The Propeller chip, a multi core micro controller...

A Presentation by Fredrik Safstrom
Bamse@alleberg.com

:: agenda ::

- An introduction to the Propeller.
- The internal workings, what makes it tick.
- Programming the Propeller in Spin and PASM.
- The Propeller Object Exchange and Wiki.
- Adding "Virtual peripherals".
- Programming the Propeller in Linux
- Demo / Questions
- Introduction to the Hydra.
- Game development on the Hydra.

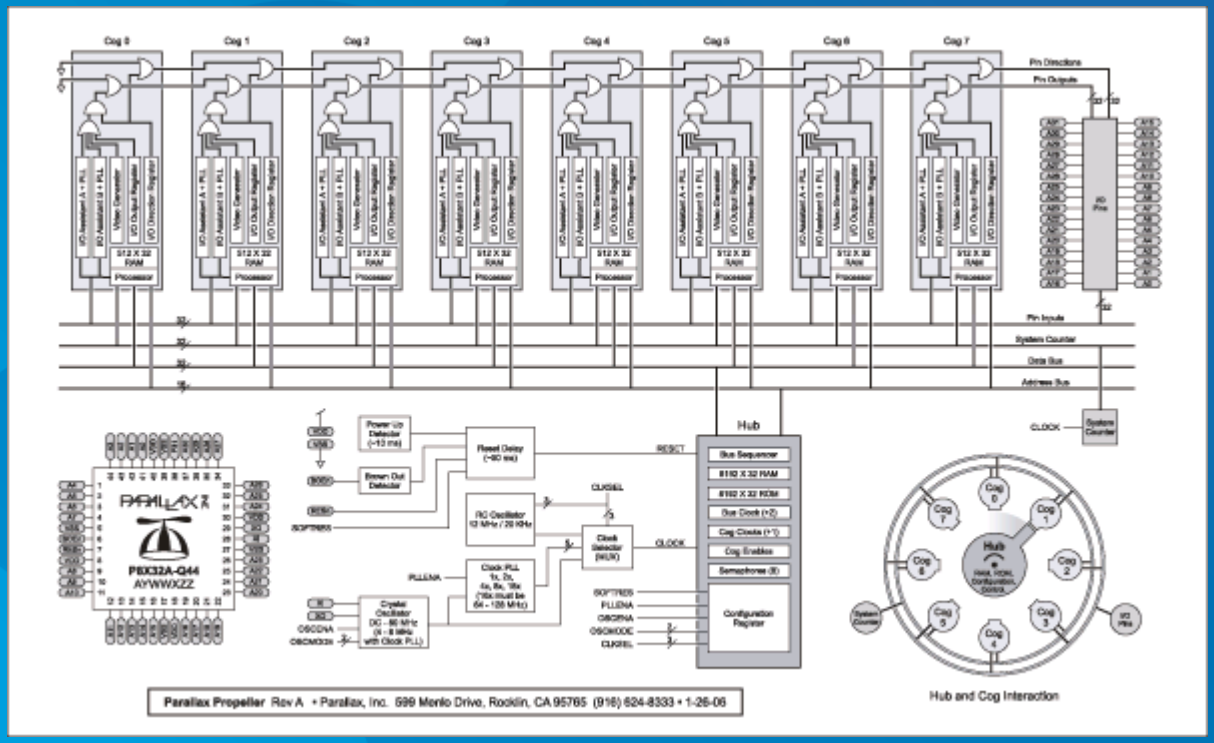
:: introduction ::

- Developed in house by Parallax
- Runs at 3.3V
- 8 32bit CPUs called COGs
- Each COG have 2KB of RAM
- Shared 32KB ROM and 32KB RAM
- The HUB manages shared resources
- 32 Input/Output pins
- Two timers per COG
- One Video Generator per COG

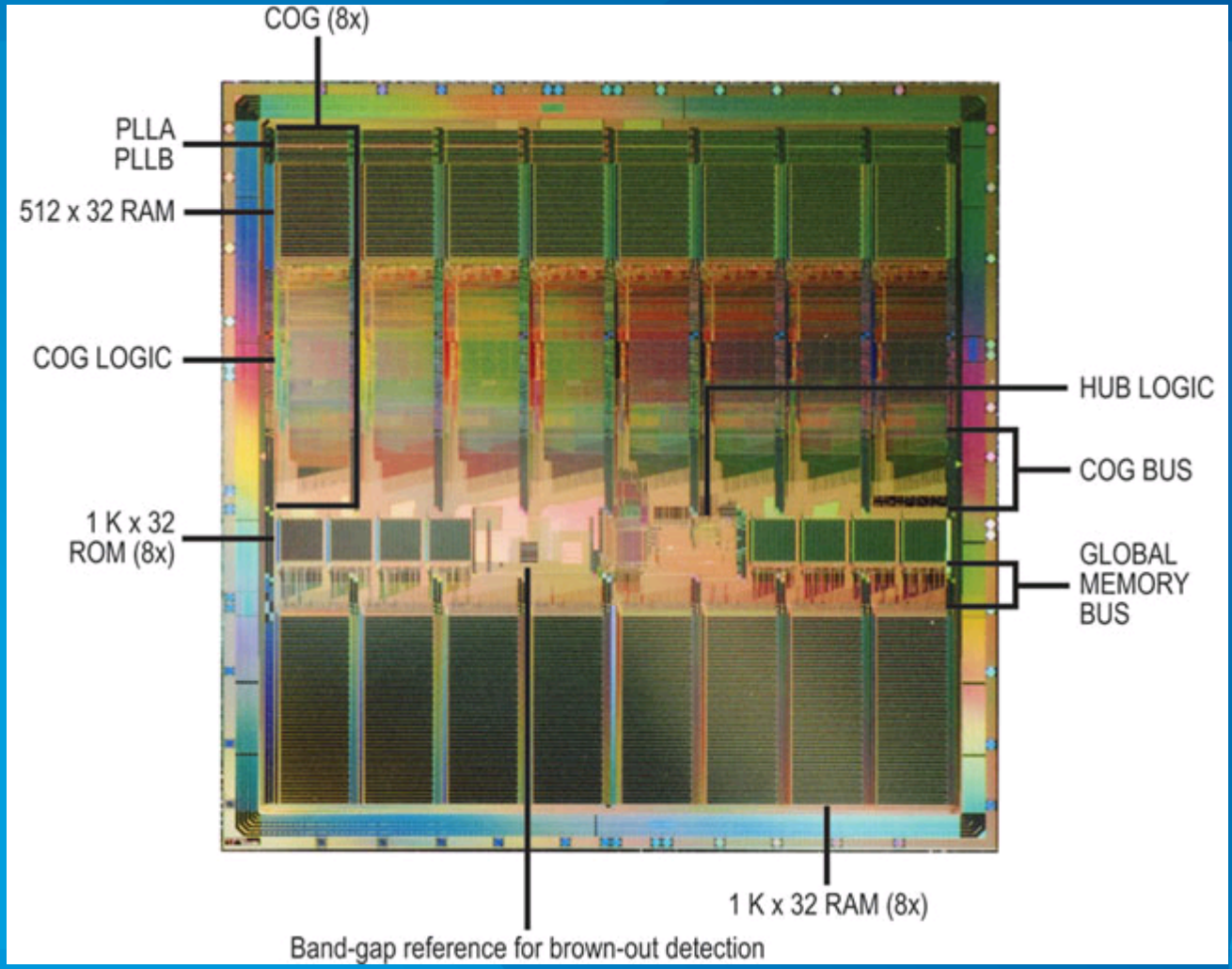
:: introduction ::

- Normally runs at 80 MHz, 20 MIPS
- 8 COGS give you 160 MIPS
- Programmed in SPIN or PASM
- SPIN is a High Level Language
- Compiled into byte code like Java
- Runs about 0.1 MIPS
- Propeller Assembler (PASM) runs at 20 MIPS
- Boot from either Serial or external EEPROM

:: internal workings ::



:: internal workings ::



:: internal workings ::

- 8 32bit CPUs “COGS”, runs independently
- Each cog has 2KB or 512 Longs
- Assembler program must fit within 512 Longs
- Most instruction takes 4 cycles
- Each COG has two timers (A and B)
  - Waveform Generation, PWM, Digital to Analog
  - Analog to Digital, Frequency counting
  - Measuring pulse width, RF carrier
- Each COG has one Video Generator
  - Can generate NTSC, PAL or VGA

:: internal workings ::

- Shared resources
  - 32 I/O pins
  - System clock
  - 32KB or RAM and 32KB ROM
  - Locks aka Semaphores
  - COG instructions
- System Clock and I/O pins are common
- Rest are Mutually Exclusive
  - Controlled by the HUB
  - Gives access in a “round robin” fashion.

:: internal workings ::

- I/O Pins
  - Input only if no other COG set it to output
  - Output low only if no other COG set it to high
  - Output high if any COG set it to high
  - 40mA Source/Sink each, total of 300mA
- System counter
  - Derived from X-tal and PLL or Internal RC
  - PLL 1x, 2x, 4x, 8x or 16x X-tal
  - Normally 80MHz by 5MHz X-tal and PLL 16x
  - PLL should be between 64MHz to 128MHz

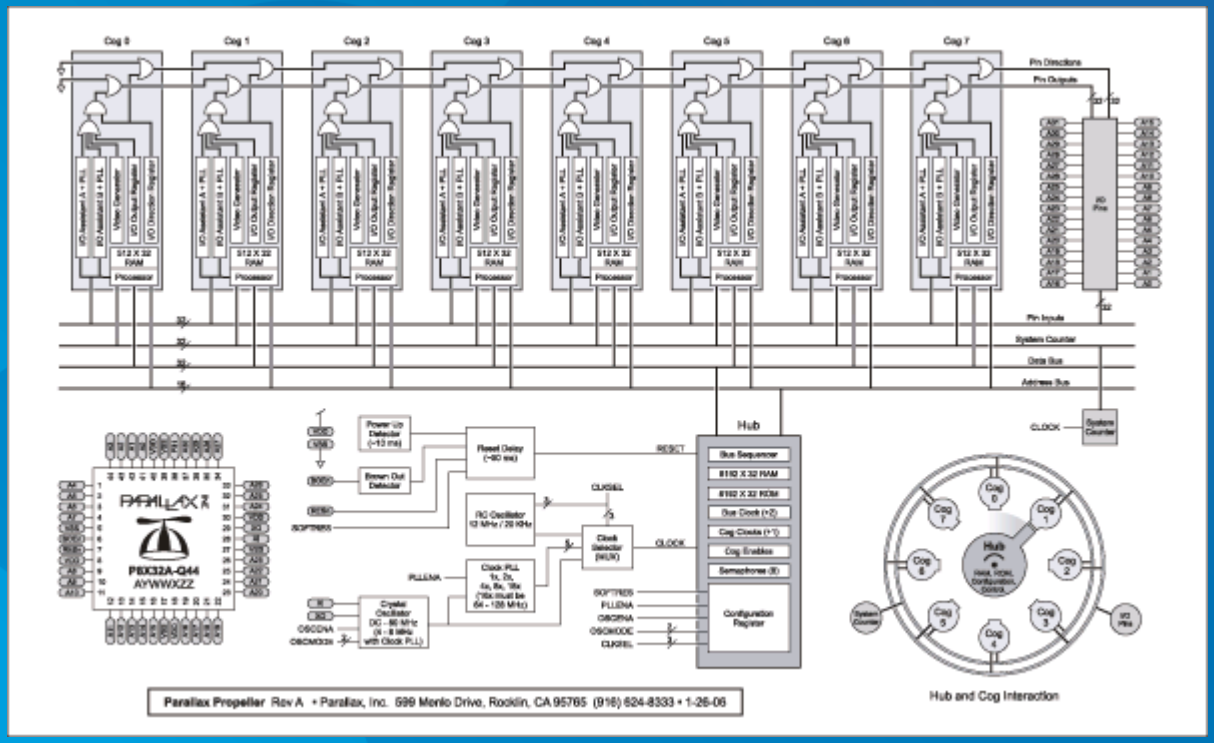
:: internal workings ::

- HUB
  - Gives access to Mutually Exclusive resources
  - Memory, Locks, COG operations, System clock
  - Round robin fashion, COG# 0, 1, 2, 3, 4, 5, 6, 7
  - Runs at half speed, 16 cycles to make one turn
  - HUB instructions takes 7-22 cycles
  - 7 if lucky
  - 15+7 worst case scenario
  - HUB operation + 2 instructions to synchronize

:: internal workings ::

- No interrupts
  - Wait for pin or System clock in real time
  - Start new COG and continue wait
- Boot procedure
  - 1. Try detect a host on pin 30 and 31
  - 2. External EEPROM on pin 28 and 29
  - 3. Stops and goes into shutdown mode
  - Loads 32KB of data into RAM
  - Starts Spin interpreter on COG 0
  - Starts the main Spin program

:: internal workings ::



:: programming ::

- SPIN
  - High level language
  - Easy to learn
  - Mix between Pascal, C and Python
  - Relies on indentations for blocks { code }
  - Runs about 0.1 MIPS
  - Compiled into Byte code
  - 32KB code space shared with data
  - Supports Objects but it's not an OOP Language

:: programming ::

- Propeller Assembler
  - Low level language
  - Runs at 20 MIPS
  - Relies on self modifying code
  - Instructions as movi, movd and movs
  - Instructions may or may not set C or Z flags
  - All instructions conditional on C or Z flags
  - [INSTR][ZCRI][CON][DEST][SRC] 6-4-4-9-9
  - 512 instructions per COG
  - No stack, no recursive subroutines

:: Propeller Object Exchange & Wiki ::

- Propeller Object Exchange
  - Way to share code supported by Parallax
  - Anyone can upload/download
  - Entries are moderated by Parallax for quality
  - Free under MIT License, X11
  - http://obex.parallax.com/
- Propeller Wiki
  - FAQ, Tutorials, Tips and tricks etc...
  - I'm working on my second tutorial for this Wiki
  - http://propeller.wikispaces.com/

:: Virtual peripherals ::

- Propeller have no built in peripherals
  - No serial communication
  - No Analog to Digital or Digital to Analog
- Use Virtual peripherals
  - Download Objects from the Objects exchange
  - More flexible than peripherals on fixed pins
  - Switch functionality on pins
  - Usually one COG per peripheral

:: Virtual peripherals ::

- Examples of Virtual peripherals
  - RS 232, I2C, SPI, 1-wire, TCP Stack protocol
  - Analog to Digital or Digital to Analog
  - Signal generation, PWM, Duty, sound
  - PAL, NTSC, VGA, LED, VFD, LCD displays
  - Servo controller, Stepper motor, Wheel encoder
  - Keyboard, mouse, joystick, PS2 pad
  - Floating point functions, PID control, FFT
  - Sensors, Temperature, GPS, Accelerometer
  - External RAM, ROM, SD cards, Memory stick

:: Linux ::

- Not officially supported by Parallax
- Propellent command line compiler by Parallax
- Works under Wine
- Compile spin code to binary/EEPROM
- Use loader.py script to program propeller
- loader.py requires pyserial
- I got it to work on Ubuntu 8
- Supports MacOS as well
- More instructions on Propeller Wiki

:: How to get started ::

- Parallax have started kit
  - Hydra $200, includes book + examples
  - Hydra also available @ XGameStation.com
  - Propeller started kit $100, printed manual
  - Propeller Education kit $80
  - Propeller Demo board $80
  - Propeller Protoboard $20 + Prop plug $25
  - Propeller Protoboard with USB $40
  - Propeller Protoboard $20 + $5 serial components

Demo

- Show SPIN and Assembler language
- Dummy C=64 Terminal Demo
  - My example uses two virtual peripherals
  - One VGA and one RS232
  - Reads a Commodore 64 Keyboard
  - Displays on Monitor and sends over serial port
  - Also receives data from serial port

:: Introduction to the Hydra ::

- Developed by André laMothe
  - A demo board for Game development
  - Built in USB port for programming/serial
  - Two NES game pad sockets
  - Keyboard and mouse
  - Expansion port, replaceable X-tal
  - 128KB EEPROM, debug LED
  - Hydra Net to connect Hydras
  - PAL/NTSC video and sound
  - VGA output shared with expansion port

:: Game development on the Hydra ::

- Propeller powerful enough for games
- Learn low level game development
  - Generate NTSC/PAL/VGA signals from scratch
  - Generate sound from scratch with timers
  - How to read game pads with shift registers
  - Read Keyboard/mouse signals
  - Hydra net communication protocol
  - Use add-on Memory, EEPROM or SD cards
  - Make your own add-ons with expansion port

Demo