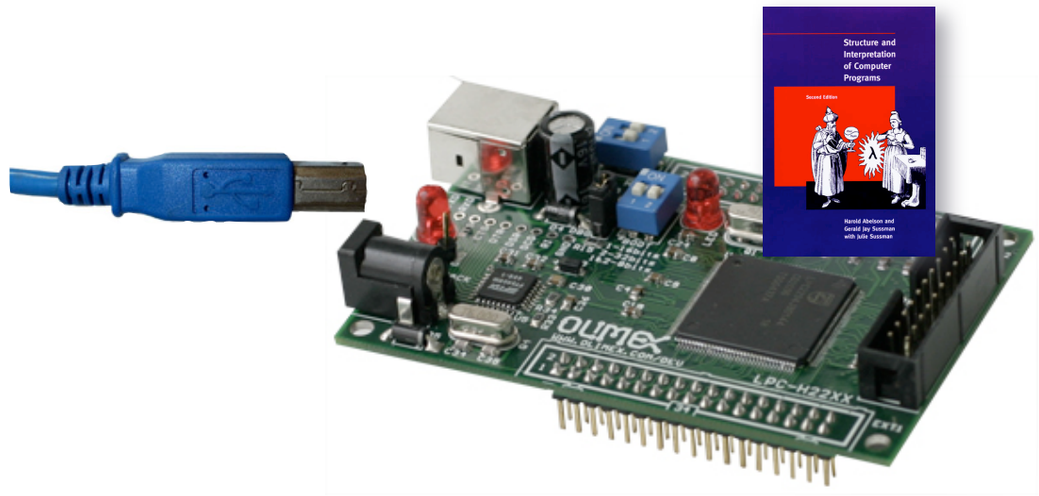
A Scheme Interpreter on a Microcontroller

repl accessible through serial communication



runs **Armpit Scheme** interpreter

+/- R5RS compliant

Hands on: have the board execute a factorial function

notice any non-R5RS compliance?

Essential µC-specific Scheme extensions

binary operations

binary literals

base to format 32-bit number in

```
(number->string (logior #b1100 #b1010) 2) ; -> "00000000000000000000000000001110"
(number->string (logxor #b1100 #b1010) 2) ; -> "00000000000000000000000000000110"
(number->string (logand #b1100 #b1010) 2) ; -> "00000000000000000000000000001000"
(number->string (lognot #b1100) 2)        ; -> "11111111111111111111111111110011"
(number->string (ash #b1100  4) 2)        ; -> "00000000000000000000000011000000"
(number->string (ash #b1100 -2) 2)        ; -> "00000000000000000000000000000011"
```

arithmetic shift

direction

Essential µC-specific Scheme extensions

reading and writing to and from memory

optional arguments

```
(read memory-location offset number-of-bytes)
(write value memory-location offset number-of-bytes)
```

base address — added to base address

example

(read #xE002800C 0) **=**

```
(define BASE #xE0028000)
(read  BASE #x0C)
```

Interfacing with the Physical World

microcontroller has **two general-purpose input-output (GPIO)** ports

each port has 32 pins

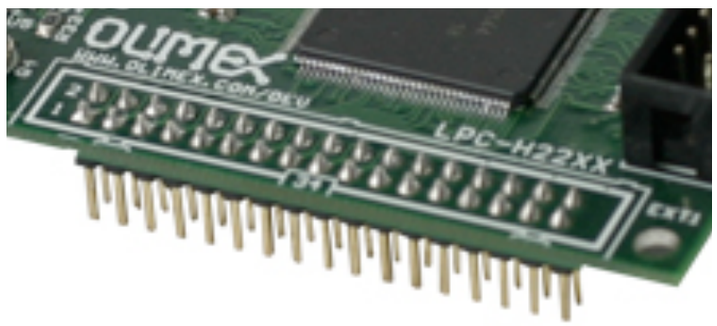
each pin corresponds to **set of registers**

32-bit *physical variables* in microcontroller

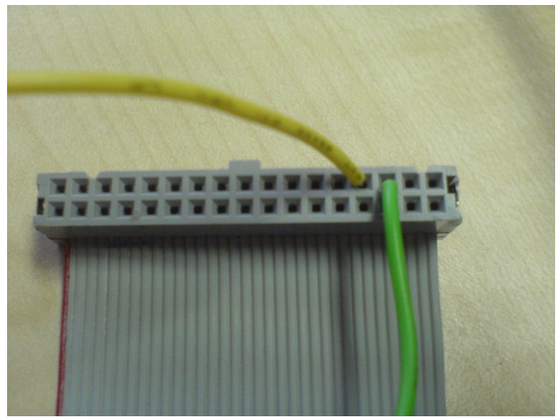
referred to by address

mapped to **EXT1 and EXT2** connectors on development board

listed in development board datasheet



2 extra pins on
flat ribbon to
provide power



Controlling Pins on Ports

base addresses of port

```
(define GPIO_0 #xE0028000) ; IO0 port

(define GPIO_1 #xE0028010) ; IO1 port
```

pin controlling registers

```
(define IOPIN #x00)  ; IOxPIN register offset

(define IOSET #x04)  ; IOxSET register offset

(define IODIR #x08)  ; IOxDIR register offset

(define IOCLR #x0C)  ; IOxCLR register offset
```

addresses and offsets can be found in the data sheet →

(cfr. references on last slide)

Table 68: GPIO Register Map

Generic Name	Description	Access	Reset Value	PORT0 Address & Name
IOPIN	GPIO Port Pin value register. The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction and mode. Activity on non-GPIO configured pins will not be reflected in this register.	Read Only	NA	0xE0028000 IO0PIN
IOSET	GPIO Port Output set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	Read/ Write	0x0000 0000	0xE0028004 IO0SET
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	Read/ Write	0x0000 0000	0xE0028008 IO0DIR
IOCLR	GPIO Port Output clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	Write Only	0x0000 0000	0xE002800C IO0CLR

Configuring Pin Direction

Table 72: GPIO Direction Register (IO0DIR - 0xE0028008, IO1DIR - 0xE0028018, IO2DIR - 0xE0028028, IO3DIR - 0xE0028038)

IODIR	Description	Value after Reset
31:0	Direction control bits (0 = INPUT, 1 = OUTPUT). Bit 0 in IO0DIR controls P0.0 ... Bit 31 in IO0DIR controls P0.31	0

Setting P0.06 to output

```
(define pin6 (ash 1 6))
(write pin6 GPIO_0 IODIR)
```

Everything else is input now!!

Using logical or to set P0.06 respecting previous settings

```
(define previous-dir (read port IODIR))
(write (logior previous-dir (ash 1 6)) GPIO_0 IODIR)
```

Abstracted away into functions

```
(define (set-dir! port pin)
  (write  (logior (read port IODIR) pin)
          port IODIR))
```

```
(define (set-dir-input! port pin)
  (write (logand (read port IODIR)
                 (lognot pin))
         port IODIR))
```

Producing High Value On Pin

Table 70: GPIO Output Set Register (IO0SET - 0xE0028004, IO1SET - 0xE0028014, IO2SET - 0xE0028024, IO3SET - 0xE0028034)

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit 0 in IO0SET corresponds to P0.0 ... Bit 31 in IO0SET corresponds to P0.31	0

Setting P0.06 to high = writing 1 << 6

```
(write (ash 1 6) gpio0 IOSET)
```

Does not change other pins

Abstracted away away into a function

```
(define (set-pin! port pin)
  (write pin port IOSET))
```

Setting a pin to low is very similar

```
(define (clear-pin! port pin)
  (write pin port IOCLR))
```

Exercise: Blinking LED

What do we need to do:

1. set target pin direction
2. set pin to high
3. wait for a while
4. set pin to low
5. wait for a while
6. repeat from 2

Exercise: Blinking LED

What do we need to do:

1. set target pin direction
2. set pin to high
3. wait for a while
4. set pin to low
5. wait for a while
6. repeat from 2


```
(define led-pin (ash 1 6))
```

```
(set-dir! GPIO_0 led-pin)
```

Exercise: Blinking LED

What do we need to do:

1. set target pin direction
2. set pin to high
3. wait for a while
4. set pin to low
5. wait for a while
6. repeat from 2

```
(define led-pin (ash 1 6))
```

```
(set-dir! GPIO_0 led-pin)
```

```
(set-pin! GPIO_0 led-pin)
```

Exercise: Blinking LED

What do we need to do:

1. set target pin direction
2. set pin to high
3. wait for a while
4. set pin to low
5. wait for a while
6. repeat from 2

```
(define led-pin (ash 1 6))
```

```
(set-dir! GPIO_0 led-pin)
```

```
(set-pin! GPIO_0 led-pin)
```

```
(clear-pin! GPIO_0 led-pin)
```

Exercise: Blinking LED

What do we need to do:

1. set target pin direction
2. set pin to high
3. wait for a while
4. set pin to low
5. wait for a while
6. repeat from 2

```
(define led-pin (ash 1 6))
```

```
(set-dir! GPIO_0 led-pin)
```

```
(set-pin! GPIO_0 led-pin)
```

```
(clear-pin! GPIO_0 led-pin)
```

```
(wait-us 1000000)  ; this will wait for that
                   ; many µsecs, one second
```

References

Introduction to Electronics

http://machineproject.com/2007/06/02/electronics-for-artists/

http://library.thinkquest.org/16497/intro/index.html

http://openbookproject.net//electricCircuits/

Armpit Scheme Interpreter

http://armpit.sourceforge.net/

Datasheet NXP LPC2214 Microcontroller

http://www.nxp.com/acrobat_download/usermanuals/UM_LPC2114_2124_2212_2214_2.pdf

Datasheet Olimex H-2214 Develoment Board

http://www.olimex.com/dev/lpc-h2214.html