68HCS12

U of A System Setup

U of A System: DragonFly

U of A System: Power Supply

5 Volt Regulator, 1A
**U of A System: Led Display**

- Tristate Buffer
- Open Collector Output
- 20mA
- Output Enable

**Reset, A-In, D-In, D-Out**

- what is this buffer for?

---

**Lab 1 Introduction**

GOTO Website

---

**HC9S12C32**

**Setup**

Goal is to configure all pins to create a functional circuit

The DragonFly module already pre-configures your microcontroller
### Mode of Operation

- **Single chip mode**
- **Executing on external memory (48 pin does not have bus expansion routed to physical external pins)**
- **Debug Mode**
Mode Selection

- MODA, MODB, MODC, XCLKS, ROMCTL
- MODA, B relevant for external bus (not accessible in DragonFly)
- MODC affects debugging
- ROMCTL allows disabling of internal flash memory

<table>
<thead>
<tr>
<th>BMDM</th>
<th>MODC</th>
<th>RES + MODB</th>
<th>RES + MODA</th>
<th>PIN + ROMCTL</th>
<th>ROMCTL</th>
<th>Mode Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td></td>
<td>Separate Single Chip, BCM allowed and ACBR, BCM is allowed in all other modes but a serial start stop is required to make BCM active.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Creation: Normal Narrow, BCM allowed</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>Separate Test</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Creation: Expanded Wide, BCM allowed</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>Normal Single Chip, BCM allowed</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Normal: Expanded Narrow, BCM allowed</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>Expansion: BCM allowed (false operations would cause bus stability meet not to be used)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Normal: Expanded Wide, BCM allowed</td>
</tr>
</tbody>
</table>

Power Supply

- Fast signal transitions on pins demand high instantaneous currents peaks
- Use two decoupling capacitors 1 uF and 0.01 uF as close as possible to $V_{cc}$ and $V_{dd}$ pins (for most integrated circuits)
- Heavily loaded expanded mode systems may require more elaborate bypassing

Bypassing

Line noise filtering @ board input

Supply high dynamic currents & limit ground bouncing @ every IC

Line noise is due to parasitic of the interconnections on the power distribution network
HC12 Power Supply

- $V_{DDR} = +5V$ and $V_{SSR} =$ external supply, ground to internal voltage regulator
- $V_{DD1} = +2.5V$ and $V_{SS1} = 0V$, bypass internal voltage regulator (output)
- $V_{DDA} = +5V$ and $V_{SSA} =$ supply, ground for A/D
- $V_{RH} = +5V$ and $V_{RL} =$ reference voltage for A/D, ground
- $V_{DDX} = +5V$ and $V_{SSX} =$ supply, ground for I/O pins
- $V_{DDPLL} = +2.5V$ and $V_{SSPLL} =$ bypass supply for PLL clock, ground (capacitor), output

Internal Voltage Regulator

What would you want to do if any of those resets are triggered?

Dragon Fly Supply Implementation 1

Dragon Fly Supply Implementation 2
Power Saving Modes

- **Shutdown Mode**
  - Requires POR
  - VDD(1), VSS(1) not available

- **Stop, [Reduced Power Mode]**
  - STOP instruction: chip in full static mode, requires external reset or interrupts to restart
  - VDD(1), VSS(1), lower but available

- **Pseudo Stop**
  - STOP instruction: COP and RTI still running, consumes more power than full stop, faster restart

- **Wait**
  - WAI instruction: data and address bus static, clocks running, faster recovery, requires reset or interrupts to restart

- **Run, [Full Performance Mode]**

Clocks / Oscillators

why do we need a clock in a microcontroller?

Clock Operation

- **Run Mode**
- **Wait Mode allows to disable clocks**
- **Stop Mode**
  - Full: all clocks disabled
  - Pseudo: oscillator runs but system clock stopped, COP and RTI still running
- **Self Clock Mode**
  - If input clock not sufficient quality

External Oscillator

XCLKS pin determines oscillator type

Figure 2-6 - Oscillator Connections (PET™)

Figure 2-7 - External Clock Connections (PET™)
Dragon Fly Clock Implementation

What will happen if you check E/XTAL with oscilloscope while in run mode?

Alternative PLL Clock

\[
PLLCLK = 2 \times OSCCLK \times (SYNR - 1) \times (REFDV - 1)
\]

PLL Clock Registers

PLL Clock Filter

- Passive external loop filter, second order low pass required
- Determines stability of PLL
- See “Device Users Guide” to calculate component values
- XFC pulled high if PLL not needed

how do you pull high?
DragonFly PLL Clock Filter

what PLL clock speed was this designed for? email me answer and you will get bonus points for first lab

PLL versus External Oscillator

It takes 4 OSC + 4 PLL clock cycles to transition between clocks

where is external clock and PLL clock?

Figure 4-2. System Clocks Generator

Clock Summary

• Clock provides time base for microcontroller (execution speed, synchronization)
• Oscillator clock is 8MHz (DragonFly) and derived from an external oscillator
• The core clock is either the oscillator clock or derived from a PLL
• PLL clock provides an alternative clock speed
• PLL clock is derived from oscillator clock

Clock Summary

• Bus clock is 2 times slower than the core clock
• A stable clock is essential for proper MC operation
• Clock monitor circuit detects clock failures
• Do not measure EXTAL & XTAL with oscilloscope (why?)
• We can stop the clock (stop/wait) (how do we start it again?)

1 core clock cycle

1 bus clock cycle
Reset & Interrupts

- A reset is required to restart your microcontroller
- There are several type of resets
  - Power on is detected (CREG)
  - External (by pressing the reset button)
  - Low voltage is detected (VREG)
  - COP watchdog times out (CREG)
  - Clock monitor failure detected (CREG)
- A reset brings you microcontroller into a predefined state

Reset, Clock and Voltage Regulator are connected, how?

Clock & Reset Generator

Internal Voltage Regulator

what program is executed after POR?
Power On Reset

• Detection of $V_{DD}$ transition
• Assert reset until clocks are stable

External Reset

• Restart of your hardware and application
• After the reset is completed the microcontroller executes your main program
• Reset recovery time depends on clock speed

Low Voltage Reset

• If $V_{DD}$ is below desired state we need to reset system to prevent undesirable behavior:
  – protect unintentional corruption of internal registers and flash memory

Watchdog / COP

• Protection against software failures
• User application will need to write code sequence to register periodically
• If software fails to write to register COP occurs
• Based on OSCILLATOR clock
• Occurs at user defined intervals
Clock Monitor

• If clock outside of specifications (too slow) interrupt occurs

what program do you want to run if CM reset occurs?

Real Time Interrupt

• RTI
  – based on OSCILLATOR clock
  – high priority interrupt which occurs at user defined intervals

what is a real time requirement? do we have a need for that at this time?

External Hardware Interrupt

(in addition to RESET)

Interrupt Pins

• Interrupts handle asynchronous requests
• XIRQ (PE0) and IRQ (PE1) are active low
• External hardware can be attached (e.g. keypad)
• XIRQ non-maskable interrupt can always interrupt CPU, for serious system problems, e.g. battery problems
• IRQ software maskable interrupt, used for general requests
Interrupt Pins

- Can be used with multiple interrupt sources
  - Each source must be an open drain type driver to avoid contention between multiple devices (wired OR network)
  - Interrupt service routine will need to determine device that caused interrupt
  - Single pull-up resistor close to interrupt pin disables interrupt

do we have external devices requiring interrupts currently?

I/O Ports

Port Integration Module
I/O

- PA/B[7:0] core logic and multiplexed bus
  - PA0, PB4 (minimal functionality available in 48pin package)
- AD[7:0] analog/digital module
  - 8 channels
- PT[7:0] timer module
  - PWM module routed to this port
- PS[3:0] SCI module
  - one RX/one TX
- PM[5:0] MSCAN, SPI
  - TXCAN, RXCAN, MISO, MOSI, SCK, SS
- PP[7:0] PWM module
- PJ[7:6] interrupt sources
- PE[7:0] general control
  - BKGDB*, XIRQ, IRQ*, ECLK

If a port input/output is not connected to an external pin what is happening to that resource?

Example PORT T (PT)

- Writing to port: output
- Reading from port: input
- Output/Input has different hardware
- Can connect
  - General purpose digital I/O (5V level)
  - Timer (Input Capture, Output Compare)
  - PWM (Pulse Width Modulation)

Example PT

- Address Offset $240: I/O register, holds value driven out or read in

- For every port you can select
  - I/O direction (in/read, out/write)
  - Drive strength (full power, reduced power)
  - Enable pull (up/down) resistors

- Every port
  - 5V digital input/output or analog input
  - Input with selectable pull-up or pull-down resistors
  - 4.7k Ohm pull up or down to 5V (1) or ground (0)

Why do we use a resistor for pull up/down?

On how many pins can we have PWM? Can we have Timer functions and PWM on different pins simultaneously?
Example PT

- Address Offset $241: Read Status of Pins regardless of connected hardware
  - Bit 7: P0T7
  - Bit 6: P0T6
  - Bit 5: P0T5
  - Bit 4: P0T4
  - Bit 3: P0T3
  - Bit 2: P0T2
  - Bit 1: P0T1
  - Bit 0: P0T0

- Address Offset $242: Data Direction, input or output
  - Bit 7: DDRT7
  - Bit 6: DDRT6
  - Bit 5: DDRT5
  - Bit 4: DDRT4
  - Bit 3: DDRT3
  - Bit 2: DDRT2
  - Bit 1: DDRT1
  - Bit 0: DDRT0

- Address Offset $243: Reduced Drive
  - Bit 7: R0RT7
  - Bit 6: R0RT6
  - Bit 5: R0RT5
  - Bit 4: R0RT4
  - Bit 3: R0RT3
  - Bit 2: R0RT2
  - Bit 1: R0RT1
  - Bit 0: R0RT0

- Address Offset $244: Pull up/down enable
  - Bit 7: PER7
  - Bit 6: PER6
  - Bit 5: PER5
  - Bit 4: PER4
  - Bit 3: PER3
  - Bit 2: PER2
  - Bit 1: PER1
  - Bit 0: PER0

Port M,S Pins

- Six bi-directional pins (PM5-PM0)
- General purpose port / Serial Port (SCI and SPI)
- Asynchronous SCI (serial communications interface) PS0 becomes RxD and PS1 becomes TxD
- Synchronous SPI (serial peripheral interface) PM2, PM3, PM4, PM5 become SPI functions
- CAN (controlled area network) PM0, PM1 for Rx and Tx data
Serial IO

- SPI (synchronous serial peripheral interface)
  - MOSI (control out, receiver in), PM2
  - MISO (control in, receiver out), PM3
  - SCK, clock, PM4
  - SS* (receiver select), PM5
  - With 25 MHz bus rate, maximal 12 MHz serial transfer (approx. 1 Mbyte/second)

Serial IO

- SCI (asynchronous serial communication interface)
  - RX data, PS0
  - TX data, PS1
  - 25 MHz bus speed: 38,400 baud, sends up to 11 bits per byte, 1 baud = 1 bit/second

Serial Communication

- Will need additional external electronics
- RS232 Maxim Driver Signal levels on the RS232 cable are 10V
- When the microcontroller sends data on Tx it will have to reach Rx on your COM port of your desktop computer
- Pin 3 on COM port is Rx.
- Pin 2 on COM port is Tx.
- Pin 5 is ground (goes on cable shield)

RS232 connectors

*Pin Insertion* side of the DB9F

1 2 3 4 5

6 7 8 9

"Pin Insertion" side of the DE9M

5 4 3 2 1

9 8 7 6
**MSCAN**
- Motorola Scalable Area Network 12, CAN2.0 A/B
- BOSCH 1991 standard
- Can attach multiple devices
- RX PM0
- TX PM1
- 0-8 bytes data length in a frame
- 1 Mbps
- Requires additional transceiver electronics

**In Circuit Debugging**
- Background Debug Module (BKGD pin)
- Single wire serial interface
- On chip hardware/firmware
- Hardware commands to read/write memory
- Software commands to read/write CPU resources / registers
- TRACE: execute one instruction from program than return to debug
- TAG instruction: execute until tagged instruction in execution queue (once CPU status is sent to host computer, microcontroller already has started executing the instruction => need to tag)

**Backgrnd Debug Module**