How RoboBricks Were Developed

Wayne C. Gramlich

Wednesday, June 26, 2002

Problem

- Building robots is hard
- Mechanical, electrical, and software
- Real−time programming is hard

Solution

- Use modules
- Off load real−time to dedicated µC
- Simplify top level programming!
- Simplify electronics, too!

History

- Use '509's as hubs
- Processor non−integrated hub (RJ11)
- Processor combined hub (1×5 Header)

Final Architecture

- 4−wire (1×5 header) 0, +5, TTL up/down
- Asynch. 8N1 2400 baud
- Hierarchical master/slave
- Processor neutral
- Interrupt mode

PIC's

Part #	Pins	ROM	RAM	IO	A/D	Ser	I^2C	Cost (25)
'509	8	1K	41	6	0	0	0	$1.01
'505	14	1K	72	12	0	0	0	$1.26
'672	8	2K	128	6	4	0	0	$1.94
'F84	18	1K	68	13	0	0	0	$3.94
'F628	18	2K	224	16	0	1	0	$2.21
'F876	28	8K	368	22	5	1	1	$5.49

PIC Limitations

- '509, '505, '672 have no interrupts
- '509, '505, '672 have 2 level stack
- 1K ROM is not a lot
- Code and data banks are gnarly
- Only '628 and '876 have USART

µCL

- µCL = Micro Controller Language
- C−like
- Data types: bit, byte, string, byte array
- Does code/data banking (barely)
- Written in 12K lines of Tcl/Tk (ugh!)
- Excellent code generation
- Counts instructions!

Code Layout

- Main
- Get_Byte
- Send_Byte
- Delay
- Reset (Optional)

Get_Byte

```
procedure get_byte {
    arguments_none
    returns byte
    variable count byte
    variable char byte

    while (serial_in) {
        call delay()
    }

    call delay()
    call delay()
    call delay()
    char := 0
    count_down count 8 {
        call delay()
        char := char >> 1
        if (serial_in) {
            char@7 := 1
        }
        call delay()
        call delay()
        nop extra_instructions_per_bit - 7
    }
```

```
    call delay()
    call delay()
    return char
}
```

Get_Byte Assembly

```
get_byte:
get_byte__464while__continue:
        btfss serial_in__byte,serial_in__bit
        goto get_byte__464while__break
        call delay
        goto get_byte__464while__continue

get_byte__464while__break:
        call delay
        call delay
        call delay
        clrf get_byte__char

        movlw 8
        movwf get_byte__count
get_byte__483_loop:
        call delay
        bcf c___byte,c___bit
        rrf get_byte__char,f
        btfsc serial_in__byte,serial_in__bit
        bsf get_byte__char,7
        call delay
        call delay
        decfsz get_byte__count,f
```

```
        goto get_byte__483_loop

get_byte__483_done:
        call delay
        call delay
        movf get_byte__char,w
        movwf get_byte__0return__byte
        retlw 0
```

Delay Code

```
procedure delay {
    arguments_none
    returns_nothing
    uniform_delay delay_instructions
    variable counter byte
    variable temp0 byte
    variable temp1 byte

    watch_dog_reset
    if (ir_in) {
        if (counter != 0) {
            # We've got a pulse:
            if (counter >= 12) {
                # We've got a start:
                byte0 := temp0
                byte1 := temp1
                temp0 := 0
                temp1 := 0
            } else {
                # Shift 10 bits:
                temp0 := temp0 >> 1
                temp0@7 := temp1@0
                temp1 := temp1 >> 1
                if (counter >= 6) {
```

```
                        temp1@3 := 1
                    }
                }
            }
            counter := 0
        } else {
            counter := counter + 1
        }
}
```

Main Code

```
procedure main{
    arguments_none
    returns_never
    variable command byte

    call reset()
    loop_forever {
        command := get_byte()
        switch (command >> 6) {
          case 0 {
            switch ((command >> 3) & 7) {
              case 0 {
                switch (command & 7) {
                  case 0 {
                    # Command 0000 0000:
                  }
                  #...
                  case 7 {
                    # Command 0000 0111:
                  }
                }
              }
              #...
              case 7 {
```

```
                    # ....
                  }
                }
              }
              default 3 {
                #...
              }
            }
        }
}
```

Shared Commands

- Glitch (0xFF = Increment; 0xFE = Read)
- ID (0xFD = Reset; 0xFC = Read)
- Clock adjust (0xFB = Pulse, 0xFA = Read)
- Clock adjust (0xF9 = Incr, 0xF8 = Decr.)
- 4/6−bits of clock adjust ('509/'509A)

Id

- Header (8−bytes)
- Unique ID (16−bytes of random bits)
- Brick String (variable length)
- Vendor (variable length)

Interrupt

- Initialize
- Enable
- Trigger
- Detect
- Respond

Debugging

- `Crash and burn' debugging
- Erasing using UV is a pain
- Use unused pins for `heartbeat' signals
- Buggy µCL compiler did not help
- Code/data bank switching is gnarly
- Should have invested in an ICE (Big $$$)
- Wrote an emulator to help debug µCL compiler
- Wrote a test suite for µCL compiler
- Tcl/Tk is a bad choice for writing a compiler

Boards

- Used rapid prototype services (APC, Olimex)
- Gerber/excellon files really work
- Used private PCB layout software (HobECAD)
- Solder bridges happen when no solder mask
- PCB milling was close
- Constantly undersizing holes
- Patching boards was pretty easy

Web Site

- Very successful
- http://web.gramlich.net/projects/robobricks/
- 24/7/365 availability
- Steady trickle of interest via search engines
- Added PDF files for printer friendly

Harness

- Have harness board
- Have manual test command interpreter
- Each board has a test suite
- Each RoboBrick has internal state commands

Lessons

- Weekly meetings made a big difference
- Invest in an ICE
- Fewer boards sooner better than many boards later
- Way too much preselling
- Moore's Law ('F628, AVR's, etc.)
- Servo board was ***very*** hard
- Don't write compilers in Tcl/Tk

Current Status

- Taking orders for 6 boards tonight
- (AnalogIn4, BS2Hub8, InOut10, Light4, Servo4, SonarDT1)
- ~$35 for 6 boards
- ~$35 for parts
- ~$25 for SRF04
- Everything `at cost'
- Full support from Wayne and Bill
- More opportunities coming

Future

- More boards; 6 new ones every few months
- Newer processors
- Surface mount is getting hard to avoid
- Lower prices as volumes ramp up
- More clubs: TCRG, SRS, DPRG, TRCY, RSA, ...
- Magazine article (Circuit Cellar? Nuts & Volts?)
- Book?
- License deal? Acroname? Robot Store? Sombody else?
- World domination!!!