```
'********************************************************
'*                                                      *
'*              Propeller II ROM Booter                 *
'*                                                      *
'*              Version 0.1                             *
'*                                                      *
'*              11/01/2012                              *
'*                                                      *
'********************************************************

CON

  rx_pin = 91
  tx_pin = 90
  spi_cs = 89
  spi_ck = 88
  spi_di = 87
  spi_do = 86

  base = $E80


DAT
'
'
' Version (@$000)
'
                        byte    "Prop2.0 "
'
'
' Shut down (@$008)
'
                        org

                        clkset  h001+offset             'set clock to rc slow
                        cogstop h200+offset             'stop cog0
offset
'
'
' Entry, read fuses (@$010)
'
                        org

                        reps    #256,@:fuse             'ready to read 256 fuses
                        setport #rx_pin                 'set rx_pin port for booting

                        cogid   fuse_read       nr      'read fuses (172 fuses + 84 zeros)
                        cogid   fuse_read       nr,wc   '(last iteration initializes cnt to
$00000000_00000001)
                        add     fuse_read,#1
                        test    fuse_read,#$1F  wz
:fusex                  rcr     fuses,#1
:fuse   if_z            add     :fusex,h200

                        cogid   spi_read        nr      'disable fuses and enable cnt
(spi_read[10..0] = 0)
'
```

```
'
' Attempt to boot from serial
'
                        jnp     monitor_ptr,#boot_flash 'if rx_pin is low, skip serial and
                        boot from flash

                        call    #rx_bit                 'measure low rx calibration pulses
                        (host $F9 -> %1..010011111..)
                        mov     threshold,delta         'and calculate threshold
                        call    #rx_bit                 '(any timeout results in flash boot)
                        add     threshold,delta
h001                    shr     threshold,#1            '(9 lsb's are $001)

                        mov     count,#250              'ready to receive/verify 250 lfsr bits
:lfsrin                 call    #rx_bit                 'receive bit ($FE/$FF) into c
                        test    lfsr,#$01       wz      'get lfsr bit into nz
        if_c_eq_z       jmp     #boot_flash             'if mismatch, boot from flash
                        test    lfsr,#$B2       wc      'advance lfsr
                        rcl     lfsr,#1
                        djnz    count,#:lfsrin          'loop for next bit in

                        mov     count,#250+8            'ready to transmit 250 lfsr bits + 8
                        version bits
:lfsrout                cmp     count,#8        wz      'if last 8 bits, set lfsr so that
version will be output
        if_z            mov     lfsr,#$52               '$52 results in version $20 being
        sent (%00000100)
                        test    lfsr,#$01       wz      'get lfsr/version bit into nz, z=1
                        on last iteration
                        call    #wait_rx                'wait for rx low (convey incoming
                        $F9 on rx_pin to $FE/$FF on tx_pin)
                        clrp    #tx_pin                 'make tx low
                        call    #wait_rx                'wait for rx high
                        setpnz  #tx_pin                 'make tx lfsr/version bit
                        call    #wait_rx                'wait for rx low
                        setp    #tx_pin                 'make tx high
                        call    #wait_rx                'wait for rx high
                        test    lfsr,#$B2       wc      'advance lfsr
                        rcl     lfsr,#1
                        djnz    count,#:lfsrout         'loop for next bit out

                        jmp     #load                   'serial handshake done, attempt to
                        load from serial (z=1)
'
'
' Wait for rx low/high - if timeout, attempt to boot from flash
'
wait_rx                 getcnt  time                    'ready timeout
                        add     time,timeout

:waitpxx                waitpne rx_mask,rx_mask wc      'wait for rx low/high with timeout

                        notb    :waitpxx,#23            'toggle waitpeq/waitpne

wait_rx_ret     if_nc   ret                             'return if not timeout (boot_flash
follows)
'
```

```
'
' Attempt to boot from flash
'
boot_flash              mov     count,#4                'ready for 3 resets and 1 read command

:cmd                    setp    #spi_cs                 'spi_cs high
                        clrp    #spi_ck                 'spi_ck low

                        reps    #32,@:bit               'ready for 32 command bits
                        clrp    #spi_cs                 'spi_cs low

                        cmpr    count,#1        wc      'first 3 commands = $FF_FF_FF_FF
                        (reset)
        if_nc           rol     spi_read,#1     wc,wz   'last command = $03_00_00_00 (read
        from 0), z=0
                        setpc   #spi_di
                        setp    #spi_ck                 'cycle spi_ck
:bit                    clrp    #spi_ck

                        djnz    count,#:cmd             'loop for next spi command
'
'
' Load from serial (z=1) or flash (z=0)
'
load                    setptra loader_pgm              'load loader into base+$000..$7DF,
HMAC into base+$7E0..$7FF

                        mov     count,h200              'ready to input $200 longs
:long                   mov     bits,#32                'ready to input 32 data bits

:bit    if_z            call    #rx_bit                 'input serial bit (serial mode)
        if_nz           getp    #spi_do         wc      'input spi_do (flash mode)
        if_nz           setp    #spi_ck                 'high spi_ck (flash mode)
        if_nz           clrp    #spi_ck                 'low spi_ck (flash_mode)
                        rcl     data,#1                 'shift bit into long
                        djnz    bits,#:bit              'loop, adequate time for next flash
                        bit

                        wrlong  data,ptra++             'store long in hub ram
                        (ptra=base+$800 after)
                        djnz    count,#:long            'loop for next long (count=0 after)
'
'
' Compute loader HMAC signature for loader authentication
'
' base+$000..$7DF = loader                              ($1F8 longs)
' base+$7E0..$7FF = loader HMAC signature               (8 longs)
' base+$800..$81F = fuses, 1st half are HMAC key        (8 longs)
' base+$820..$83F = proper HMAC signature               (8 longs)
' base+$840..$843 = sha256 command interface            (1 long)
'
                        reps    #8,#1                   'store 128-bit key + 44 extra fuses
                        + 84 zero bits
                        setinda fuses                   'into base+$800..$81F
                        wrlong  inda++,ptra++           '(ptra = base+$820, afterwards)

                        wrlong  count,sha256_ptr        'clear sha256 command
                        setcog  #1                      'launch sha256 in cog1
                        coginit sha256_pgm,sha256_ptr

                        setinda begin_hmac              'do sha256 commands to compute
                        proper loader hmac
                        mov     count,#3                'ready for 3 commands: begin_hmac,
                        hash_bytes, read_hash
:cmd                    wrlong  inda++,sha256_ptr       'set command
:wait                   rdlong  data,sha256_ptr wz      'wait for command done
                        tjnz    data,#:wait
                        djnz    count,#:cmd             'loop for next command (count=0, z=1
                        after)

                        cogstop h001                    'done with sha256, stop cog1
'
'
' If loader authenticates, run it
'
                        reps    #8,@:cmp                'verify loader hmac signature (z=1
                        on entry)
                        setcog  #0                      'ready to relaunch cog0 with
                        loader/monitor

                        rdlong  bits,ptra[-$10]         'get loader hmac signature long
                        rdlong  data,ptra++             'get proper hmac signature long
:cmp    if_z            cmp     bits,data       wz      'compare, z=1 if authenticated

        if_z            coginit loader_pgm,loader_ptr   'if loader authenticated, relaunch
        cog0 with loader
'
'
' Authentication failed, hide fuses and clear memory
'
                        reps    #$20000/8,@:clr         'clear all memory
                        cogid   monitor_pgm     nr      'hide fuses (bit 10 set)

                        wrlong  count,ptra++            '(count=0)
:clr                    wrlong  count,ptra++
'
'
' If key <> 0, shut down - else, monitor
'
                        or      fuses+0,fuses+1 wz      'check if 128-bit key = 0
        if_z            or      fuses+2,fuses+3 wz

        if_nz           mov     monitor_pgm,#$008       'if key <> 0, shut down

                        coginit monitor_pgm,monitor_ptr 'relaunch cog0 with shut down or
                        monitor
'
'
' Receive bit (c) - compare incoming pulse to threshold
'
rx_bit                  call    #wait_rx                'wait for rx low
                        getcnt  delta                   'get time
                        call    #wait_rx                'wait for rx high
                        subcnt  delta                   'get time delta

                        cmp     delta,threshold wc      'compare time delta to threshold

rx_bit_ret              ret
'
'
' Constants
'
fuse_read               long    $200                    '(gets modified to $300)
timeout                 long    20_000_000 / 1000 * 150 '150ms @20MHz (rcfast)
rx_mask                 long    1 << (rx_pin & $1F)
lfsr                    long    "P"
spi_read                long    $03_000000
h200                    long    $200

begin_hmac              long    1<<30 + (($004<<2)-1)<<17 + base+$800   'begin_hmac, loads
key at base+$800 (4 longs)
hash_bytes              long    2<<30 + (($1F8<<2)-1)<<17 + base+$000   'hash_bytes, hashes
message at base+$000 ($1F8 longs)
read_hash               long    3<<30                     + base+$820   'read_hash, writes
hash at base+$820 (8 longs)

sha256_pgm              long    $1D0                    'sha256 program address
sha256_ptr              long    base+$840               'sha256 parameter (points to command)

loader_pgm              long    base+$000               'loader program address
loader_ptr              long    base+$800               'loader parameter (points to fuses)

monitor_pgm             long    $55C+$1B4               'monitor program address
monitor_ptr             long    tx_pin<<9 + rx_pin      'monitor parameter (conveys pins)
'
'
' Variables
'
fuses                   res     8
count                   res     1
bits                    res     1
data                    res     1
time                    res     1
delta                   res     1
threshold               res     1
```