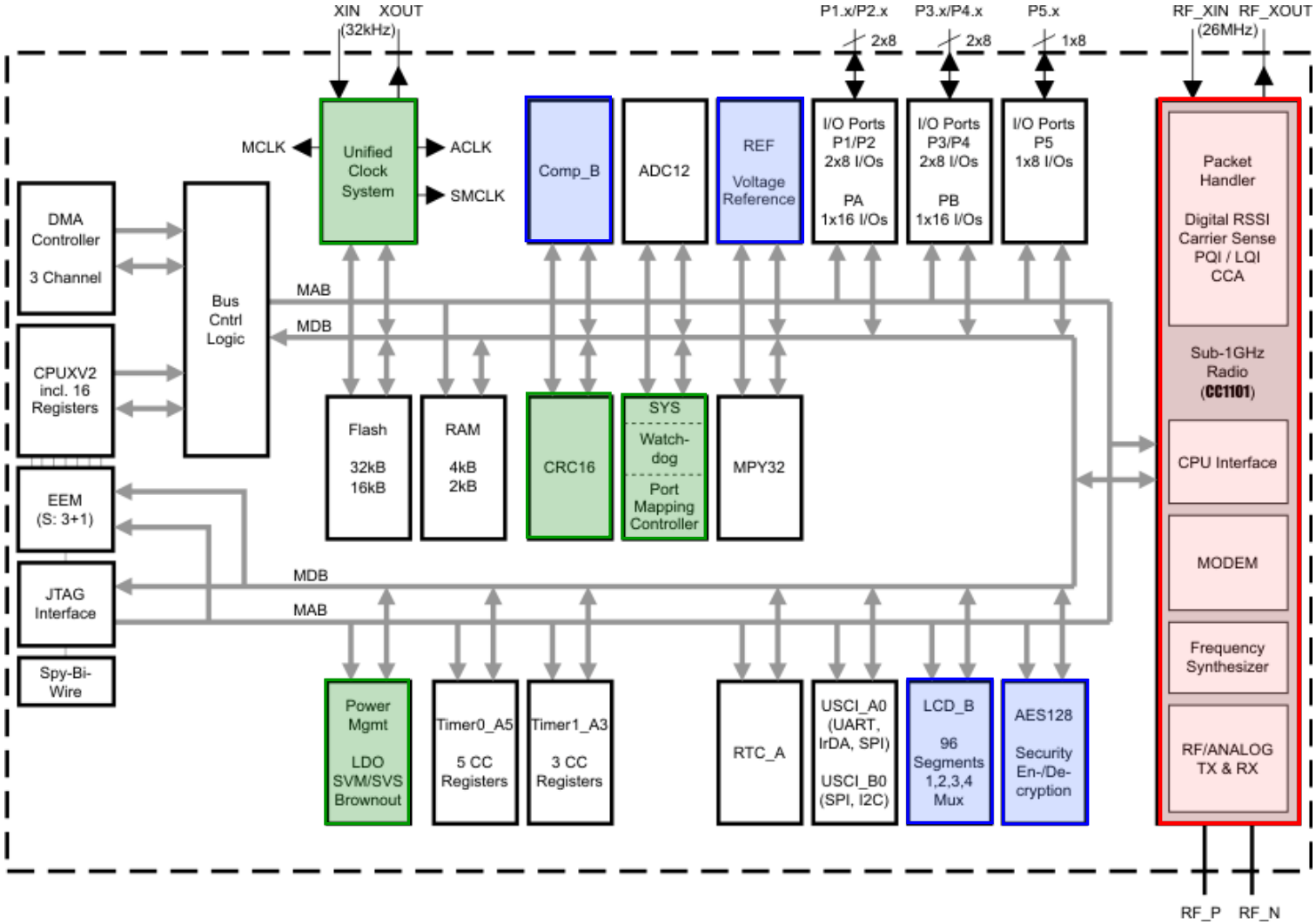


- Blinking of individual segments
- Programmable frame frequency
- Regulated charge pump
- Software-driven contrast control
- Integrated drivers to generate fractional LCD bias voltages internally

- Encryption and decryption according to AES FIPS PUB 197 with 128-bit key
- 128-bit input block size
- Byte and word access to key, input, and output data
- Off-line key generation for decryption
- AES ready interrupt flag

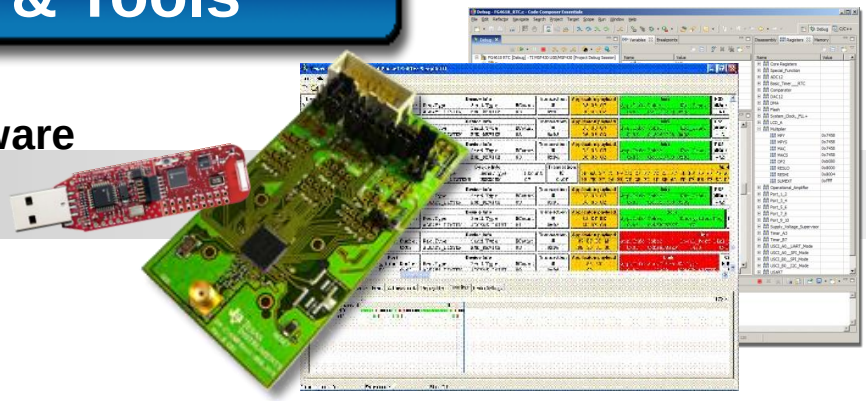


CC430F613x block diagram



Get to market fast – complete ecosystem for ease of use and fast time to-market

Software & Tools



Easy to use, low cost hardware and software tools get customers up and running fast:

RF support: RF reference designs, SmartRF Studio software, RF packet sniffer, design notes

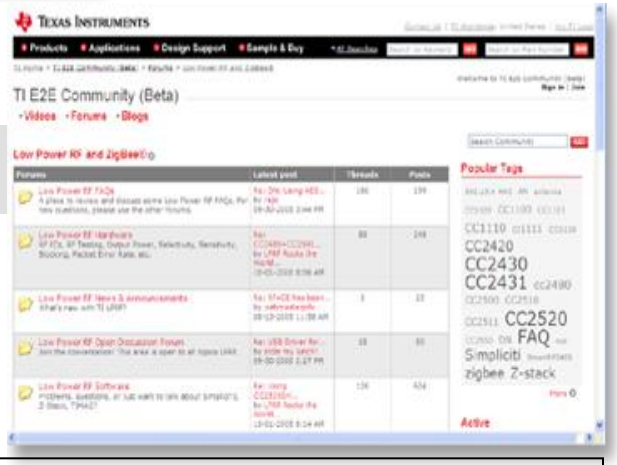
Support

Extensive community of third party and academia technology solution providers

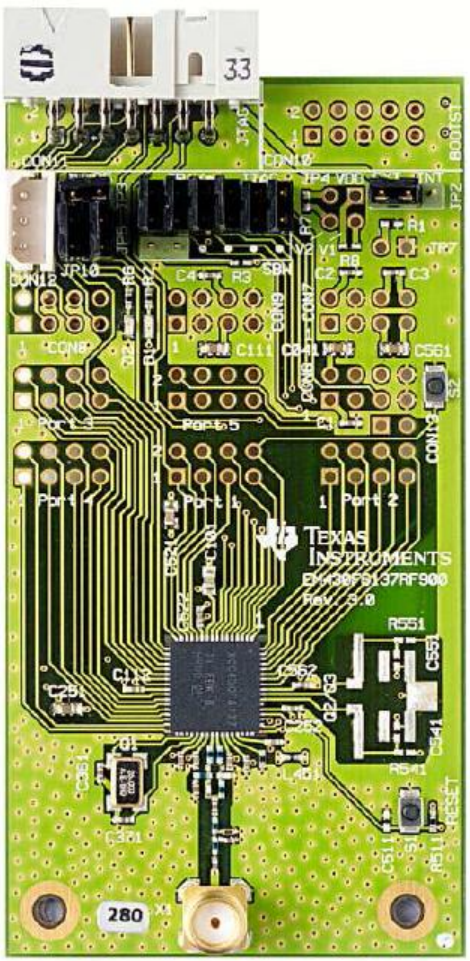
Comprehensive collateral, extensive application notes, code examples and libraries

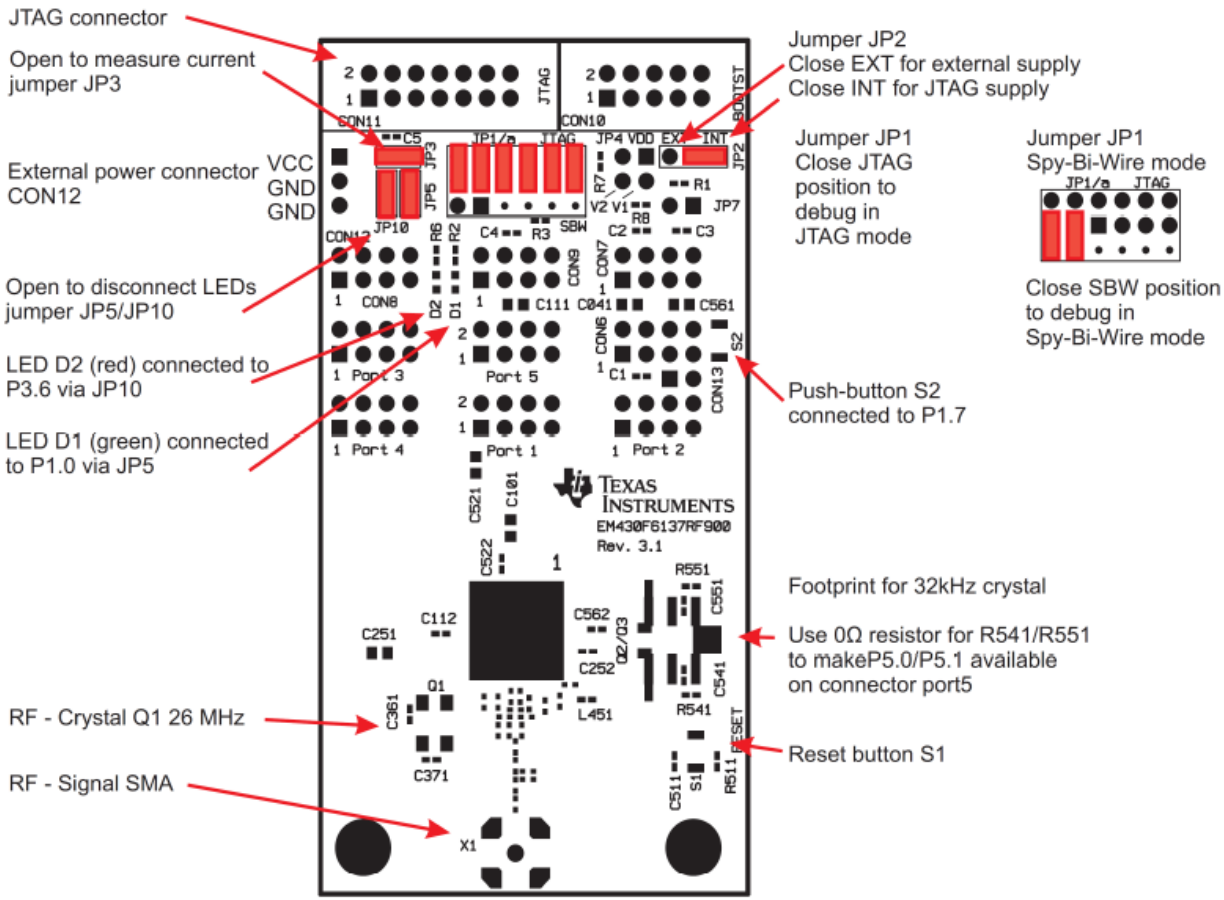
Global customer support network – TI has the most feet on the ground

World wide training options ranging from online to hands on deep dive technical training



EM430F6137RF900 Sample Kit





Chronos | Advanced Features at Your Disposal

CC430F6137 MCU

<1GHz RF
- 433, 868 & 915 MHz

2-Wire JTAG Access

96 segment LCD

Buzzer

3-Axis Accelerometer

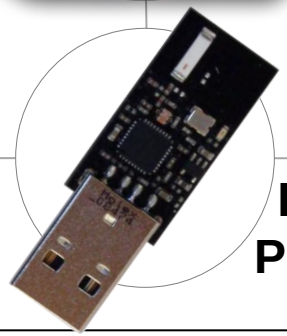
Pressure & Altitude Sensor

Temperature Sensor

Voltage & Battery Sensor

CR2032 Battery

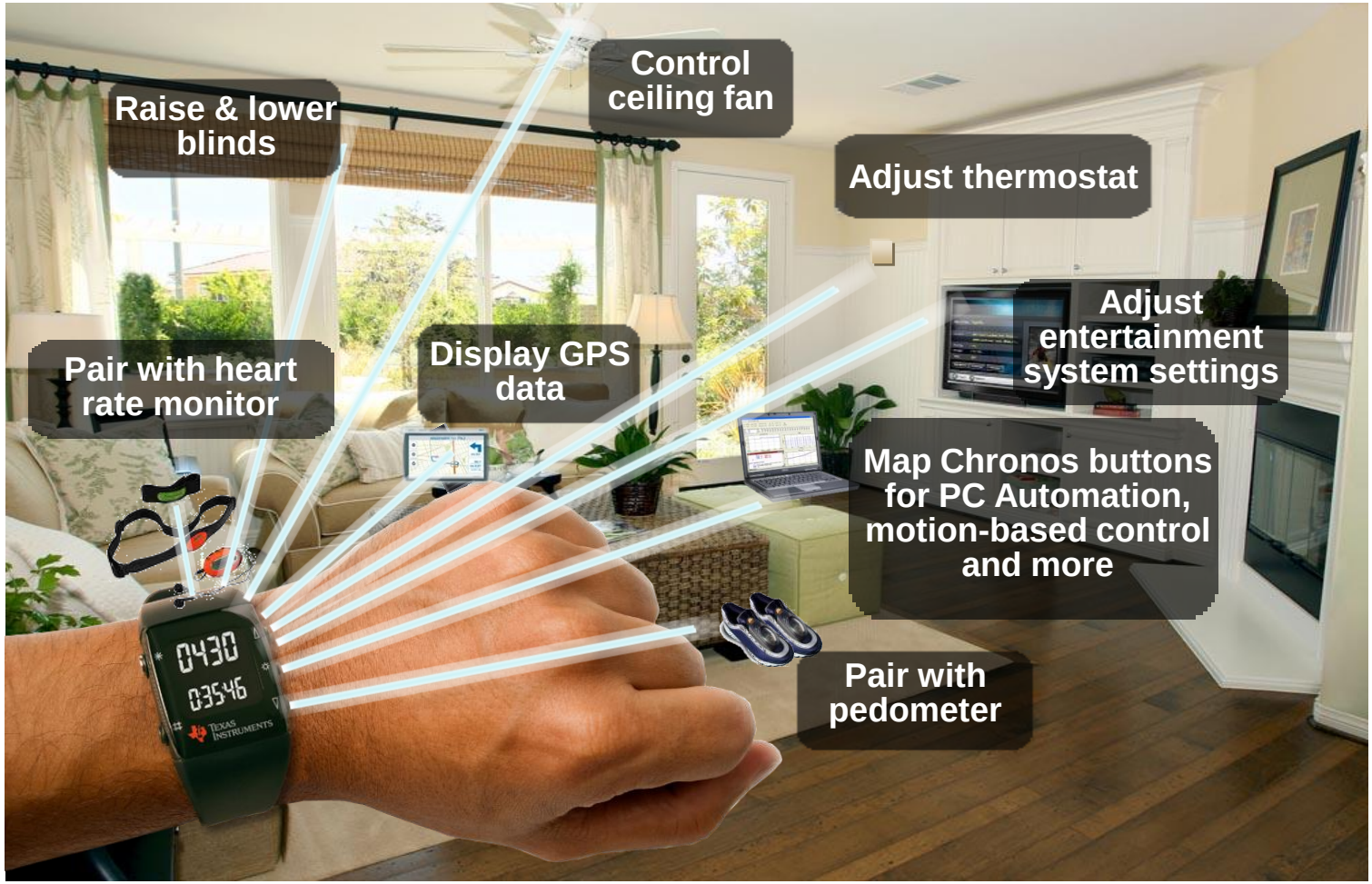
eZ430 Programmer



RF Access Point

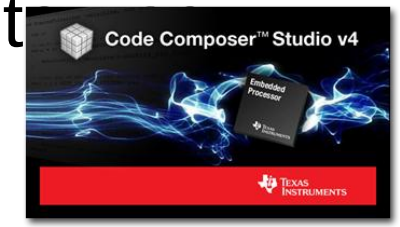
Chronos Disassembly Tool

Endless possibilities: Chronos serves as a central hub for nearby wireless sensors



Chronos | The Software Difference

- Free development software
- Production-ready, open-source projects
- Multiple easy-t RF stacks
  - SimpliciTI
  - BlueRobin
  - More coming…
- User generated apps and support on **www.ti.com/chronoswiki**

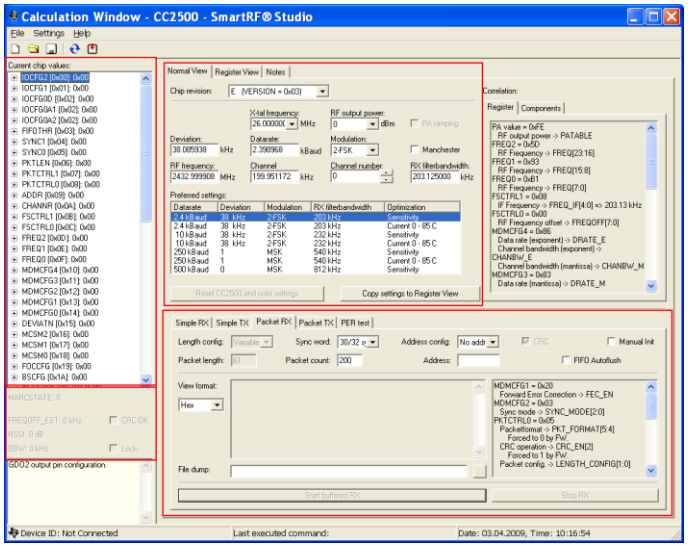


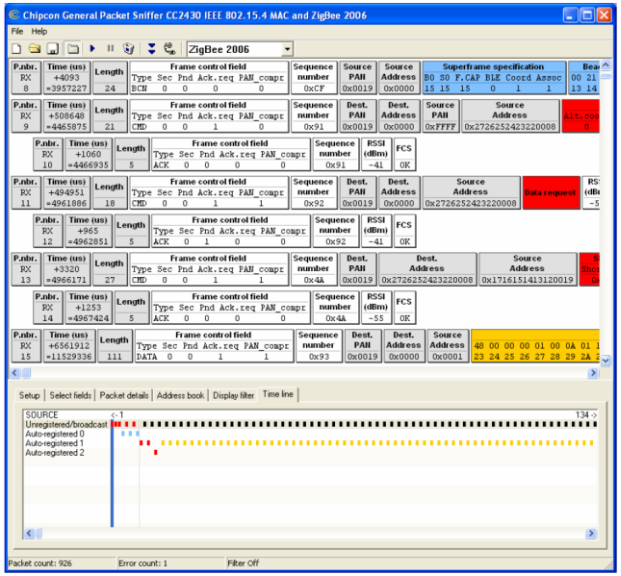
- Chronos Projects
  - Watch functions (time, date, alarm, stopwatch)
  - Fitness programs: running speed, distance, heart rate, calories burned
  - Sensor data logging w/ wireless PC download
  - PowerPoint Control
  - Media Remote
  - Motion-based mouse/PC game control
  - RF control of Beagle board game
  - Elderly fall detection*
  - Wireless App Updating*
  - Car Horsepower/G-Force Estimator*
  - Skier Speed & Altitude Logger*
  - More apps to come…

*App in works

CC430 Software Tools

- **SMARTRF Studio**
- Automatically configure your RF settings
- Test the RF front-end
  - Continuous RX / TX
  - Packet RX / TX
- No MSP430 code required!





SimpliciTI Packet Sniffer

- Analyze RF packets in real-time
- Greatly simplifies debugging
- Requires the SmartRF04 or SmartRF 05 platform

Select

Proprietary or Standard

Layer \ Solution	Proprietary	SimpliciTI	IEEE 802.15.4	RF4CE	ZigBee
Application	Design Freedom	Design Freedom	Design Freedom	Design Freedom	Design Freedom
Higher Layer Protocol	Design Freedom	Design Freedom	Design Freedom	Remo TI	Z-Stack + Simple API
Lower Layer Protocol	Design Freedom	SimpliciTI	TI MAC	TI MAC	TI MAC
Physical Layer	all LPRF devices	CC111x, CC251x, CC243x, CC253x, CC430, MSP430+CC1101, CC2500 or CC2520	CC253x CC243x MSP430+CC2520	CC253x CC243x	CC253x CC243x CC2480
RF Frequency	2.4 GHz Sub 1 GHz	2.4 GHz Sub 1 GHz	2.4 GHz	2.4 GHz	2.4 GHz

MSP430 Ultra-Low Power MCUs

BlueMSP development platform (Dual-Mode)

Features

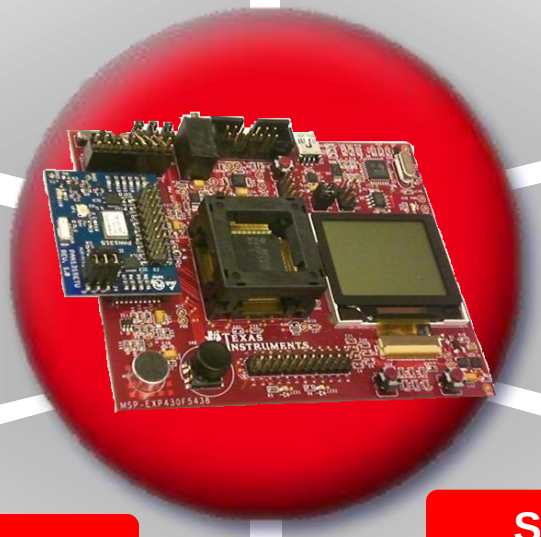
- Serial link Bluetooth applications on MSP430
- *Bluetooth* 2.1+EDR
- Bluetooth Low Energy (BLE) upgradeable
- Pre-integration into the MSP430F5438
- FM support not included

Bluetooth® Connectivity Card

- BL6450-based module
- Chip antenna
- Plugs into MSP430F5438 Experimenter Board

Schedule

- Connectivity Card available from Panasonic:
  - *Bluetooth* 2Q 2010
  - *Bluetooth* + BLE 2H 2010
- Software
  - *Bluetooth* SPP / HDP 2Q 2010
  - *Bluetooth* + BLE 2H 2010



Out of the Box

- *Bluetooth* demo applications (source code format)
  - Accelerometer-based game
  - **Continua-compliant Thermometer readings**
  - Embedded RF link tool

Platform Partners

Panasonic ideas for life

BL6450 Module Supplier

MindTree

System Integrator & *Bluetooth* / BLE Stack Supplier

Software

- Data over SPP (serial port profile)
  - Production quality licensed from TI
- Medical over HDP (health device profile)
  - Demonstration from TI
  - Product engagements via MindTree
- Bluetooth Low Energy (2H 2010)
  - Demonstration from TI
  - Product engagements via MindTree

MSP430 Value Line

The Value Line

- Ultra Low Cost product family – with volume pricing starting at $0.25
  - Maintains the MSP430 DNA (ULP, Analog Integration, Ease of Use)
- Multi-Phase launch
  - Phase 1: Launch April 2010
    - 0.5 KB – 2 KB Flash
    - Analog/communication modules: Comparator, 10-bit ADC, SPI/I2C
  - Phase 2: Multiple sub families, first samples Q3, 2010, launch Q4, 2010
    - 0.5KB - 16KB Flash
    - Optional analog/communication modules: Comparator, 10-bit ADC, SPI/I^2C/UART
  - 8 total packages (including DIP, SOIC, TSSOP, QFN)
- Pin compatibility across MSP430 Value Line family
  - Also compatible with MSP430F20xx family

Active current (1 MHz, 3V):	300 µA
Standby current (LPM3):	0.5 µA
Power down current (LPM4/5):	0.1 µA
Wake-up Time from LPM3:	1 µs

16-bit performance for enhanced accuracy and headroom for differentiation

Safety & Security

Lighting

Touch pads

Consumer Electronics

Personal Health & Fitness

Intelligent Sensors

- Lower maintenance cost and longer product life
- Headroom and intelligence for new functionality
- More precise settings and faster response time

8-bit developers no longer need to sacrifice performance, power efficiency or scalability for price

Deliver increased features and functionality with optimized 16-bit MSP430 architecture

- Up to 10X more MIPS throughput
- Reduced cycles/task
- 50% greater code density

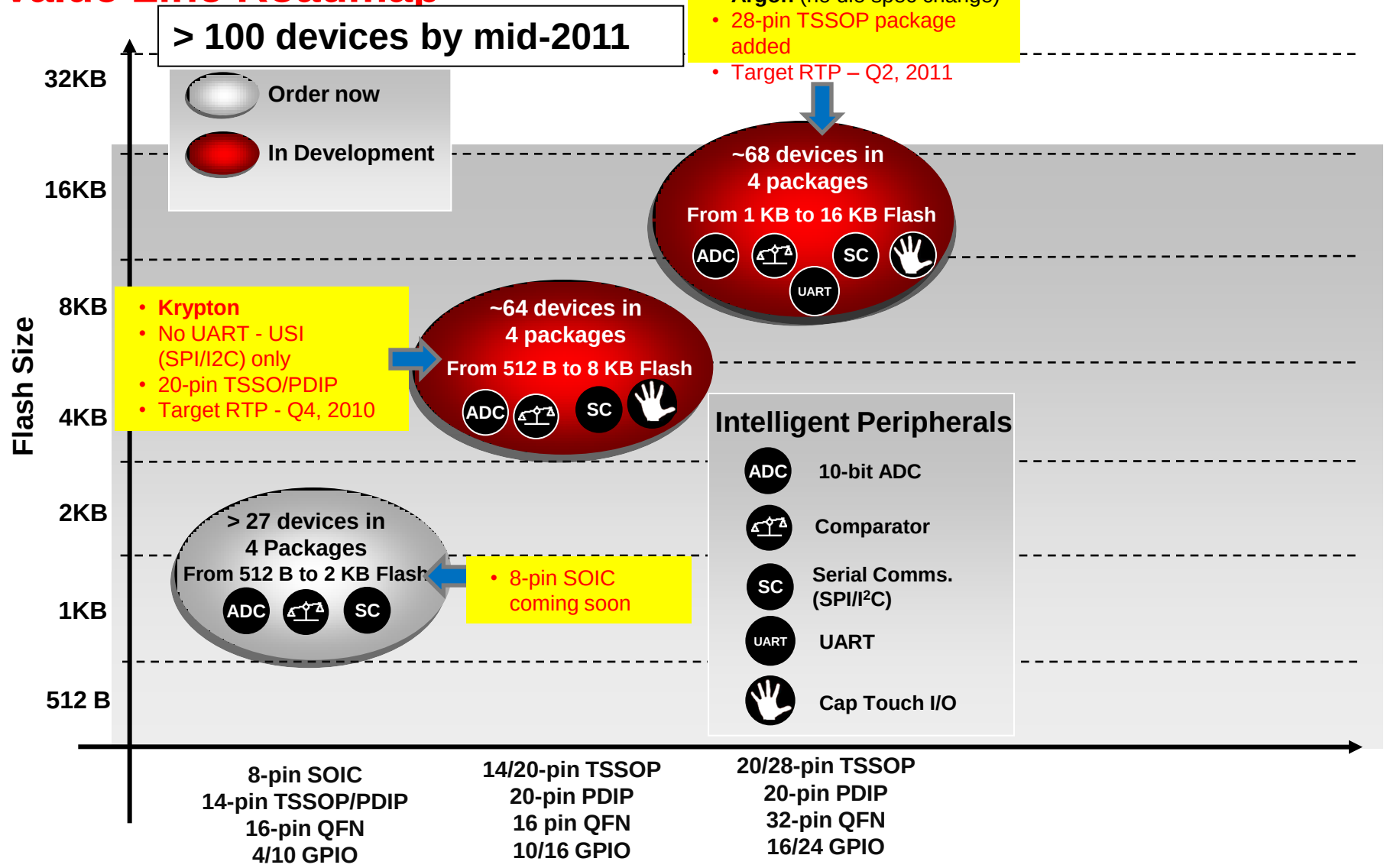
Design products with up to 20 years of battery life through leading ultra-low power consumption

- 10X lower power
- Smart ADC
- Sub 1μs wakeup

Accelerate time to market with easy-to-use tools, free software & extensive third party support

- 100 new MCUs as low as $0.25
- Full compatibility across entire MSP430 platform

Value Line Roadmap



The Value line Overview

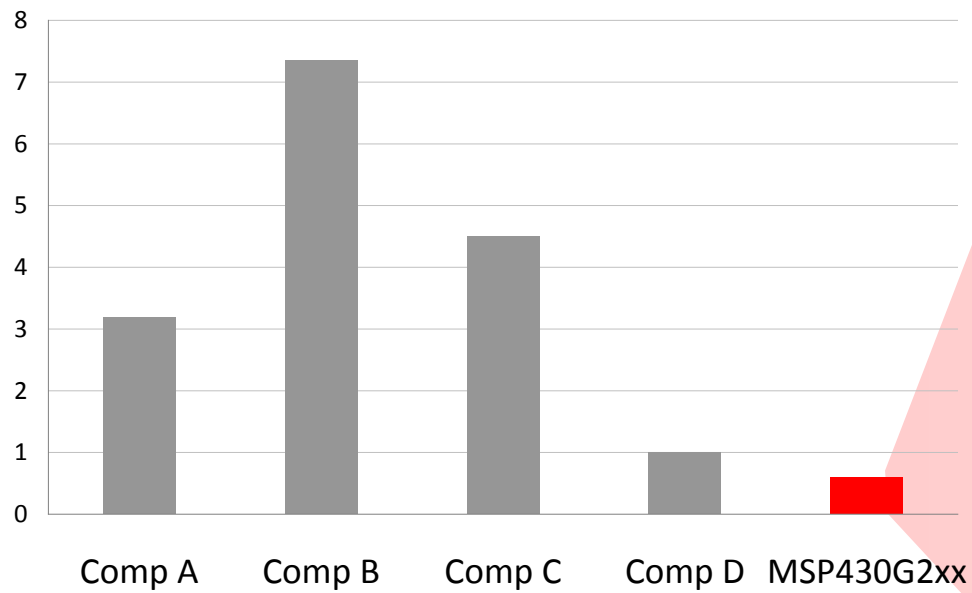
	G2x01 G2x11	G2x21 G2x31	G2xx2 (Krypton)	G2xx3 (Argon)
Package Options	14-pin TSSOP/PDIP 16-pin QFN 8-pin SOIC	14-pin TSSOP/PDIP 16-pin QFN 8-pin SOIC	14-pin TSSOP 16-pin QFN 20-pin TSSOP, PDIP	20-pin TSSOP, PDIP 28-pin TSSOP 32-pin QFN
Max Flash Memory in KB	2	2	8	16
ISP	Yes	Yes	Yes	Yes
Max RAM (B)	128	128	256	512
Clock (DCO with calibration)	VLO, DCO,LF	VLO, DCO,LF	VLO, DCO,LF	VLO, DCO, LF
Max Performance	16 MHz	16 MHz	16 MHz	16 MHz
COMP_A+	Yes	No	Yes	Yes
ADC10 (T-Sense, Batt. Monitor, Vref and T-Sense calibration)	No	Yes	Yes	Yes
Timer	1x TA2, WDT, BOR	1x TA2, WDT, BOR	1x TA3, WDT, BOR	2x TA3, WDT, BOR
EEM, DMA, MPY, SVS	EEM, -, -, -	EEM, -, -, -	EEM, -, -, -	EEM, -, -, -
SPI/I^{2}C/UART		USI (SPI/I^{2}C)	USI (SPI/I^{2}C)	USCI (SPI/I^{2}C/UART)
BSL (Boot-strap-loader), Spy-bi-wire	No, SBW	No, SBW	No, SBW	Yes, SBW

TI Confidential – NDA Restrictions

Leading ultra-low power consumption drives longer battery life

Typical battery powered applications spend 99-99.9% of their time in standby mode

Standby Mode comparison @ 3V (uA)



G2xx power consumption is lower than typical 8-bit MCUs; ULP, as low as:

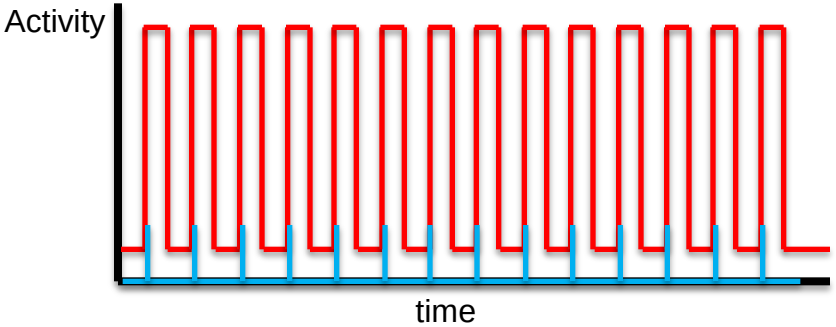
- 0.1 µA RAM retention
- 0.4 µA Standby mode (VLO)
- 0.7 µA real-time clock mode
- Ultra-Fast Wake-Up From Standby Mode in <1 µs

***10*X lower power**

Intelligent Peripherals – ADC10 + DTC

Competitor A

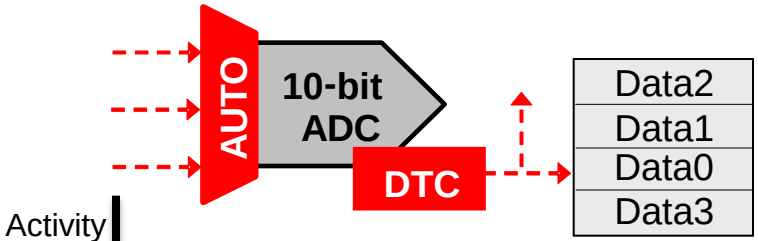
The competition requires the CPU to be active for each ADC conversion and loaded data transfer.

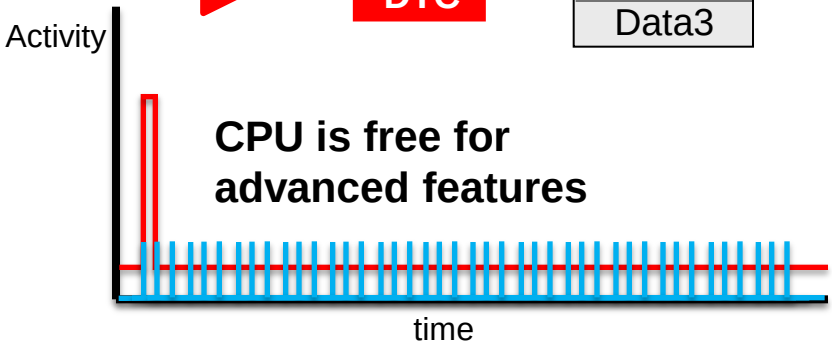



```
Competitor A
                movlw       0x20
                movwf       FSR
                movlw
                movwf
Main            movf
                movwf
                decf
                btfss       STATUS
                mov         0x03
Wait            bsf         ADCON0, GO
                btfss       ADCON0, GO
                goto        Wait
                movf        ADCRESH, W
                movwf       INDF
                incf        FSR, F
                bcf         STATUS, 0x20
                movwf       INDF
                incf        FSR, F
                btfss       STATUS, Z
                goto        main
```

Limited to 10,638 samples/s @ 100% CPU load

MSP430+DTC – Do more!






```
MSP430 + DTC
 bis.w  #CPUOFF,SR
```

Up to 200,000 samples/s @ 0.6% CPU load

MSP430's ADC10 with Data Transfer Controller (DTC) can manage ADC samples throughout entire memory range – All without CPU overhead!

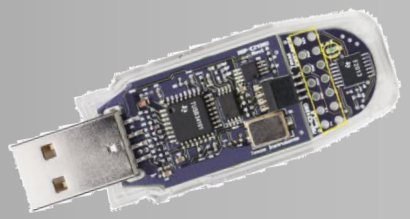
- Less time in Active Mode saves power
- CPU available for other more advanced tasks
- Intelligent autonomous sampling
- Deterministic behavior

Texas Instruments

Powerful, low cost development tools

eZ430-F2013: $20 Development System

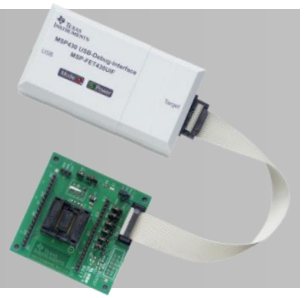
- Real-time, in-system emulation
- Removable target with full pin access



Free software IDEs

- Code Composer Studio™ v4
- IAR Embedded Workbench

- EZ430-F2013 ($20) = eZ430 + F2013 target board
  - EZ430-T2012
- MSP-FET430U14 ($149) = FET + F20xx Compatible Target board
- MSP-TS430PW14 ($75) = F20xx Compatible Target board



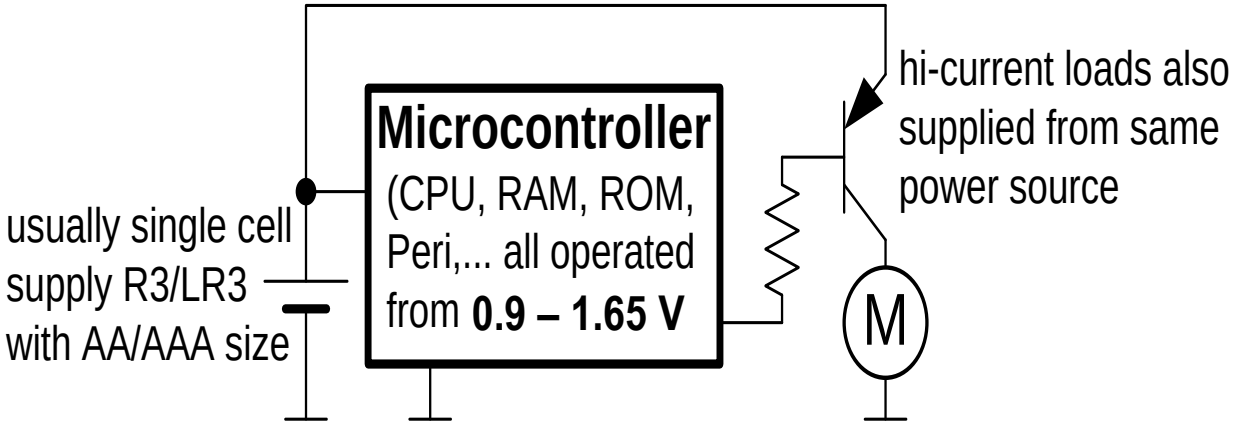
Your development tools – only $20

MSP430 Future Products / Technologies

Native
Sub 1V

Native 0.9V Operation

- **Native Means…**
- The device, including peripherals operates at specified V_{cc}
- Additional components such as charge pump are **NOT** needed to support the full V_{cc} range
- Full functionality of the device is maintained within V_{cc} range
- The microcontroller and application are supplied from one common supply that goes down to min V_{cc}



Native
Sub 1V

Native 0.9V Target Applications

- General purpose low-cost MCU
- Consumer
  - Electric tooth brush, shavers, etc.
  - Toys
  - Games
- Security devices (door sensors, passive I/R)
- Sports applications
- Sensors (motion, pressure)

FRAM: The Future of MCU Memory

Photo: forums.wow-europe.com

Non-Volatile – retains data without power

Fast Write / Update – RAM like performance. Up to ~ 50ns/byte access times today (> **1000x** faster than Flash/EEPROM)

Low Power - Needs 1.5V to write compared to > **10-14V** for Flash/EEPROM → no charge pump

RAMTRON
Automotive F-RAM Memory

Superior Data Reliability - **'Write Guarantee'** in case of power loss and > **100 Trillion** read/write cycles

FRAM: Proven, Reliable, Better

- Endurance
  - Proven data retention to 10 years @ 85°C
- Secure
  - Fast access times
  - No charge pump
  - No perceptible difference in read/write processes
- Radiation Resistance
  - Terrestrial Soft Error Rate is below detection limits
- Immune to Magnetic Fields
  - FRAM does not contain iron

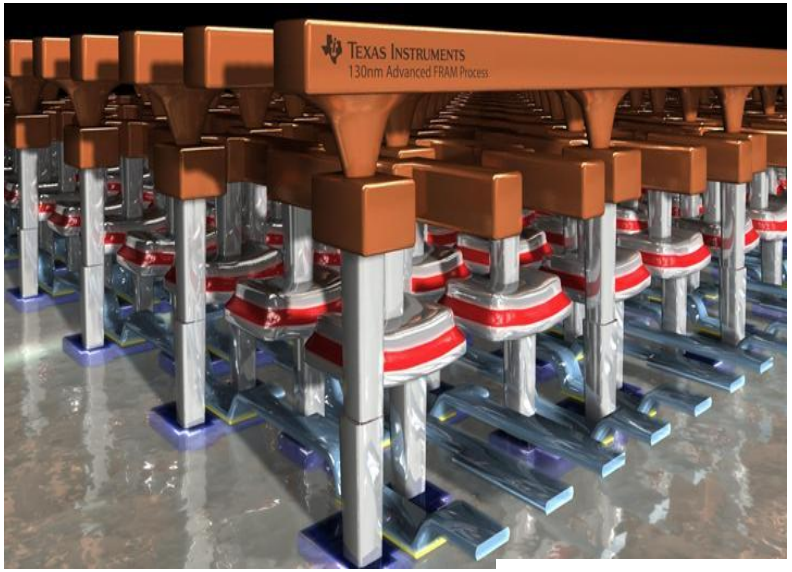


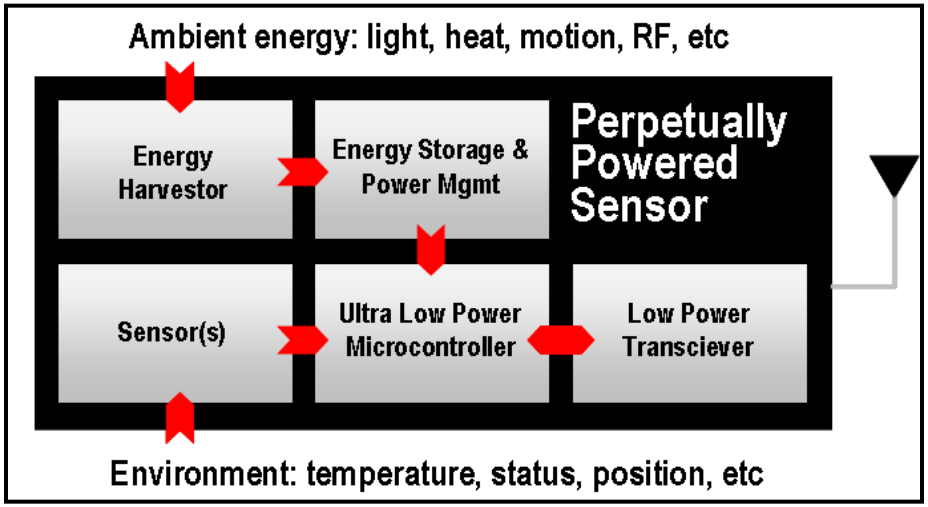
Photo: Ramtron Corporation

www.ti.com/fram

For more info on TI's FRAM technology

MSP430 for Energy Harvesting "Self or *Perpetual - Power*" Apps

- **Energy harvesting is the process by which energy is captured and stored**
- **Can substitute batteries that are costly to maintain and can extend system uptime**
- **Only possible with ultra-low power components**
- **Solar, kinetic, thermal, RF, salinity gradients, pH difference and other ambient sources available**



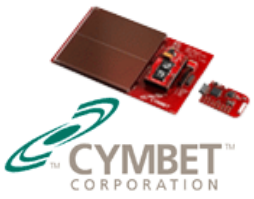
Body worn monitoring devices powered by body heat, movement

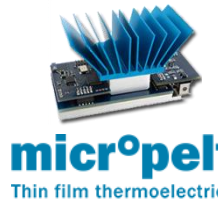
Monitor conditions on farm, winery, etc.

Mesh networking for environmental monitoring (e.g. forest fire detection)

Automotive monitoring (e.g. tire pressure gauges powered by vibration)

Energy Harvesting Made Easy – MSP430 Solutions





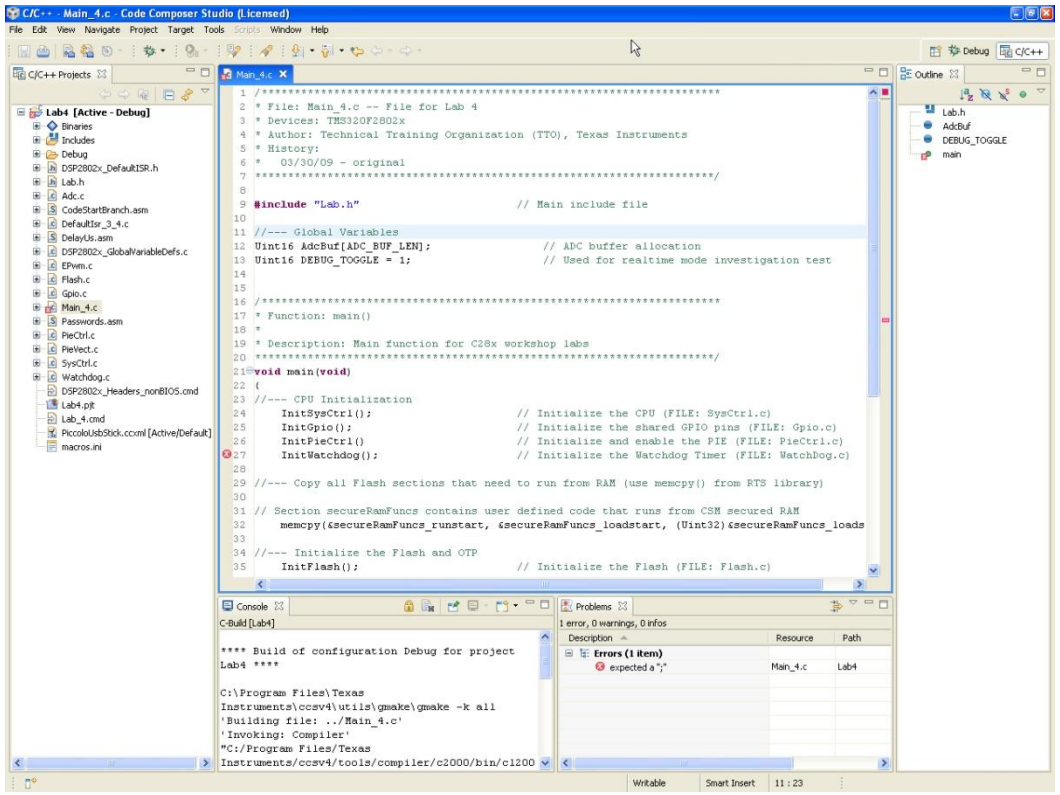
MSP430 - Introduction to Code Composer 4 Integrated Development Environment

Agenda

- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

Overview

- CCS 4 is the integrated development environment (IDE) for TI's Digital Signal Processors (DSP), microcontrollers and application processors.



- Source code editor
- Project environment
- Compiler
- Debugger
  - Software (simulator)
  - Hardware (emulator)
- Graph windows
- Image visualization
- Profiler
- Scripting capability
- Flash programmer

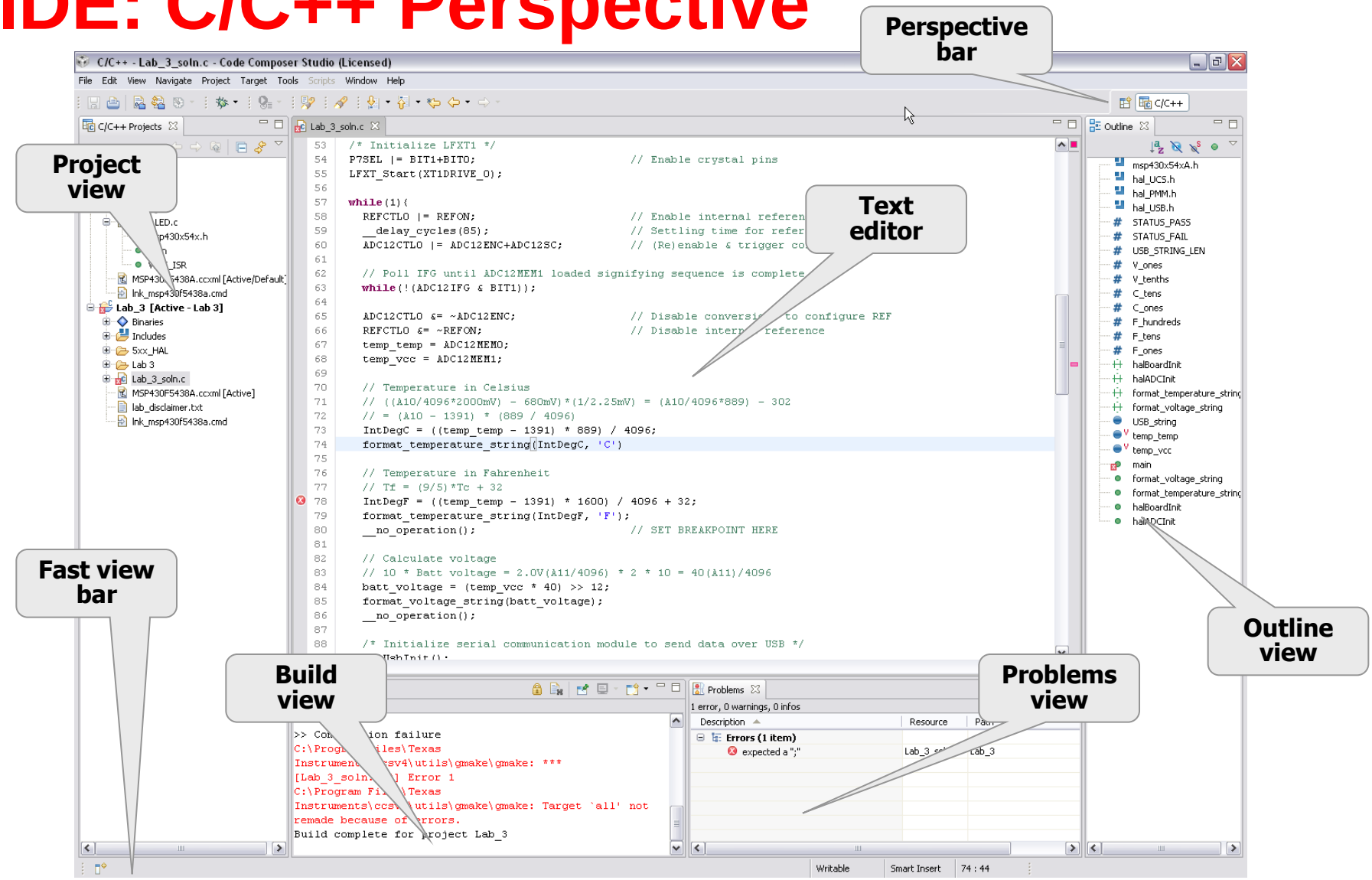
Available Versions

- Microcontroller Edition, which supports
  - MSP430
  - TMS320C2800
- Platinum Edition, which supports
  - MSP430
  - Stellaris
  - TMS320C2800
  - ARM7, ARM 9, ARM 11
  - ARM Cortex M3, ARM Cortex R4, ARM Cortex A8
  - TMS470,
  - TMS570
  - TMS320C5500
  - TMS320C6000

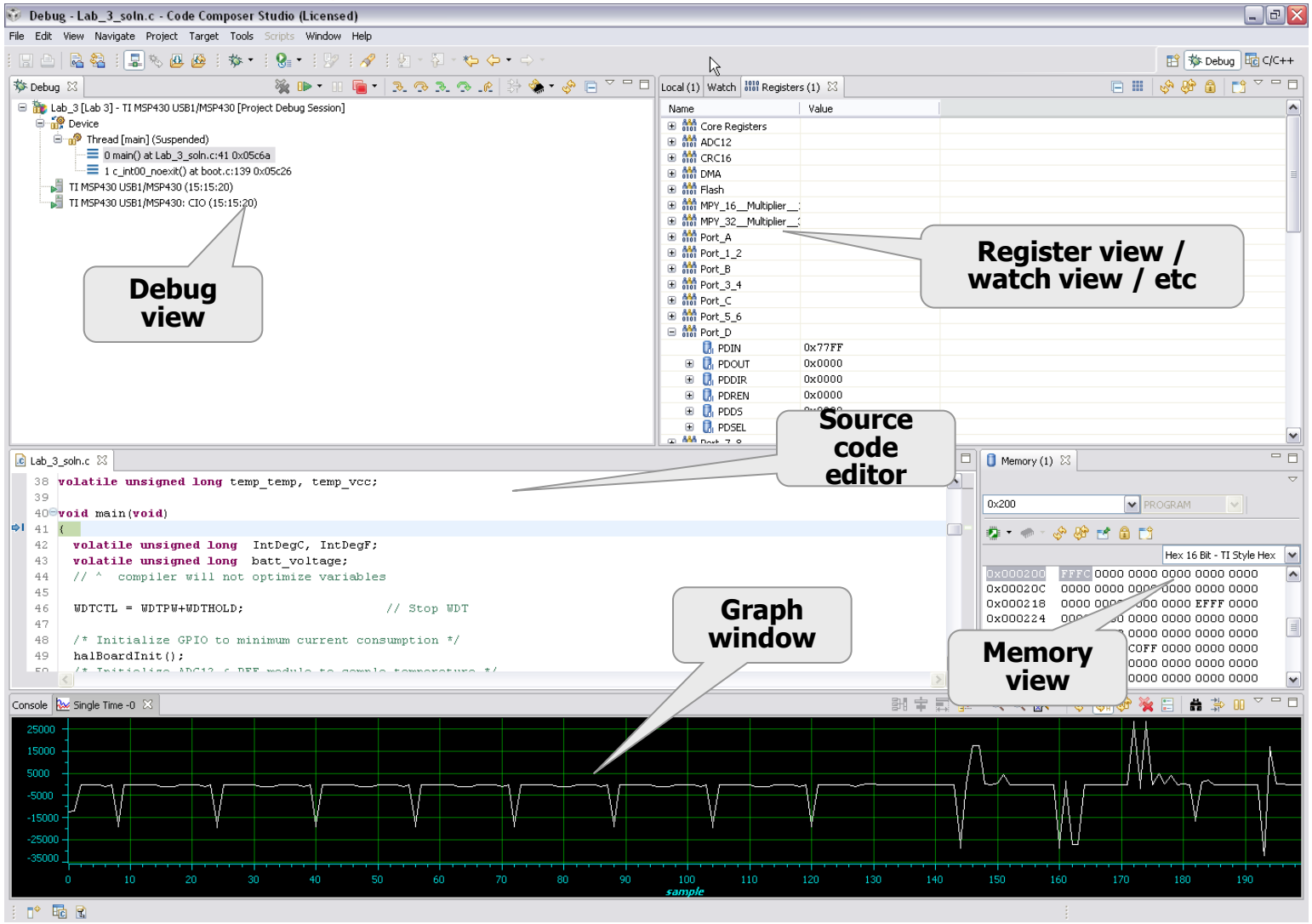
Licensing Options

- Evaluation license
  - Full version for 120 days (30 days plus 90 days extension)
- Free version
  - Tied to XDS100 emulator
- Node locked
  - Tied to one PC
- Floating
  - Install on multiple PCs
  - Use on a single PC

IDE: C/C++ Perspective



IDE: Debug Perspective

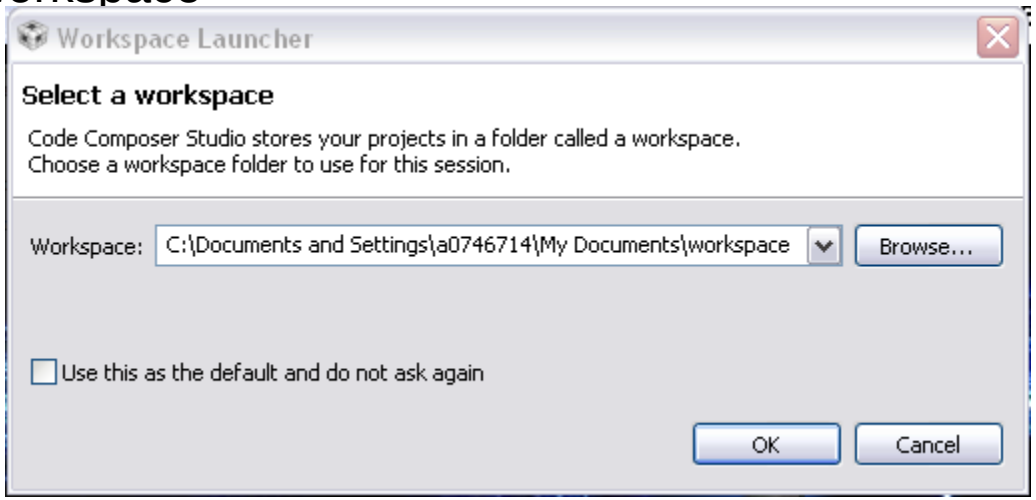


Agenda

- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

Starting a project

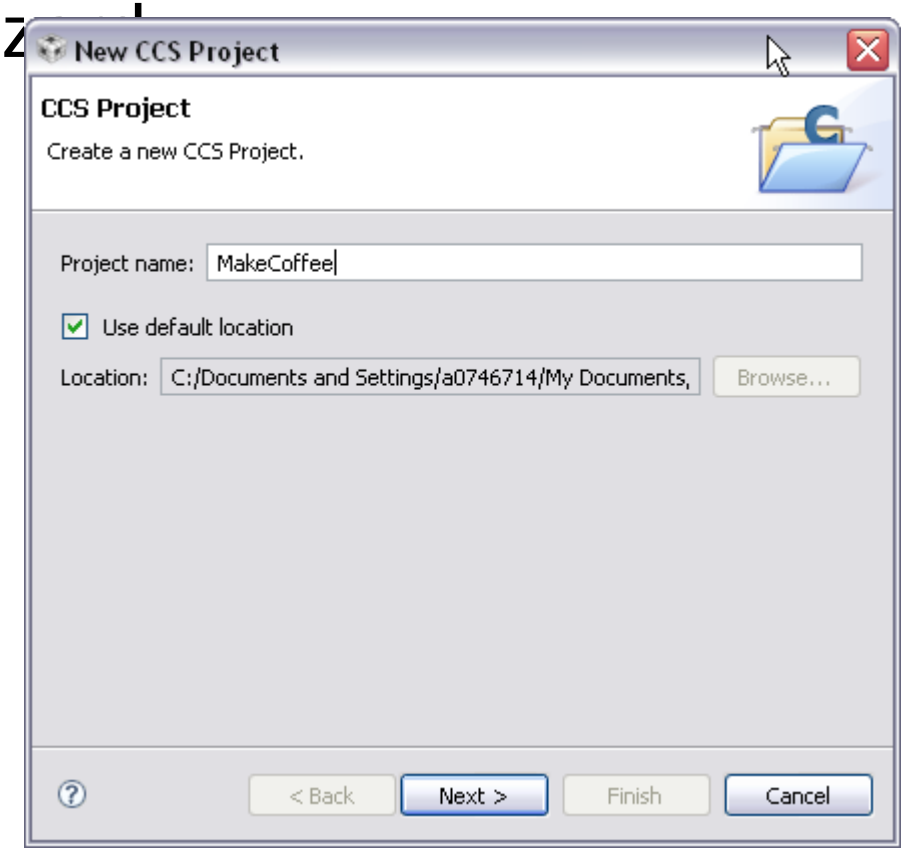
- Select the workspace



- Defines the location of your project
- Contains:
  - Project settings
  - Macros
  - Views on close
  - Other custom settings specific for YOUR personal computer
- Automatically saved on close
- Cannot be zipped and forwarded to others
- Can be changed later on

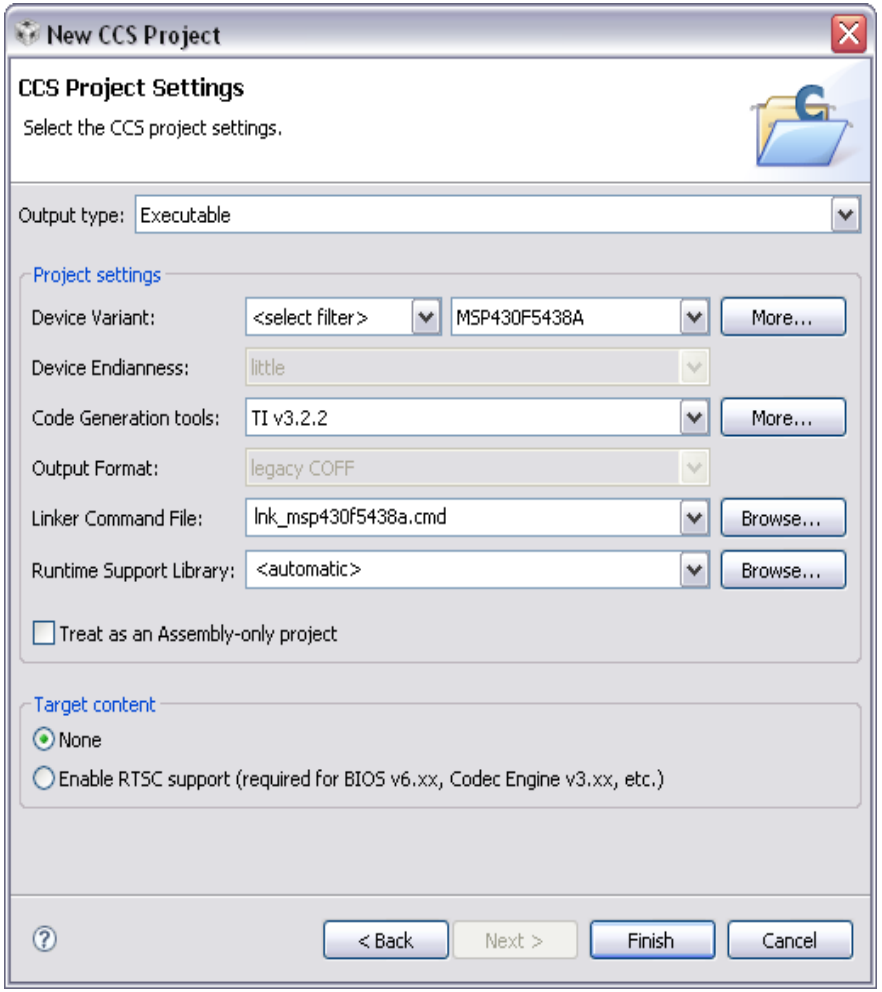
New Project Wizard

- New projects are created by using the “New Project Wizard”



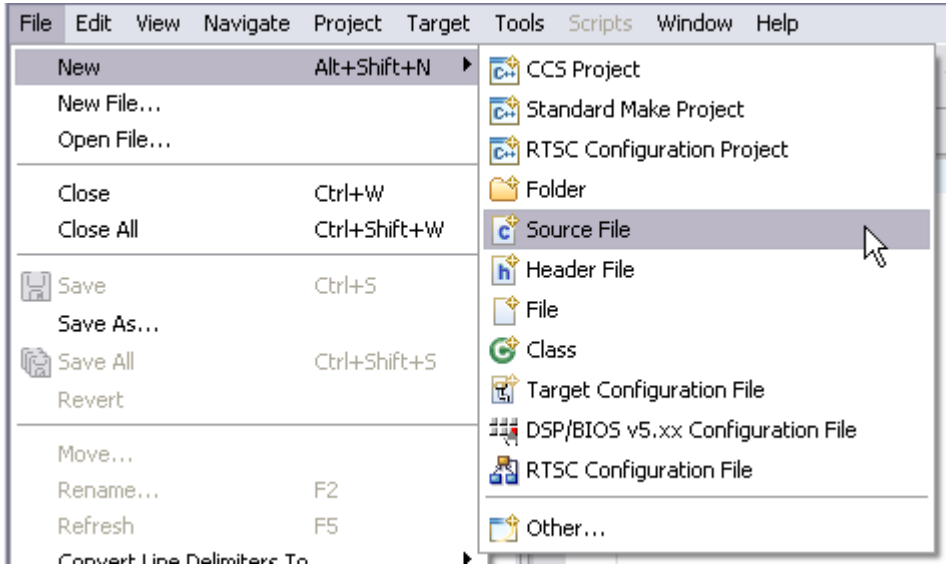
New Project Wizard

- This will allow to select:
  - The project type
    - Processor used: e.g. C2000, C6000, Stellaris, etc
  - The configurations
    - Default are
      - Debug
      - Release
  - Referenced projects (if any)
  - The used C/C++ indexer
    - Full
    - Fast
    - None
  - Output type
  - A default linker command file
  - Various project settings
    - Device variant
    - Code Generation tools version
    - Runtime Support Library
    - Target content



Adding Files

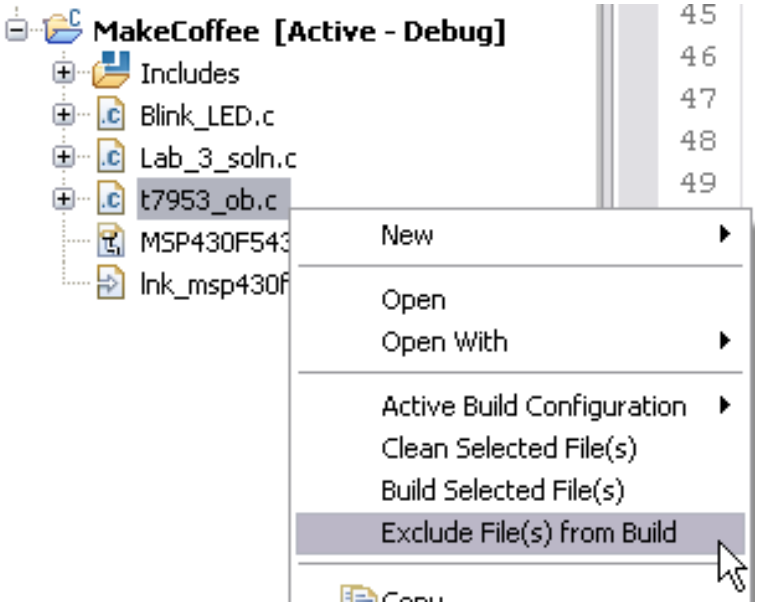
- There are two ways to add files to the project:
  - Create a new source file



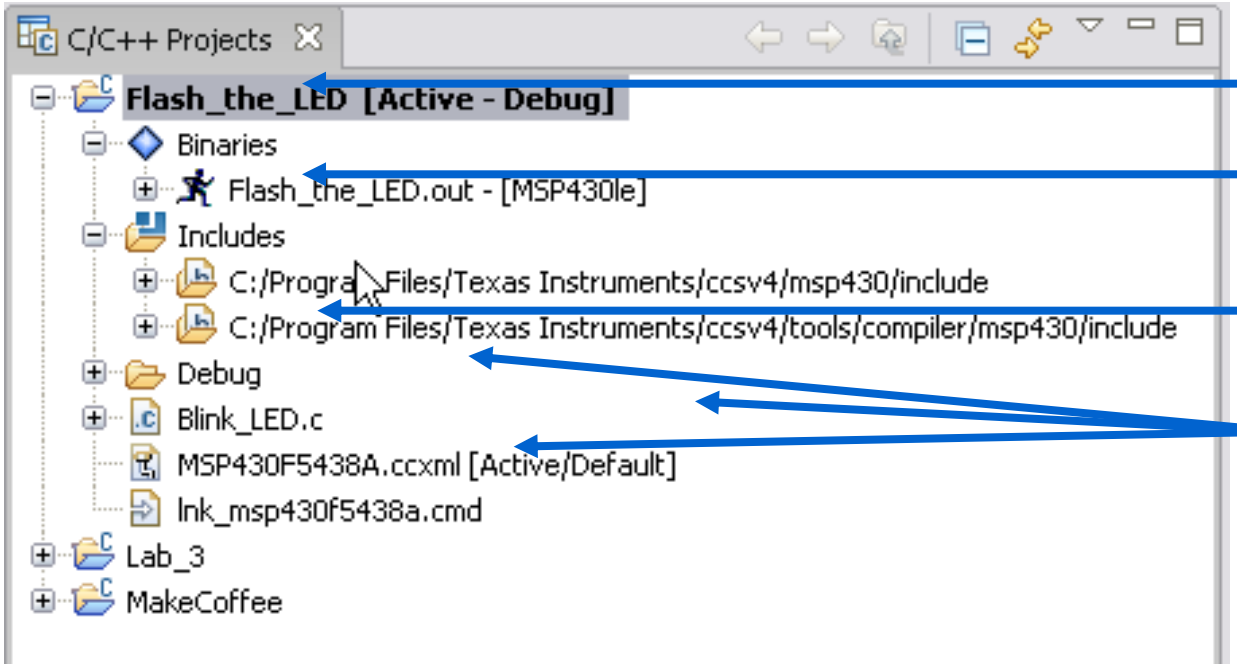
  - Drop existing source files in the project directory
    - All files in the directory of a project are **automatically** added to the project.
    - Take care that you are not adding unwanted content!

File System

- All files residing on the HDD in the project's directory will be automatically included in the project.
- All included files will be used for a build unless excluded explicitly!
  - Make sure only wanted content is in the directory
  - Exclude all needed / wanted files, which shouldn't be included in the build.



Project View



Executable output file

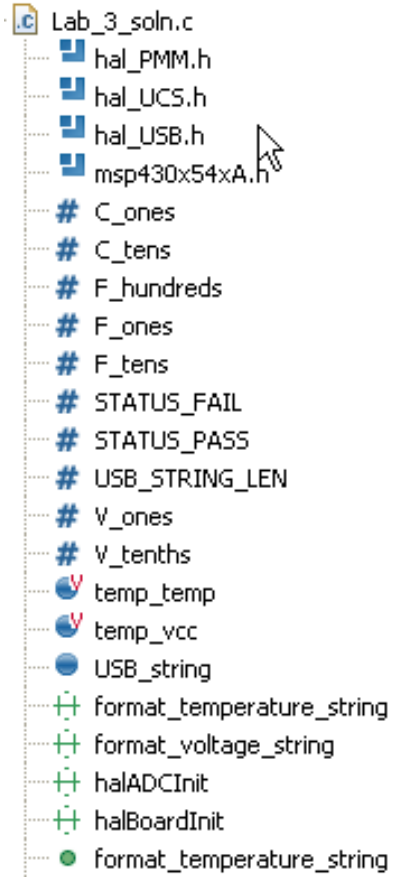
Referenced include dirs

Object- and makefiles for current configuration

Source- and command files

C Language Symbols

- Files are automatically parsed and variables, declaration and definitions displayed. These can be accessed from the outline view or from the project view

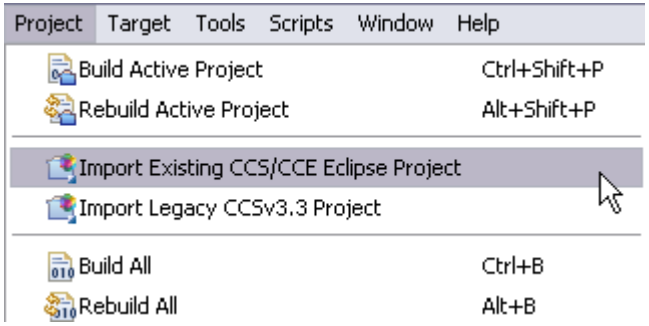


Agenda

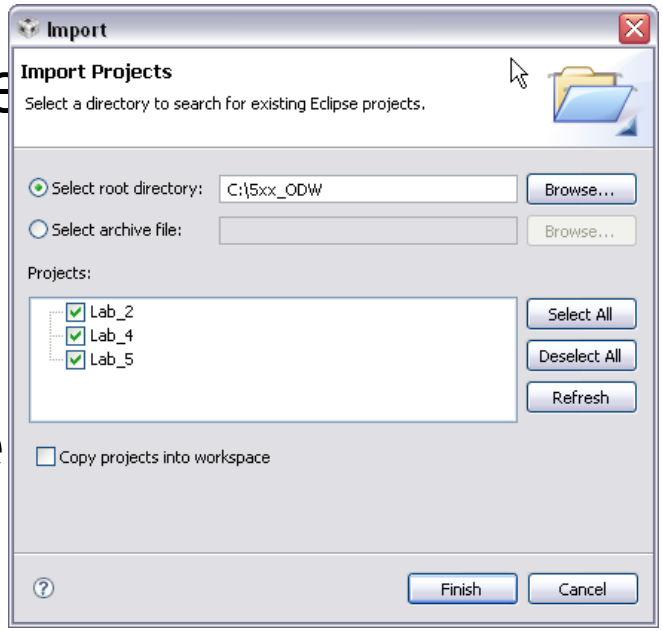
- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

Importing an Existing Project

- Accessed from

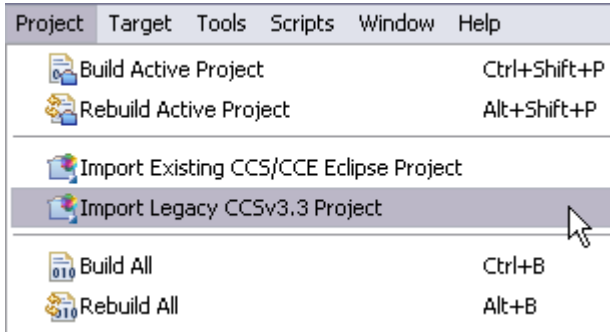


- You can import a single proje projects in one step:
  This will also give you the opportunity to copy the projects into your workspace



Importing a CCS 3.3 Project

- The CCS 3.3 import wizard is called from:

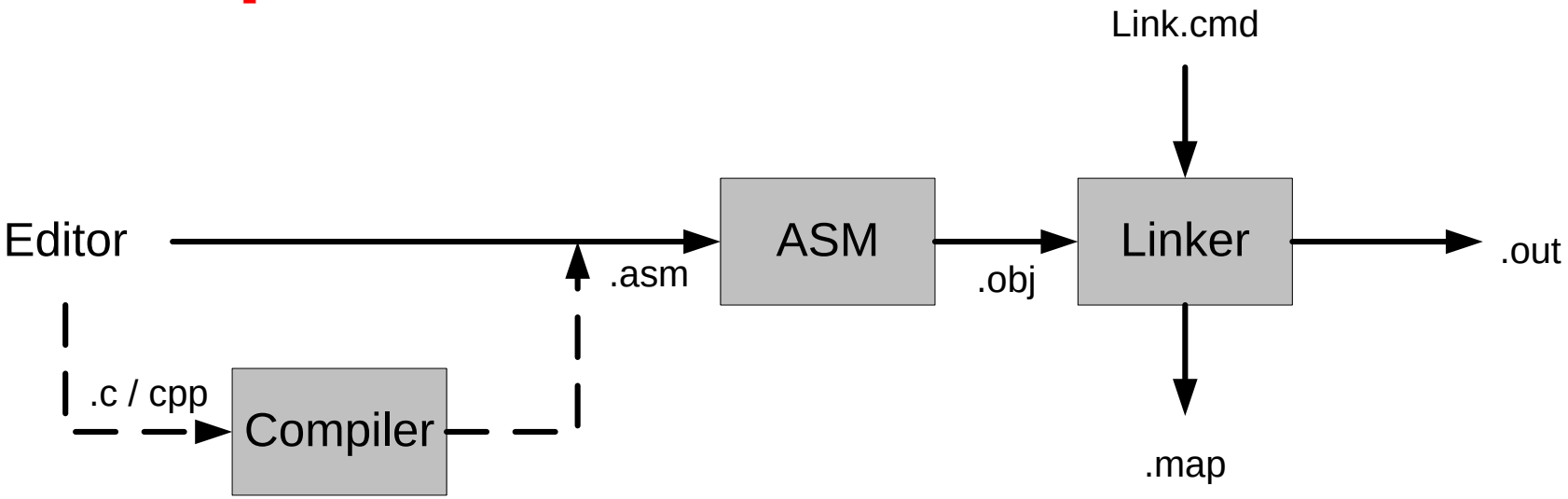


- This will help you to import the existing project, but manual re-work might still be required
  - Change or add path names for library functions
  - Change of .cdb files
- Notes:
  - CCS 3.1 projects cannot be imported automatically. They need to be converted to a CCS 3.3 project first (using CCS 3.3)
  - You can import multiple projects at the same time.

Agenda

- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

Compile Flow



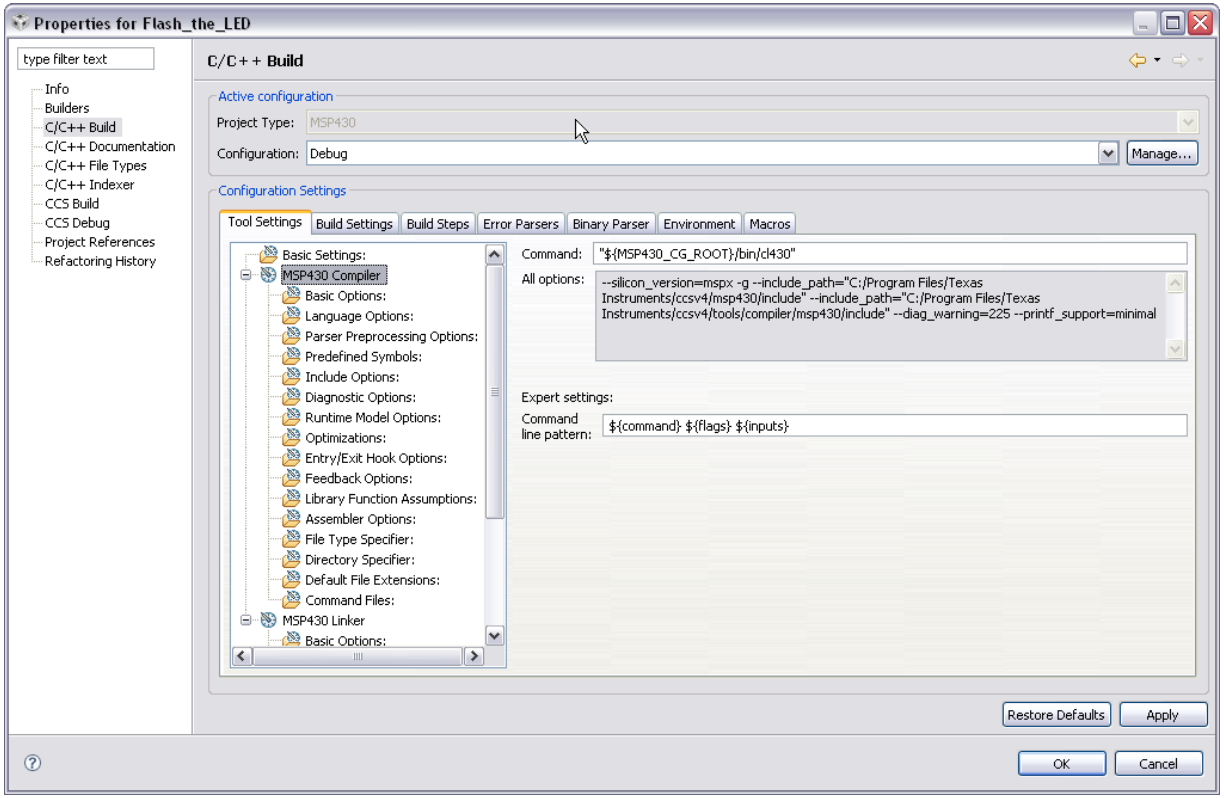
- You can either write standard assembly directly, or it can be created by the compiler.
- Object files, created by the assembler, are linked together to create the executable output file of the processor. The map file is an output report of the linker.
- The .out file can be loaded into your system by the debugger portion of CCS.

Build Configurations

- By default, CCS includes two sets of build options:
  - Debug
  - Release
- Users can create any number of additional build configurations.
- Users can modify the settings of any pre-defined build option.

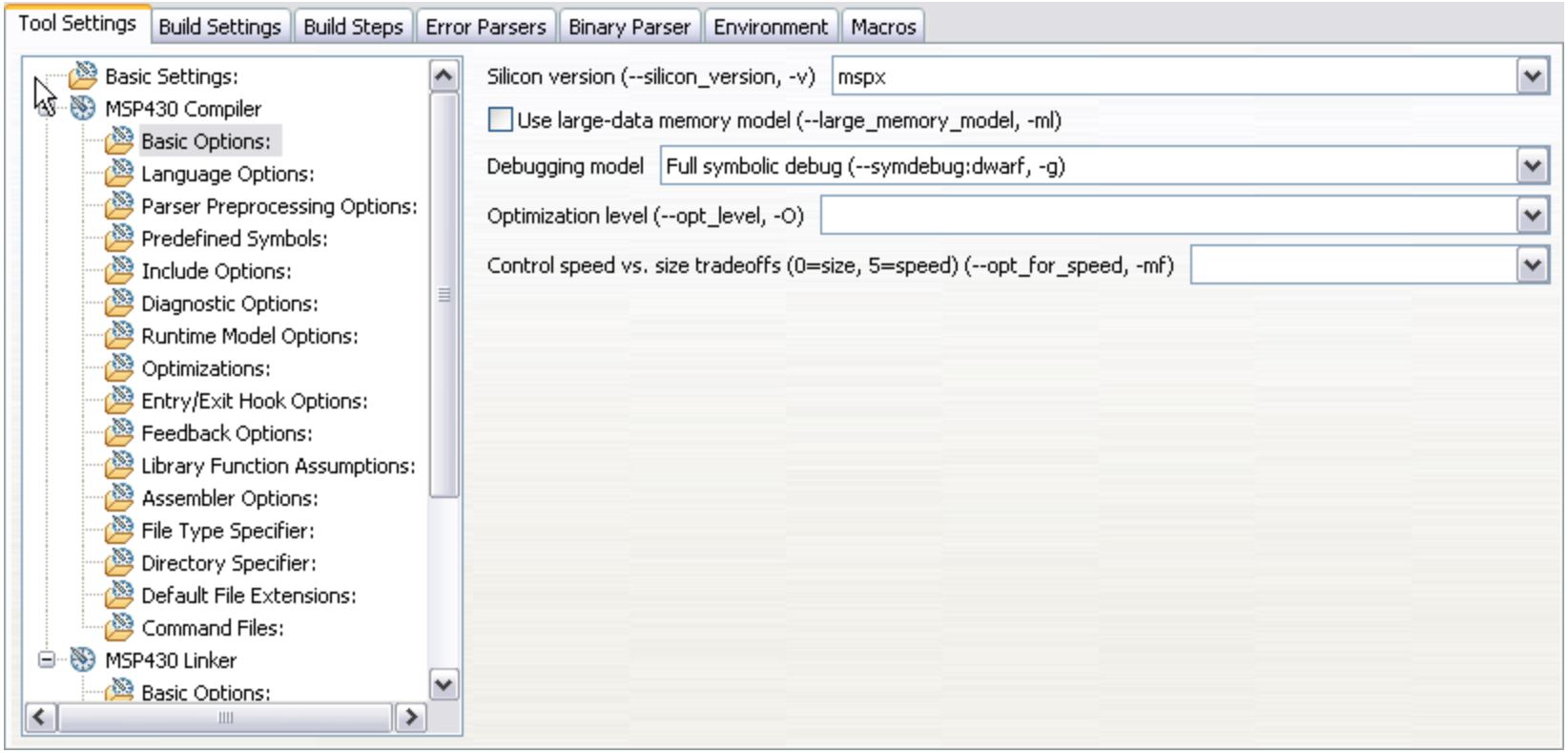
Build Options

- Build options are controlled by the “Properties” dialog of the project:

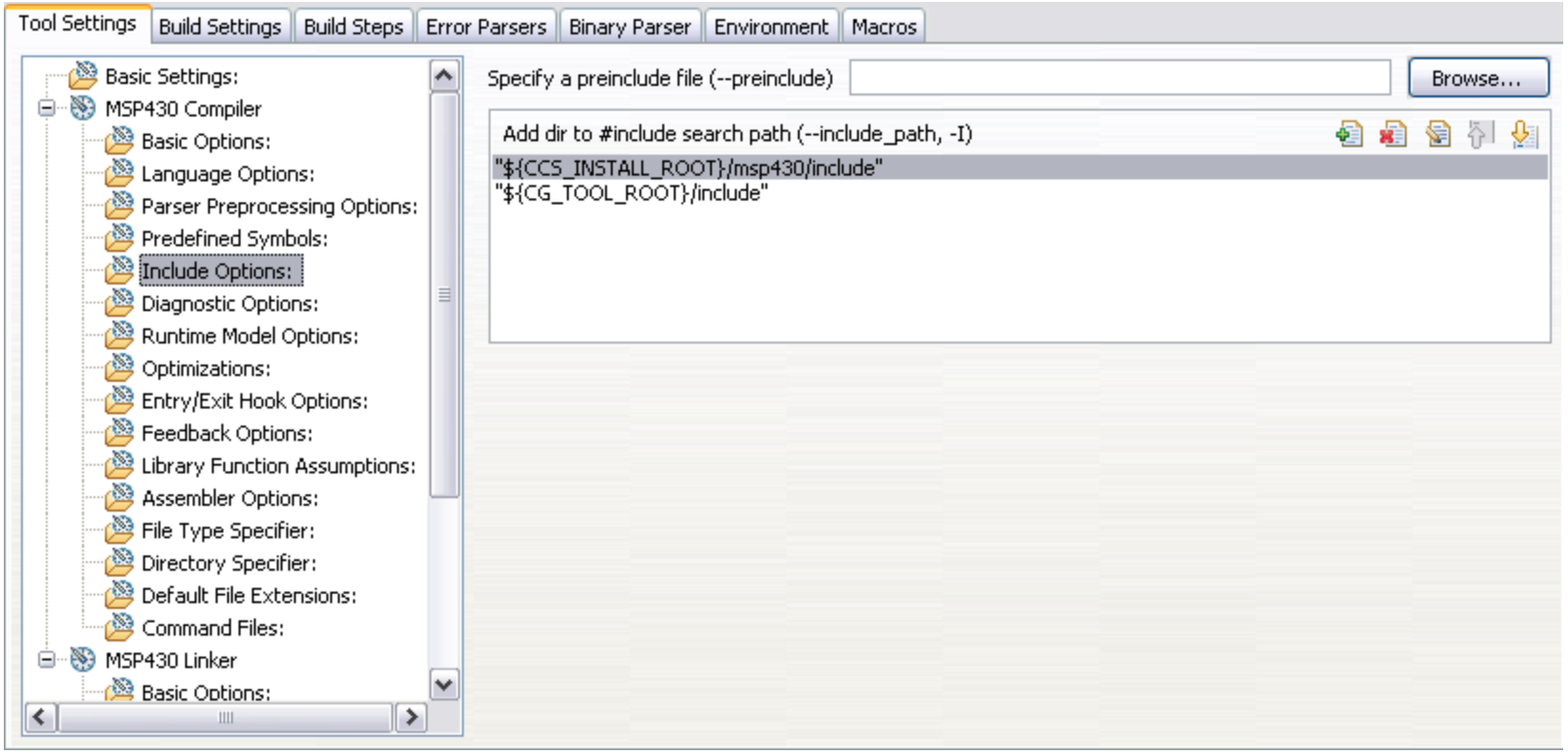


- The following slides will go into more detail on the more important options.

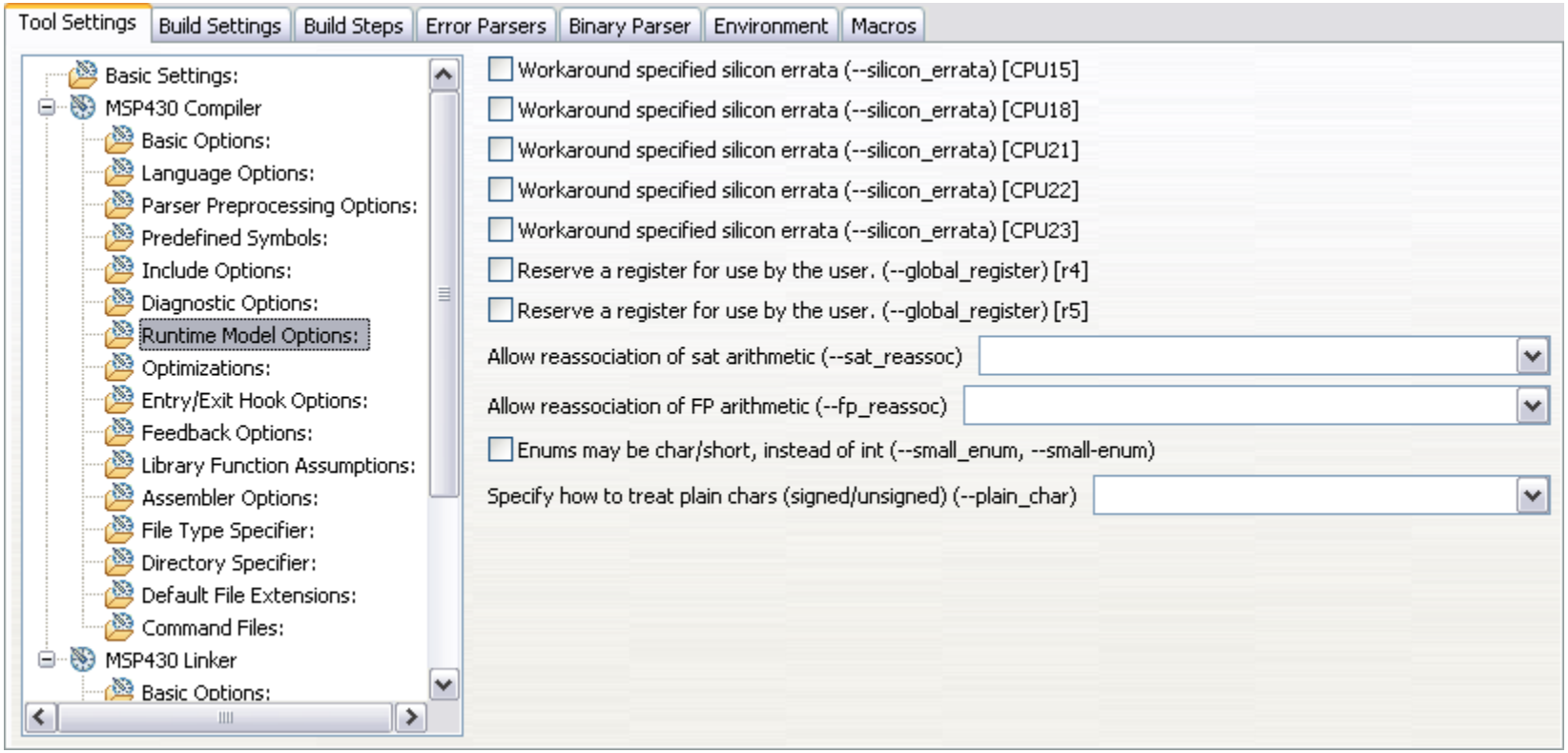
Compiler: Basic Options



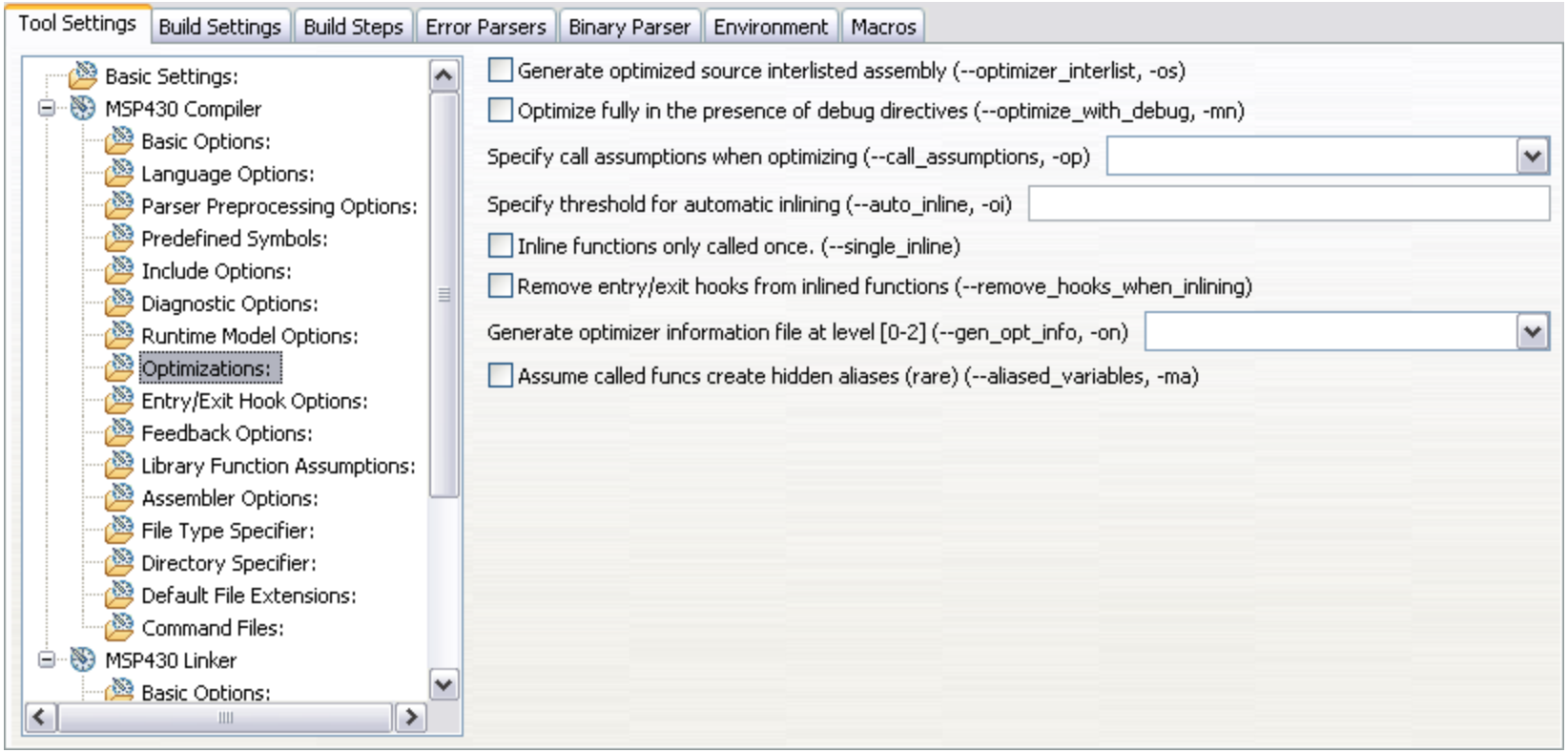
Compiler: Include Options



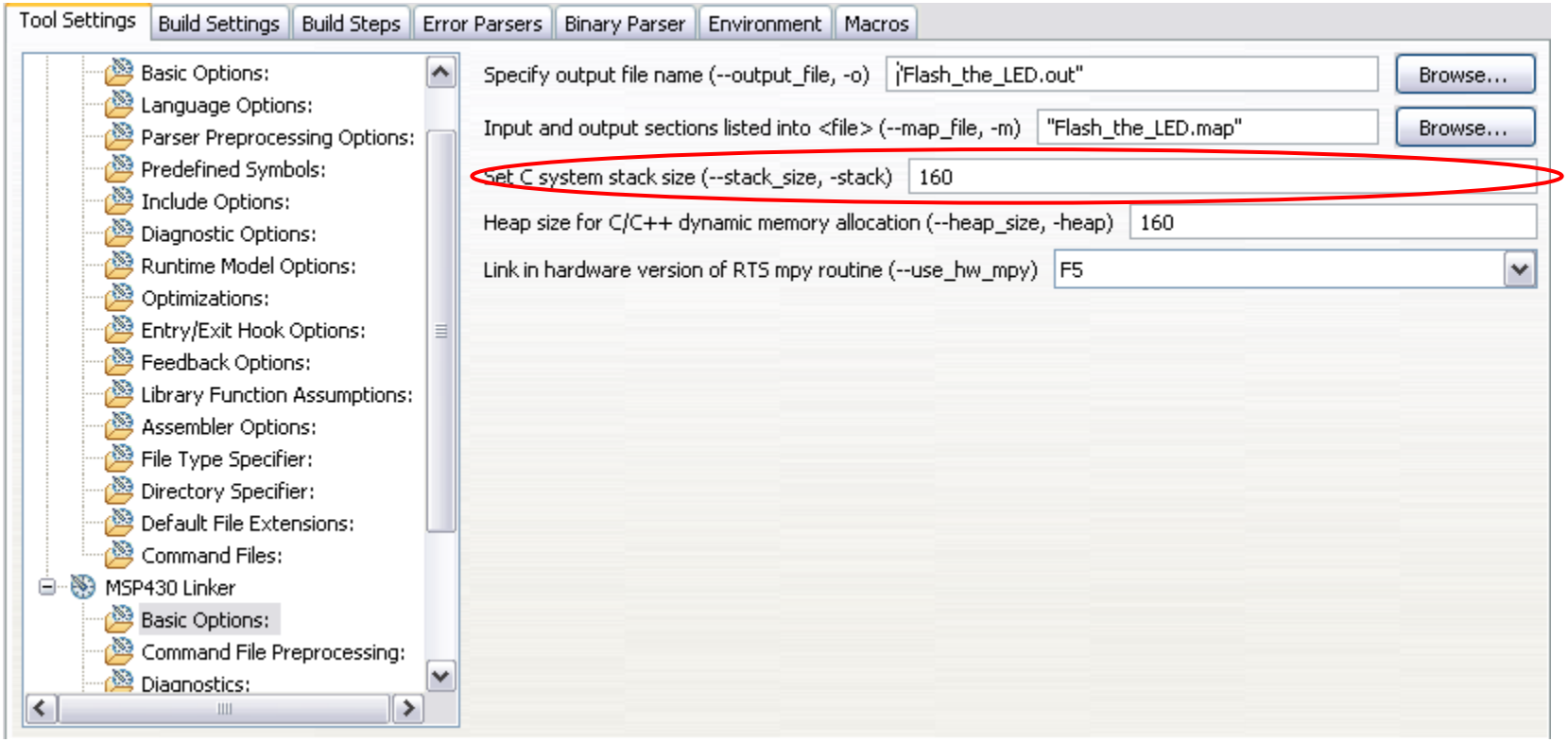
Compiler: Runtime Model Options



Compiler: Optimizations



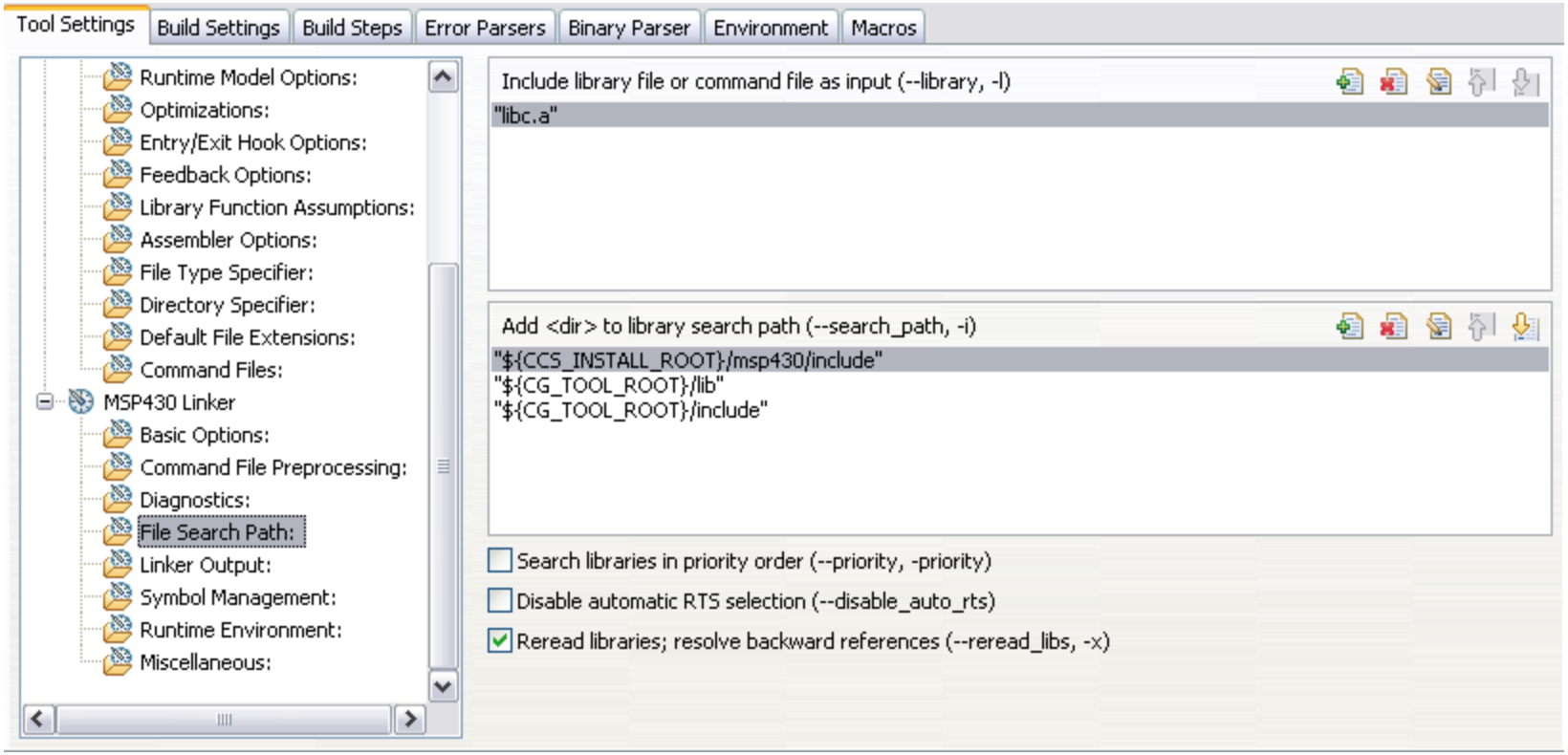
Linker: Basic Options



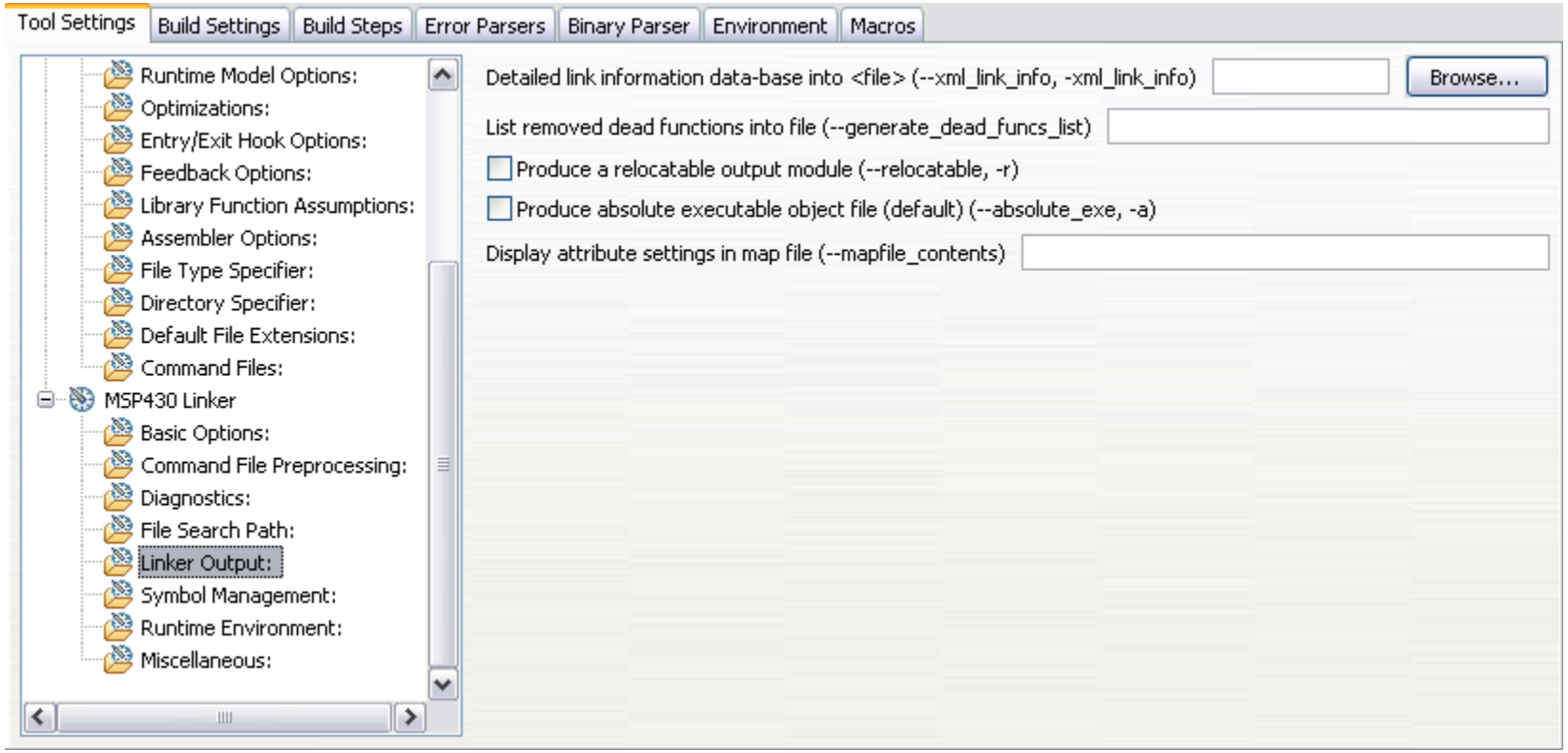
Note:

If the stack size field is empty, you will get a linker warning during build (but the default size will be used). Also, the default linker command file does not include the stack settings, so you need to set it here!

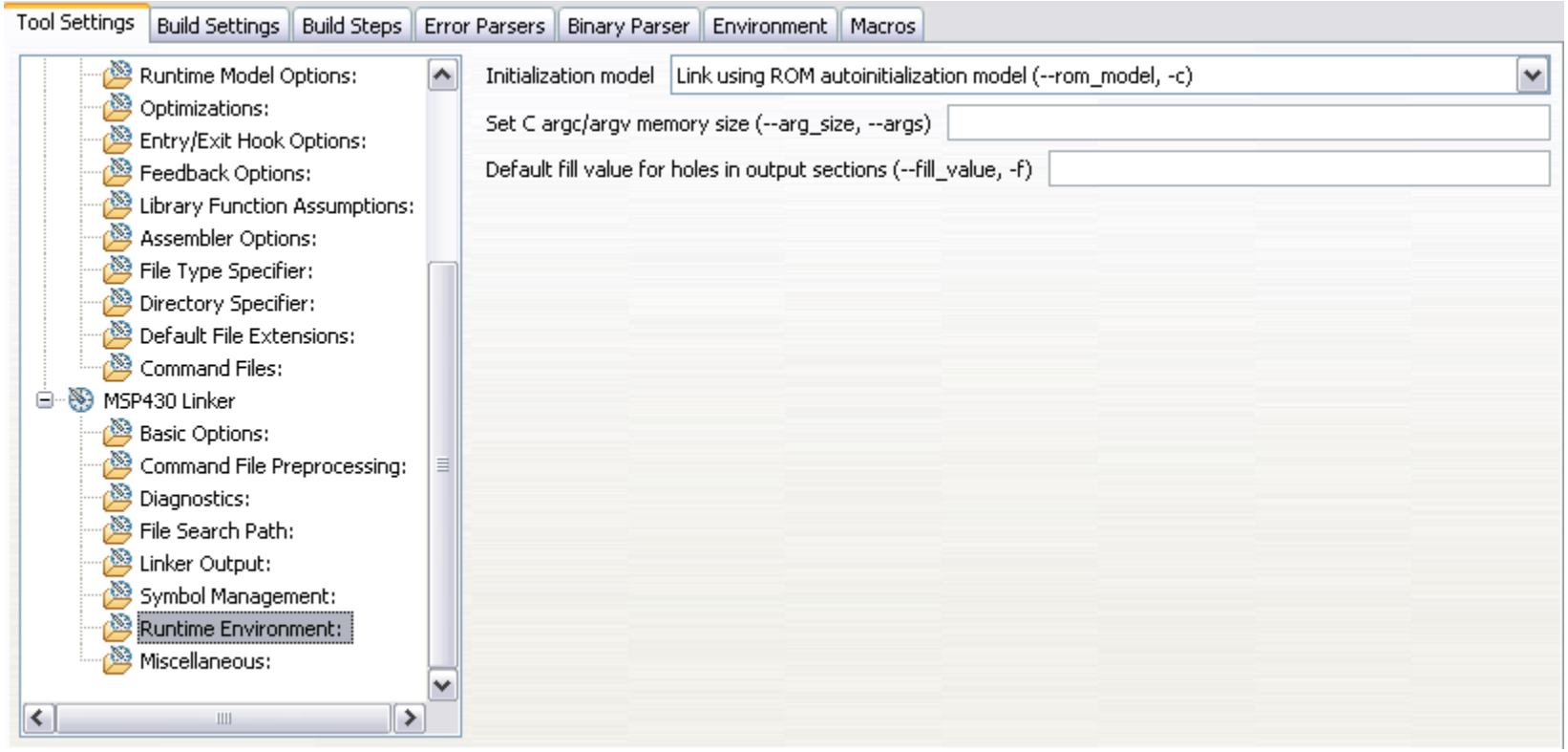
Linker: File Search Path



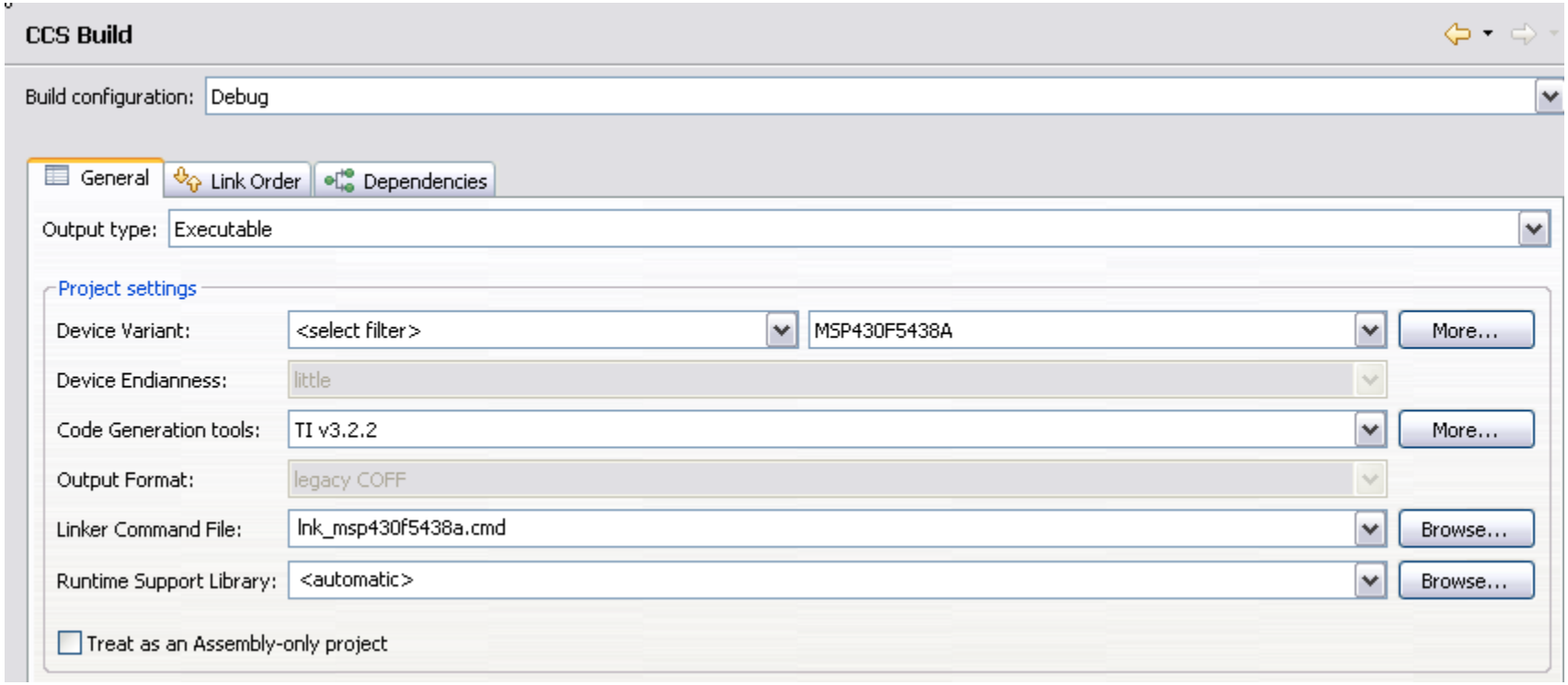
Linker: Linker Output



Linker: Runtime Environment



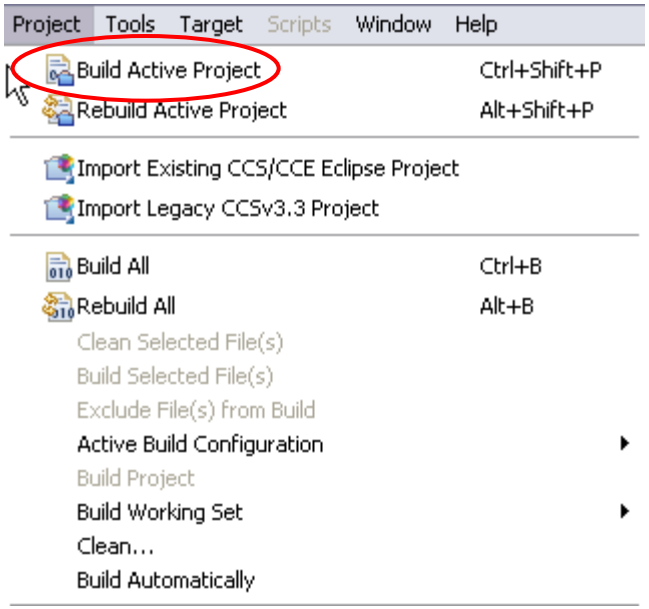
CCS Build: General



Finally, the Actual Build

- There are a couple of ways to build the project:
  - Use the menu bar:

  - Use the drop-down menu:



  - Use the keyboard –shortcuts CTRL+SHIFT+P or ALT+SHIFT+P

Build: The Differences

- Build Active Project
  - Builds only the active project
  - Compiles only new or changed files (incremental build)
- Rebuild Active Project
  - Builds only the active project
  - Compiles all files, independently of their state
- Build All
  - Builds all projects in the workspace
- Rebuild All
  - Rebuilds all projects in the workspace
- Build Project
  - The same as “Build Active Project”
- Build Working Set
  - Build a pre-defined set of projects
- Clean
  - Cleans a defined set of projects
- Build Automatically
  - Builds automatically once a changed file is saved

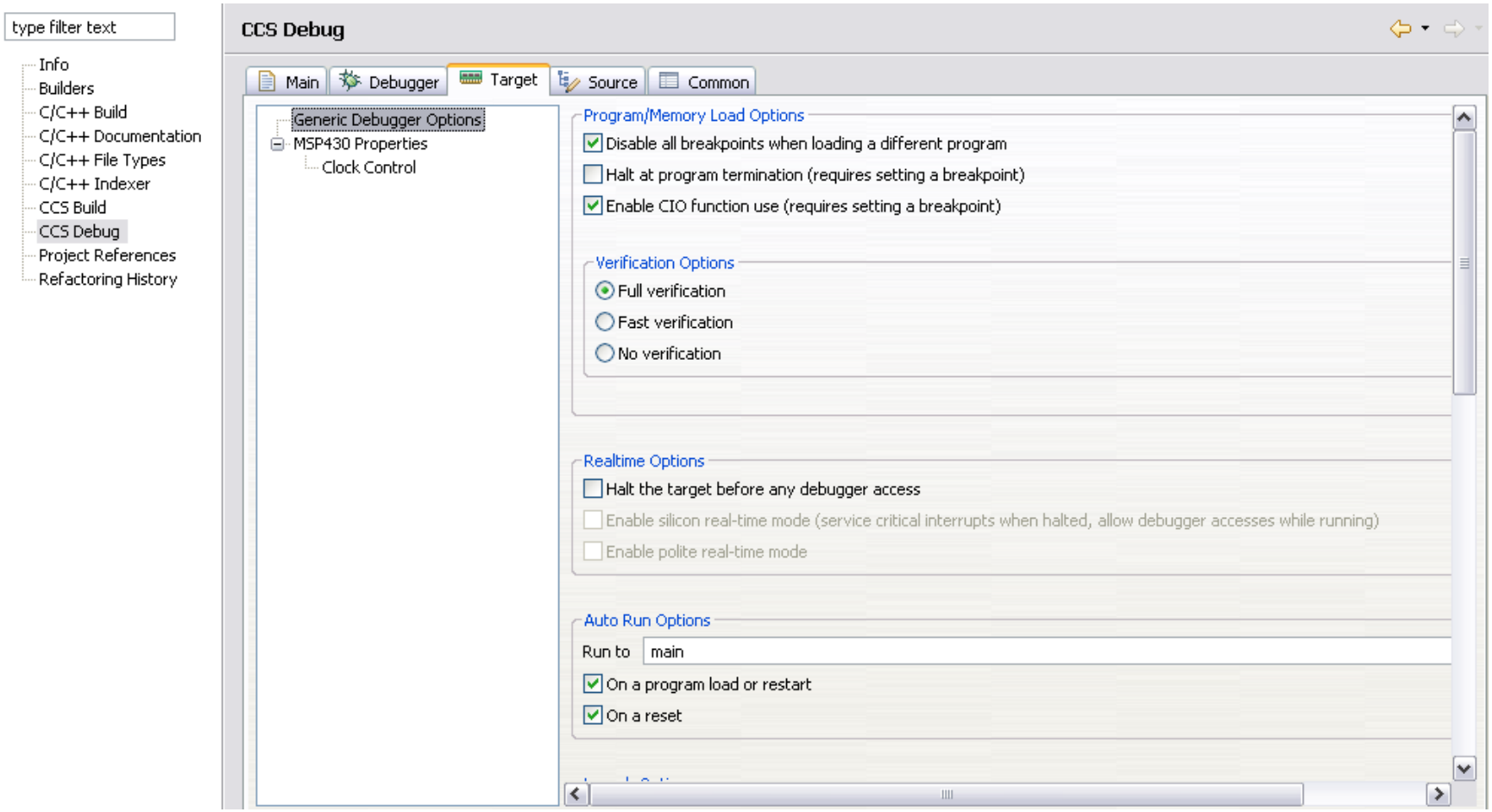
Build: One Last Way

- Clicking will build the active project, switch to the debug perspective and load the executable.
- This will work only, if a valid target configuration is loaded.
- If CCS is already in the debug perspective, the project will be recompiled and the executable loaded.

Agenda

- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

CCS Debug Options

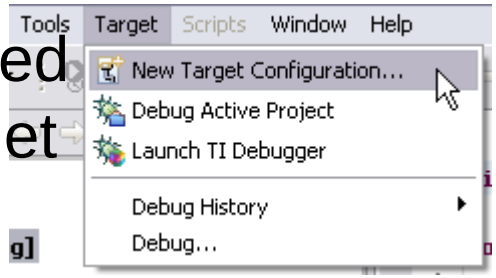


Debugging a Project

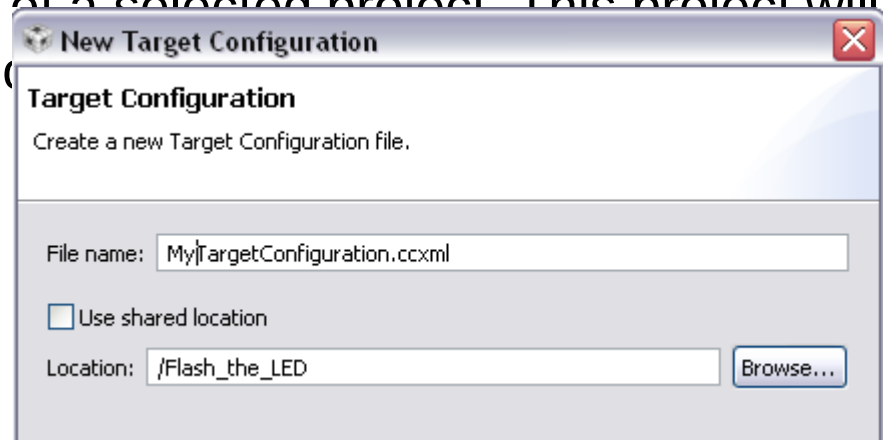
- Once all source files are added to the project and all project settings are made, the project can be debugged.
- But, before we can do that, a target description (aka a target configuration is needed)

Creating the Target Configuration

- System Setup Editor is opened from the Target → New Target Configuration menu

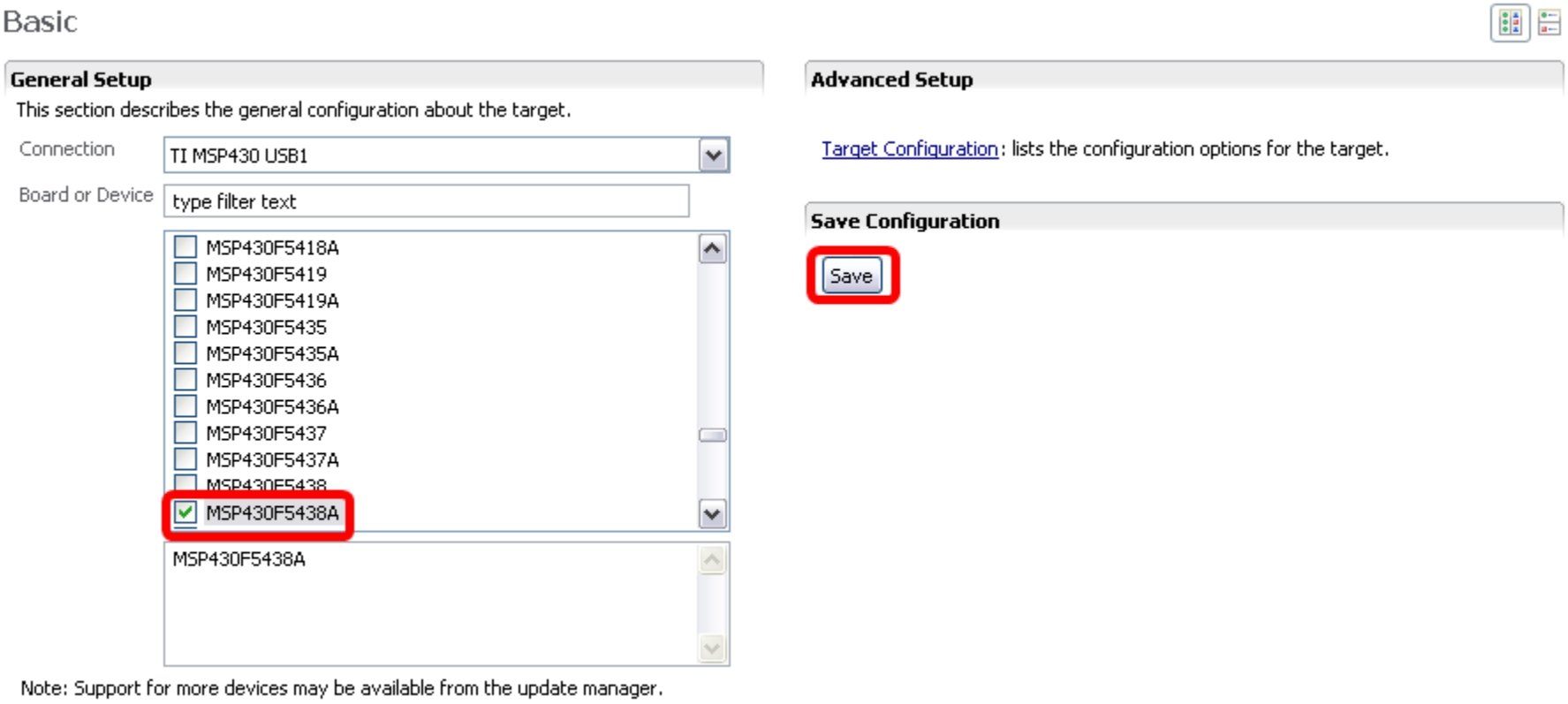


- The new configuration file needs to be named and can be stored in two locations:
  - The “User Defined” folder, which makes the configuration available to all projects.
  - The folder of a selected project. This project will then use this configuratio



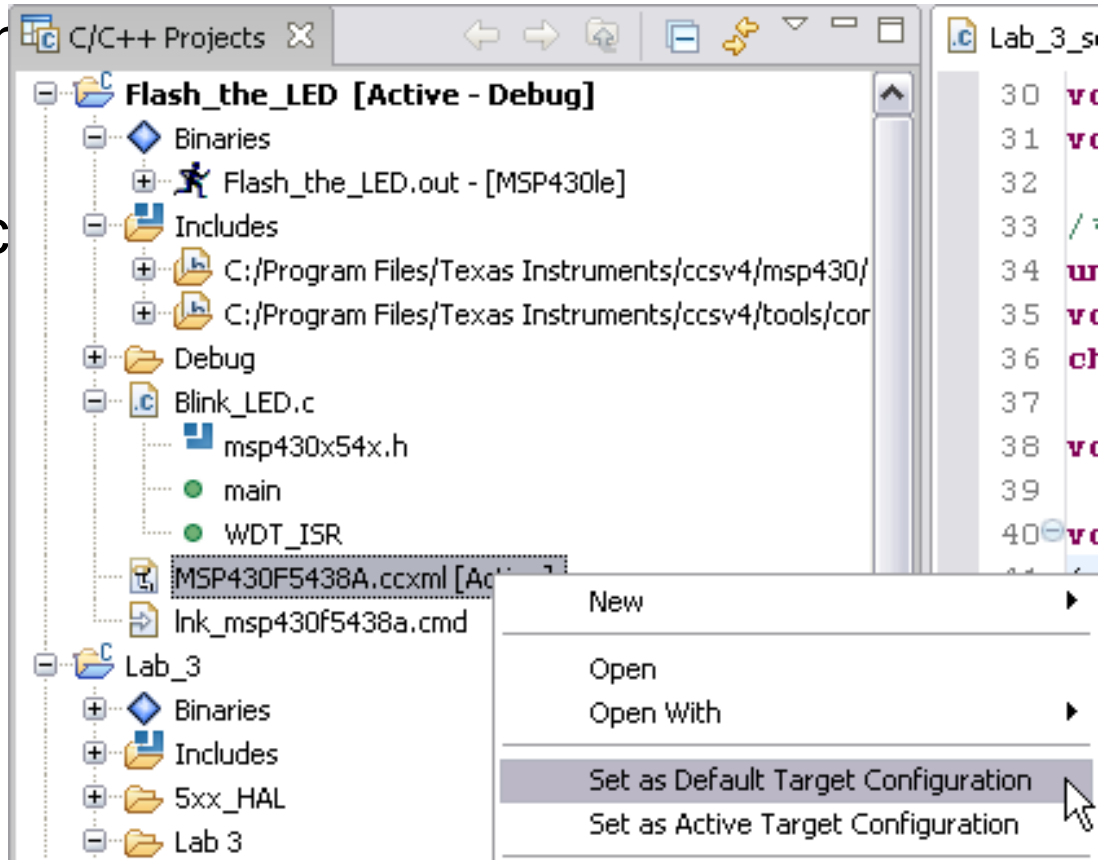
Creating the Target Configuration

- Once the file is created a connection and a device can be added:

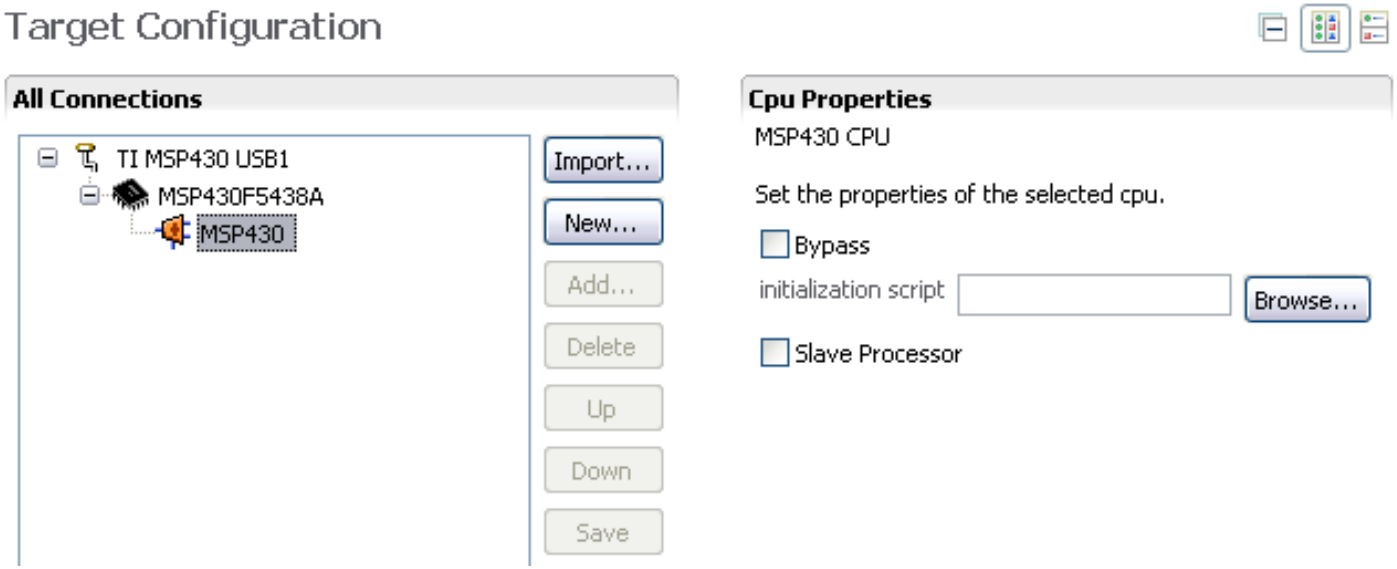


Target Configurations: Use

- Created target configurations are visible from the "Target Configuration" view. From this view, configurations car
  - Launched
  - Linked to a projec
  - Set as default
  - Set as active
  - Deleted
  - Renamed



Advanced Target Configuration

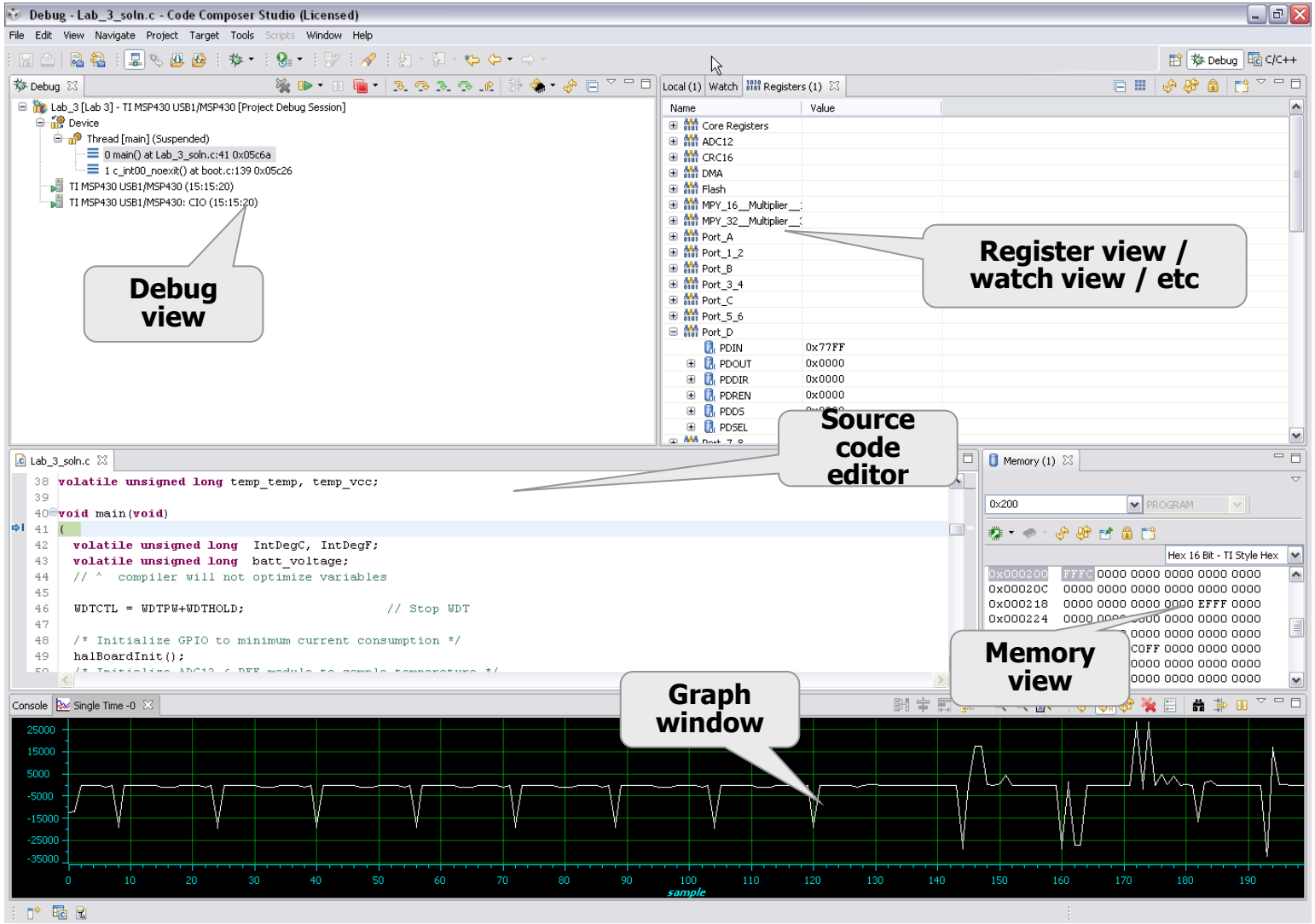


- Allows to customize the target settings properties like
  - JTAG clock frequency
  - GEL scripts
  - Emulator ports
- Allows to add boards, devices, cpu's and routers
- Allows to create new connections for emulators

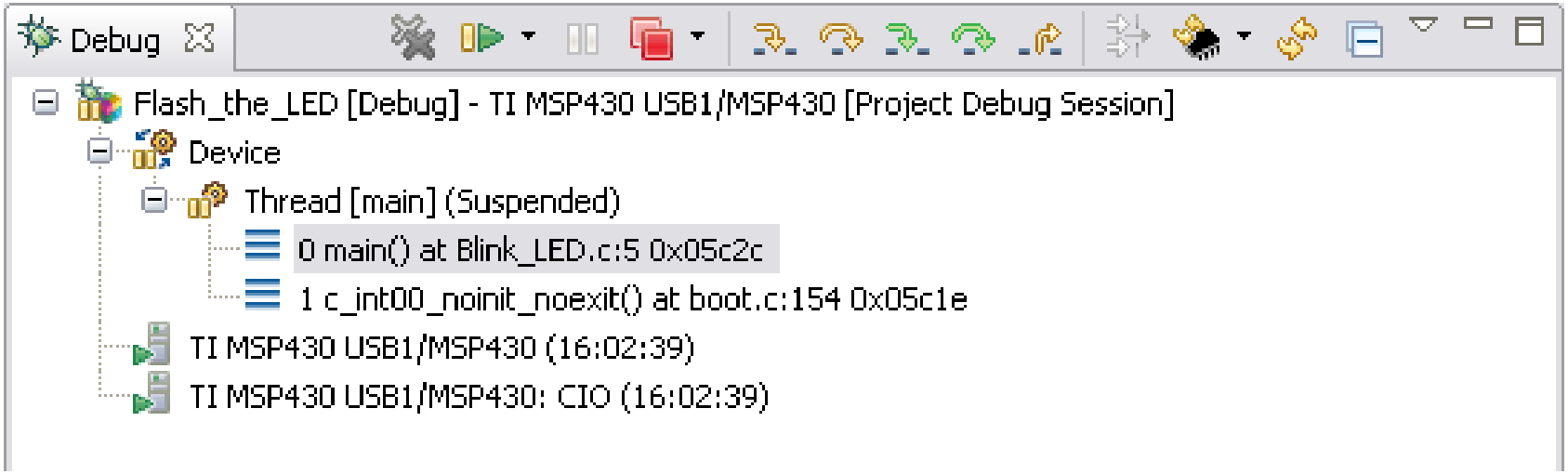
Debug Perspective: Launch

- There are several ways to launch der Debug Perspective
  - Use the perspective bar at the far right
  - Use the *Target → Debug Active Project* menu entry or the *Debug Bug* on the menu bar
    - This will (re-)build the active project
    - Switch to the debug perspective
    - Connect to the target
    - Load the program
    - Go to main()
  - Use the *Target → Launch TI Debugger* menu entry
    - This will switch to the debug perspective
    - But will **not** connect to the target
    - And will **not** load the program

IDE: Debug Perspective



Debugger View



This view tells you the state of the debugger (connected / disconnected), if the program is currently running and at which line (PC) the code is currently executed.

Loading a Program

- If *Debug Active Project* was used to switch to the debug perspective
  - The target is already connected
  - The program is already loaded
  - The PC is already at main()
- If *Launch TI Debugger* or the *Perspective Bar* was used to switch to the debug perspective
  - Connect the target by
    - Selecting *Target* → *Connect Target* or <ALT-C>
    - Using the *Connect Target* button on the menu bar
  - Load the program by
    - Selecting *Target* → *Load Program* or <CTRL-ALT-L>
    - Using the *Load Program* button on the menu bar
  - You might want to issue a *CPU Reset* and a *Go Main* afterwards.

Debugging a Program

Run (F8) / Run Low Power / Free Run

Halt program

Terminate debug session. This will switch back to the C/C++ perspective (CTRL+ALT+T)

Step into (F5)

Step over (F6)

Assembly step into

Assembly step over

Step return (F7)

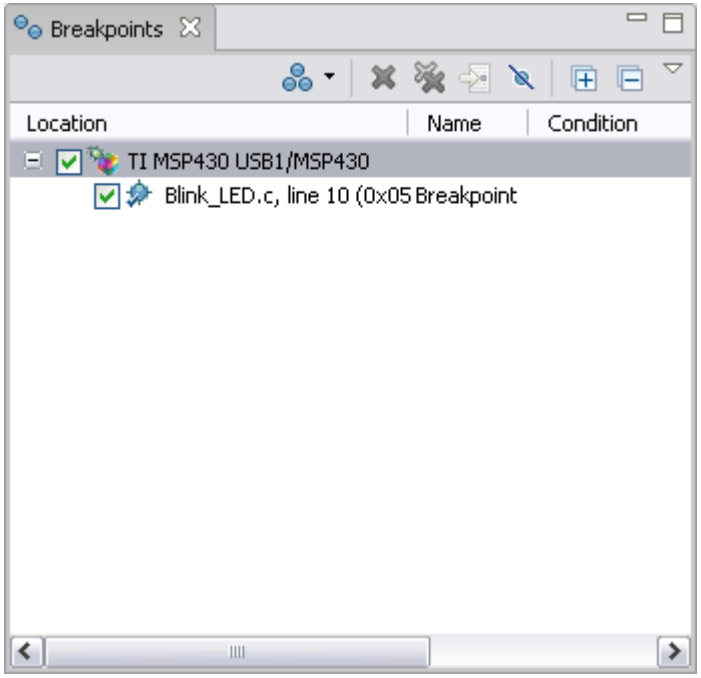
Reset CPU (ALT+R) / Reset Emulator

Restart (ALT+F8)

Breakpoints

- Breakpoints can be set in two ways:
  - Double clicking in the grey bar at the left side of the file view
  - Right-clicking on the line where the breakpoint should be located and selecting “New Breakpoint”
- Breakpoints can be viewed and manipulated (and if needed added) in the “Breakpoint View”

```
3  void main(void)
4  {
5    WDTCTL = WDTPW + WDTHOLD;
6
7    P1OUT = 0x00;
8    P1DIR |= BIT0;
9
10   WDTCTL = WDT_ADLY_1000;
11   SFRIE1 |= WDTIE;
```

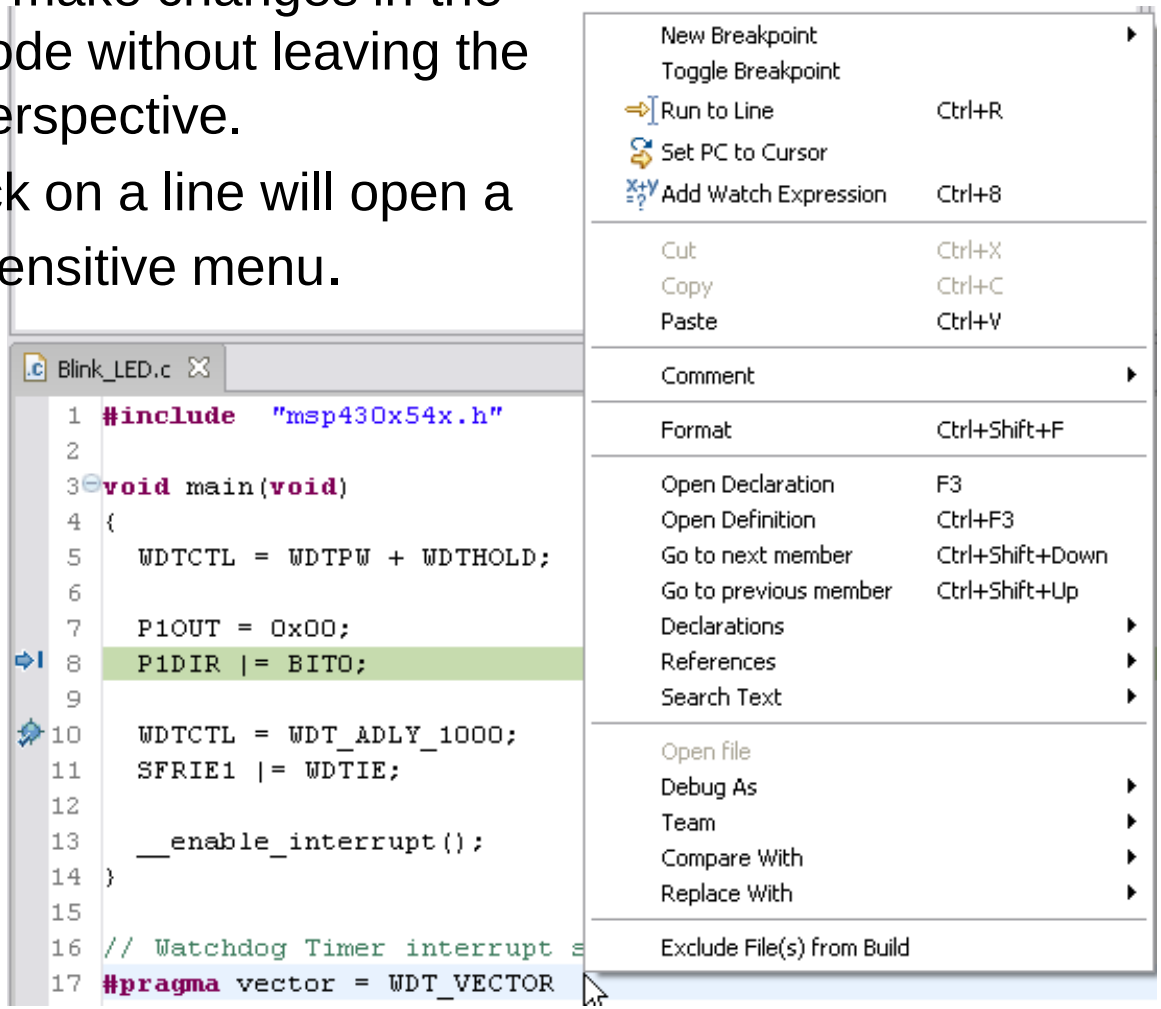


Agenda

- CCS version 4.1 overview
- Starting a new project
- Importing existing projects
- Building a project
- Running & debugging a project
- Debugger Views
- Q&A

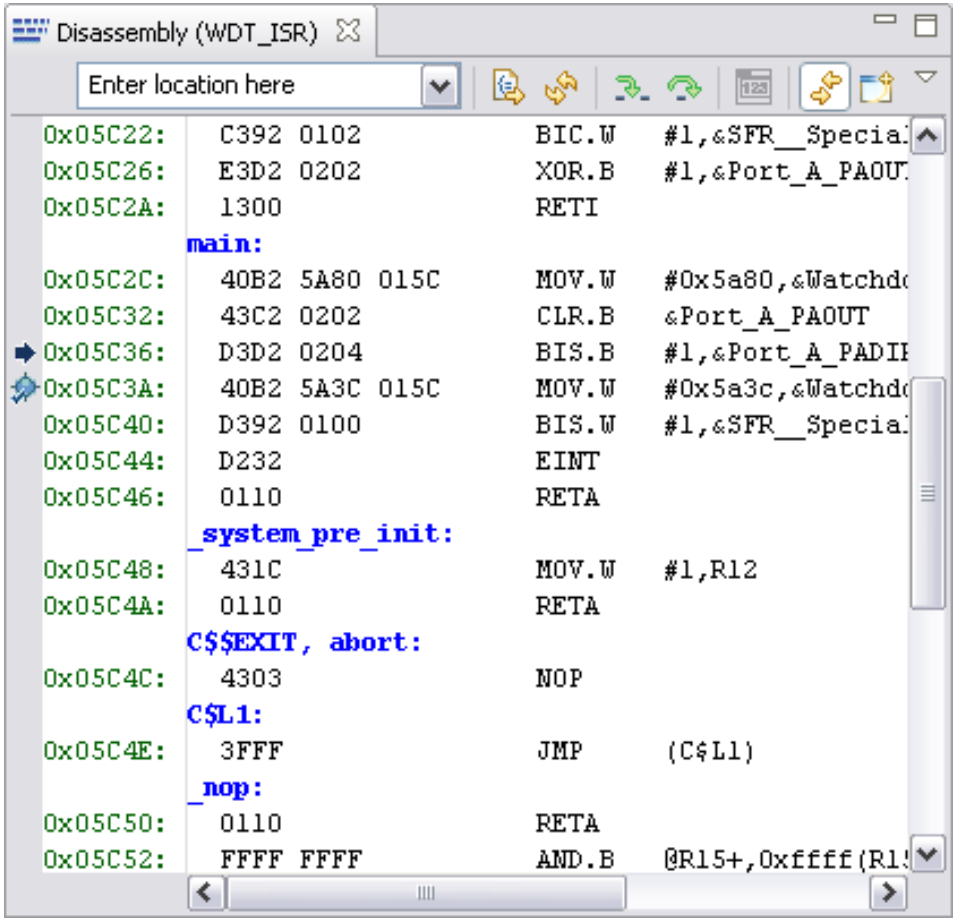
Editor View

- Allows to make changes in the source code without leaving the Debug perspective.
- Right-click on a line will open a context sensitive menu.



Disassembly View

- Opened by selecting "View → Disassembly
- Shows:
  - Address
  - Opcode
  - Disassembly
- Can be customized
- Can be shown
  - Always
  - Only if no C-source code is available
- Can also be used to set breakpoints



Watch Views

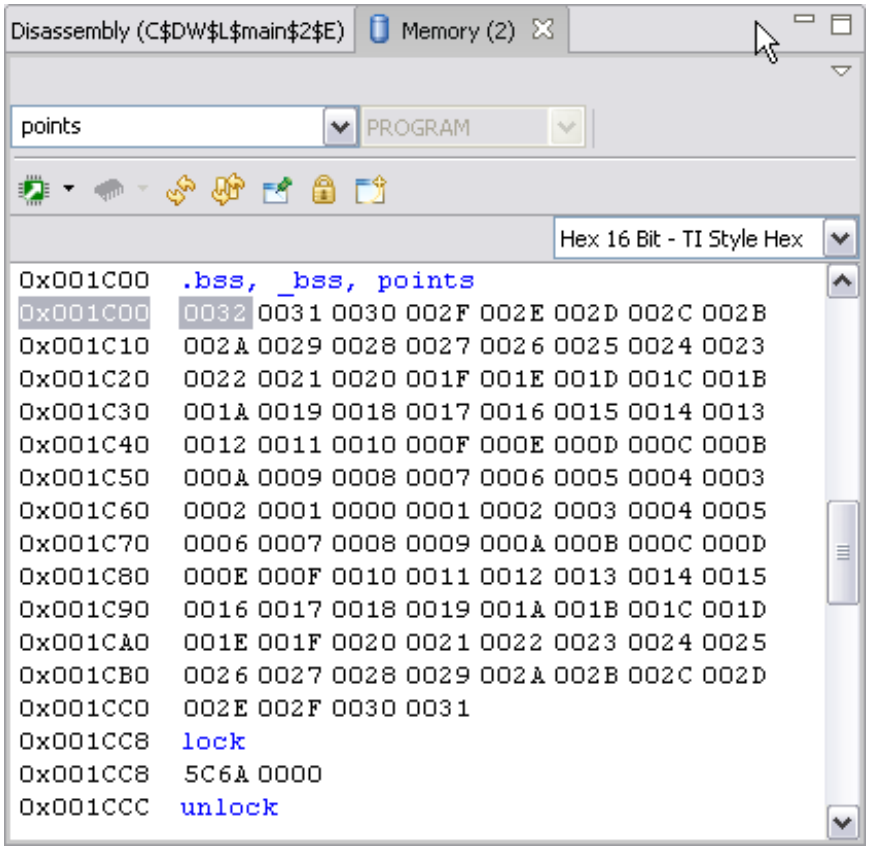
- Local variables are displayed automatically
- Other watch expressions can be added easily
- Expressions can be a global or a local variable or logical / arithmetic expressions using them (or use it as a quick calculator)
- Additional watch views can be added and the display can be customized
- Variables can also be watched in Q format

Local (1) | Watch (1)

Name	Value	Address	Type	Format
i	100	0x00000405@Data	unsigned int	Natural
value	50	0x00000404@Data	int	Natural
value * i	5000		unsigned int	Natural
<new>				

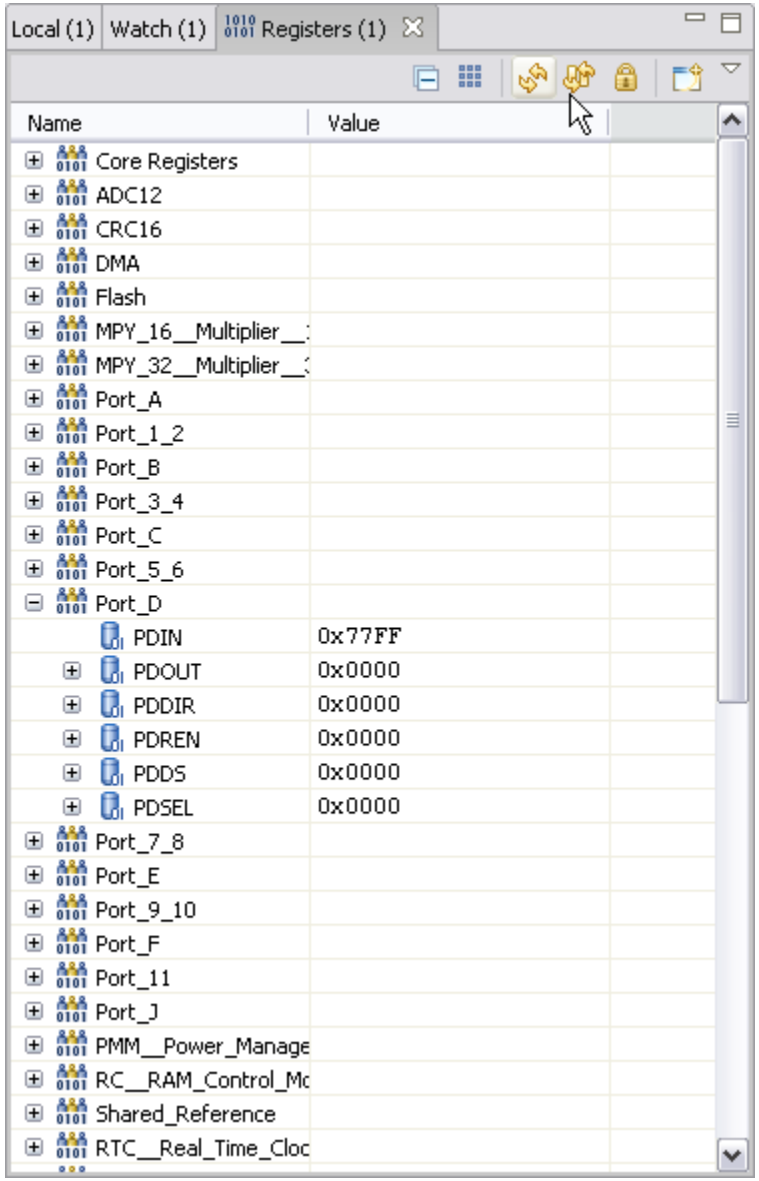
Memory View

- This view can be opened by "View → Memory"
- Displays the contents of the memory at a certain address
  - If variables are used to set the address the '&' operator needs to be used unless the variable is an array.
- Memory contents can be displayed in several ways:
  - Floating point
  - Integer
  - Hexadecimal (TI and C style)
  - Binary
  - Character
  - 8/16/32/40/64 bit
- The memory view also allows to save, load and fill memory
- Several memory views can be open at the same time



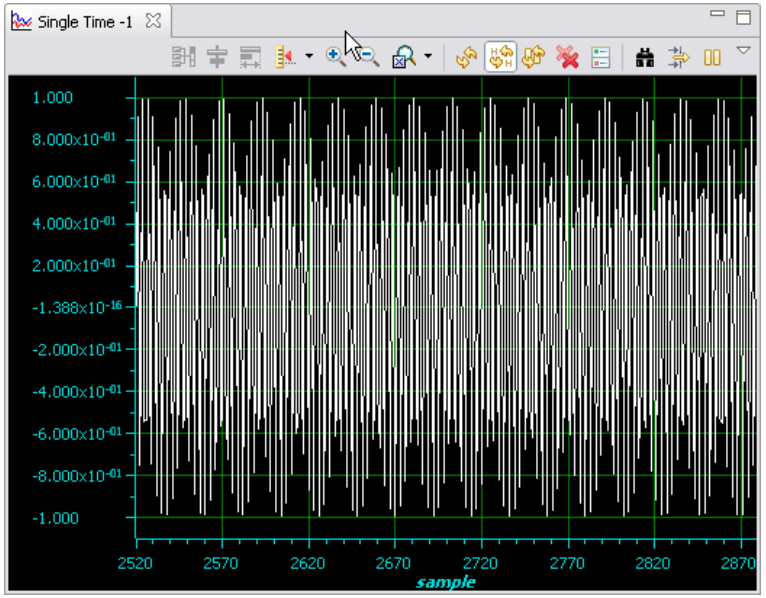
Register View

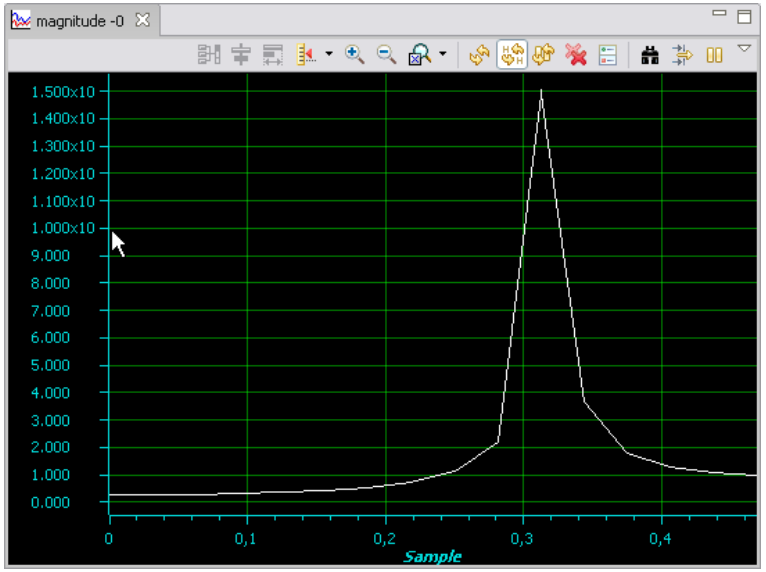
- Opened from “View → Registers”
- One window will display all the register sets
- Multiple register windows can be opened at any time
- Registers can be manipulated from this view
- Two different views are available:
  - Tree (default)
  - Grid (with a selectable number of grids)
- Registers can be copied individually or as complete set



Graph View

- Can be accessed from Tools → Graph
- Time-Domain
  - Single / Dual
- FFT:
  - Magnitude / Magnitude Phase
  - Complex
  - Waterfall
- Different update modes:
  - Manual
  - Refresh on halt
  - Continuous refresh
- Display can be customized
  - Line style
  - Grid
  - etc.
- If in continuous refresh mode, the update interval can be set in the local menu of the Watch View ☹
- Data can be exported



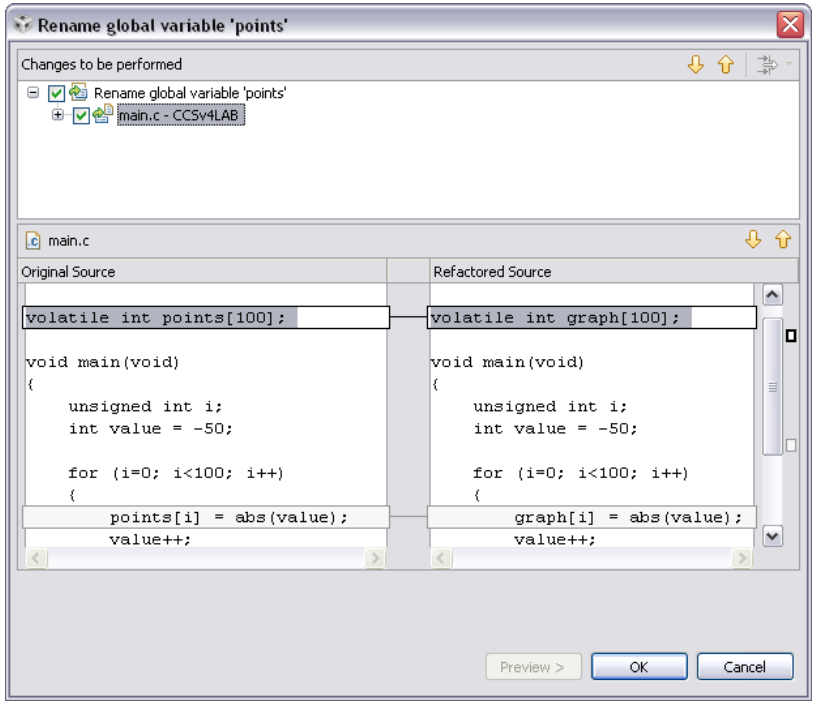


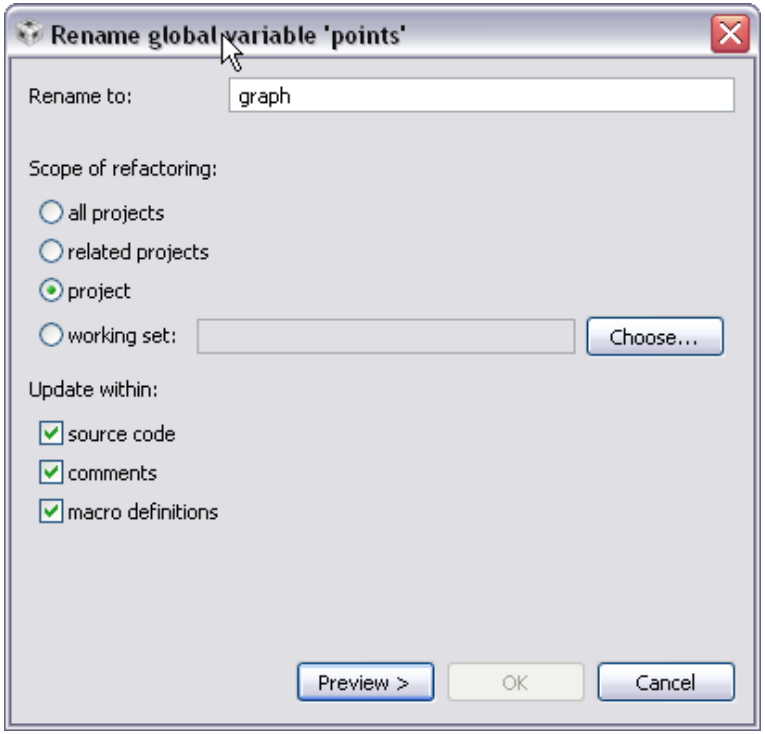
Other Debugging Options

- Under “Tools → Debugger → Generic Options other stuff can be configured as well:
  - Behavior of breakpoints if another program is loaded
  - Autorun: Where the program should run to after a load or a reset
  - Different launch options: Connect to target, restore breakpoints from previous session (can be dangerous!) etc
  - Display of processes etc
  - Interrupt handling during assembly debugging.

Refactoring

- Refactoring allows you to rename variables etc on a project or system wide level.
- Select a variable, right click and select *Refactor* or <alt-shift-r> or *Edit → Refactor*





Thank you