

```
{
```

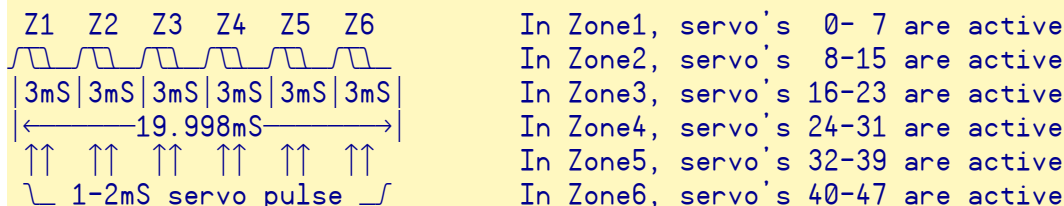
```
*****
Control up to 144-Servos      Version1.2      04-24-2006
*****
Coded by Beau Schwabe (Parallax).
*****
```

Features:

- 1) Only 1 COG used
- 2) Control up to 144-Servos
- 3) Set and forget
- 4) Wide input range of acceptable pulse widths (50uS to 3300uS)
- 5) (3x6x8 = 144 total servos)
 - Three Groups
 - Six Zones
 - Eight servos per Zone
- 6) 20mS (19.998mS) period for each servo.
- 7) Two I/O's available for communication

Theory of Operation:

Each servo requires a pulse that varies from 1mS to 2mS with a period of 20mS. If you break down the 20mS period into six groups or Zones of 8 servos, each Zone has a period of 3.333mS... Multiplied by six Zones we get 19.998mS (close enough to 20mS). By using Zones, we can ensure that at any given moment in time, a maximum of only 8 servos are receiving a pulse or turned "on" at a time. This reduces the total amount of required current to your system. Within each Zone, initially ALL servos (groups of 8) turn "on", dropping "off" one by one when the required time has elapsed corresponding to the assigned pulse width for that servo, thus ALL servos within a zone complete their assigned pulse value within a 2mS period.



Note: ALL 74xx573's are powered from a 3.3V supply. Run separate power and ground to servo groups 1-48, 49-96, and servo groups 97-144. Do NOT daisy between groups. Each group of 48 should have it's own power supply

Note: P0-P40 below are referencing the 40-Pin DIP version of the Propeller

Connection for Servos 1-48:

P0-P7 on the Propeller connect via a buss to the data inputs of six 74xx573's. ALL output enable's (OE) on each of the 74xx573 are grounded.

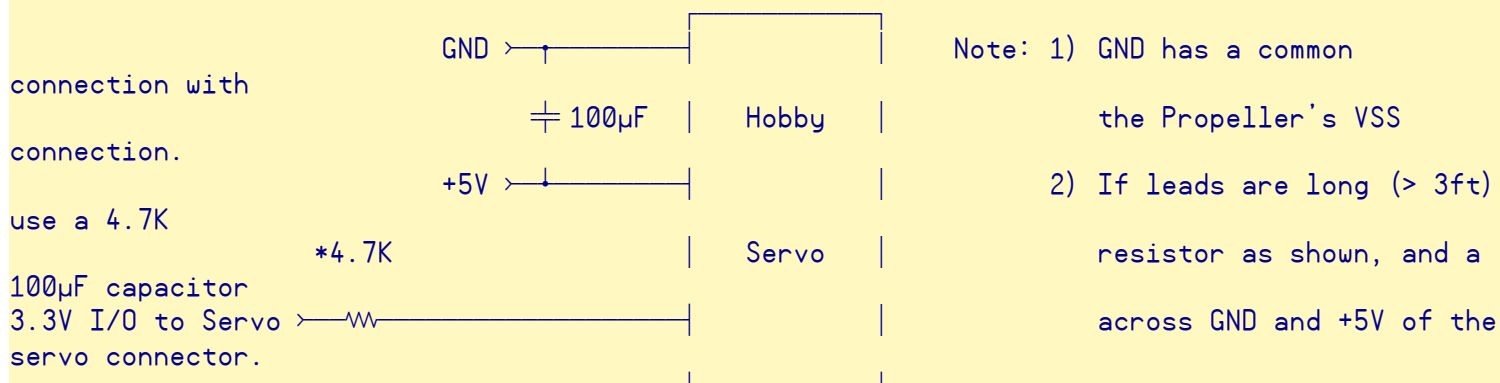
Each latch enable (LE) on 74xx573's 1 to 6 connect to P33-P38 on the Propeller.

Connection for Servos 49-96:

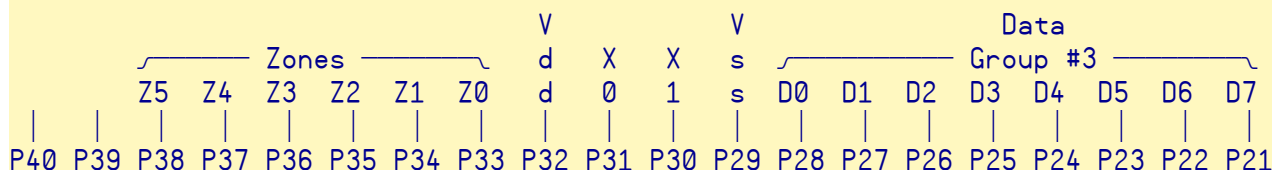
P12-P20 on the Propeller connect via a buss to the data inputs of six 74xx573's. ALL output enable's (OE) on each of the 74xx573 are grounded. Each latch enable (LE) on 74xx573's 1 to 6 connect to P33-P38 on the Propeller.

Connection for Servos 97-144:

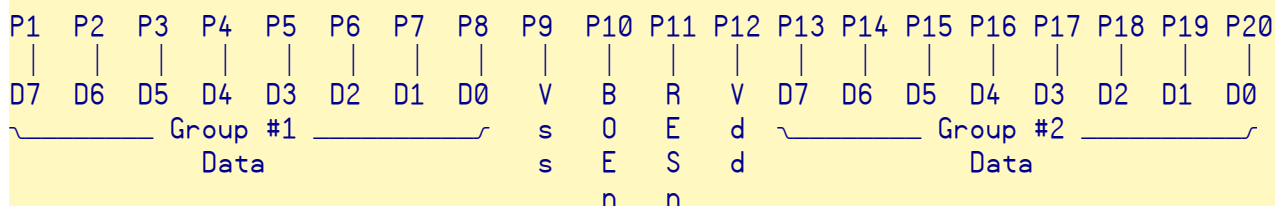
P21-P28 on the Propeller connect via a buss to the data inputs of six 74xx573's. ALL output enable's (OE) on each of the 74xx573 are grounded. Each latch enable (LE) on 74xx573's 1 to 6 connect to P33-P38 on the Propeller.



Propeller 144 Servo Pinout:



40-Pin DIP Propeller



Revision History:

- Version 1.0 - Original concept to drive up to 144 servo's
- Version 1.1 - Added cnt rollover detection to prevent "glitch"
 - Fixed bug when ALL data and zone information was written to port at once. The 573's would not latch the data properly when data was written to the port simultaneously. This was fixed, by enabling the latch pulse first, followed by the servo data.
- Version 1.2 - Small modification to the way zone and port data are handled. Update 'mov'

vs. 'movi' data collisions
 resulting in erratic output on zones above zone #1.

}}

CON

```

    _1uS = 1_000_000           'Divisor for 1
uS                               '3.333mS (
    ZonePeriod = 3_333
approx. 1/6th of typical servo period of 20mS)
    
```

VAR

```

    long      IOdirection
    long      ZoneClocks
    long      ServoData[144]   'Reserve 144
long variables (0-143) for Servo Pulse Width information
    
```

PUB Start (Mode)

```

    if Mode == 0      '-Zones-  -Data-  -Data-  -Data-
    IOdirection := %00111111_00000000_00000000_11111111   ' 48 servos or
less
    if Mode == 1      '-Zones-  -Data-  -Data-  -Data-
    IOdirection := %00111111_00000000_11111111_11111111   ' 96 servos or
less
    if Mode == 2      '-Zones-  -Data-  -Data-  -Data-
    IOdirection := %00111111_11111111_11111111_11111111   '144 servos or
less

    ZoneClocks := clkfreq / _1uS * ZonePeriod           'calculate #
of clocks per ZonePeriod
    cognew( @ServoStart, @IOdirection)
    
```

PUB Set (Servo, Width)

```

value                               'Set Servo
    Width      :=      50 #> Width <# 3300           'limit Width
value between 50uS and 3300uS
    Servo      :=      0 #> Servo <# 143           'limit Servo
value between 0 and 143
    ServoData[Servo] :=      clkfreq / _1uS * Width   'calculate #
of clocks for a specific Pulse Width 0-143
    
```

DAT

```

' *****
' * Assembly language *
' *****
    
```

org

```

-----
ServoStart      mov      Index,      par           'Pass address
of First argument
direction of pins      rdlong  _IOdirection,      Index           'Get IO
address of next argument      add      Index,      #4           'Increment to
Clocks per Zone      rdlong  _ZoneClocks,      Index           'Get Number of
                        add      Index,      #4           'Increment to
    
```

```

address of ServoData Group 1 ( 1- 48 servos)
      mov      ServoGroup1,      Index      'Load address
of ServoData Group 1
      add      Index,            #192        'Increment to
address of ServoData Group 2 ( 49- 96 servos)
      mov      ServoGroup2,      Index      'Load address
of ServoData Group 2
      add      Index,            #192        'Increment to
address of ServoData Group 3 ( 97-144 servos)
      mov      ServoGroup3,      Index      'Load address
of ServoData Group 3
      mov      dira,             _I0direction 'Set IO
directions corresponding to selected Bank
-----
MainServoLoop      mov      ZoneEnable,      #1        'Initialize
Zone value
Zone Offset      mov      ZoneOffset,      #0        'Initialize
number of Zones   mov      ZoneCount,      #6        'Initialize
ZoneCore          mov      SyncPoint,      cnt      'Create a Sync
Point with the system counter for current Zone
detect cnt rollover
      mov      temp,            SyncPoint    'No "Glitch"...
was to occur, at this point temp would be less than _ZoneClocks
      add      temp,            _ZoneClocks 'If a rollover
      sub      temp,            _ZoneClocks
      if_C     jmp      #ZoneCore          wc      'If rollover
detected, wait a bit and get new sync point
ZoneLoop          mov      ServoPulseData, ZoneEnable 'Reset
ServoPulseData
      mov      ServoAddress,      ServoGroup3 'Move address
pointer of servo data group 3 into Servo
      call     #CheckServos        'Get Servo
group 3 data (left Shift 8-bits into ServoData)
      xor      ServoPulseData,    #$FF      'Invert
ServoByte variable
      mov      ServoAddress,      ServoGroup2 'Move address
pointer of servo data group 2 into Servo
      call     #CheckServos        'Get Servo
group 2 data (left Shift 8-bits into ServoData)
      xor      ServoPulseData,    #$FF      'Invert
ServoByte variable
      mov      ServoAddress,      ServoGroup1 'Move address
pointer of servo data group 1 into Servo
      call     #CheckServos        'Get Servo
group 1 data (left Shift 8-bits into ServoData)
      xor      ServoPulseData,    #$FF      'Invert
ServoByte variable
data to port      mov      temp,            ZoneEnable 'Send zone
      shl     temp,            #1
      and     temp,            #%001111110
      mov     temp2,          ServoPulseData
      shr     temp2,            #23
      and     temp2,            #%000000001
      add     temp,            temp2
      movi    outa,            temp

```

```

data to port          mov     temp,          ServoPulseData      'Send servo
Zone is complete...  mov     outa,          temp
                    mov     temp,          _ZoneClocks        'Determine if
                    add     temp,          SyncPoint
                    sub     temp,          cnt                wc
                    if_NC  jmp     #ZoneLoop        '...if the "C
Flag" is not set stay in the current Zone
Zone Offset pointer  add     ZoneOffset,    #32                'Increment
Zone enable data.   shl     ZoneEnable,    #1                    'Left Shift
ZoneCount; Jump to ZoneLoop if not "0"
                    djnz    ZoneCount,     #ZoneCore          'Decrement
                    jmp     #MainServoLoop
;-----
CheckServos          mov     tempCount,     #8
                    mov     tempIndex,     ServoAddress
                    add     tempIndex,     ZoneOffset
                    rdlong  ServoWidth,    tempIndex          'Get Servo Data
                    add     ServoWidth,    SyncPoint          'Determine
                    sub     ServoWidth,    cnt                wc 'subtract
                    rcl     ServoPulseData, #1                'Rotate "C
Flag" left into ServoData
                    add     tempIndex,     #4
                    djnz    tempCount,     #ServoLoop        'Decrement
TempCount; Jump to ServoLoop if not "0"
CheckServos_RET     ret
;-----
Index               res     1
ServoGroup1         res     1
ServoGroup2         res     1
ServoGroup3         res     1
ServoPulseData     res     1
ZoneOffset          res     1
ZoneEnable          res     1
ZoneCount           res     1
SyncPoint           res     1
ServoAddress        res     1
temp                res     1
temp2               res     1
tempCount           res     1
tempIndex           res     1
ServoWidth          res     1
;-----
_I0direction        res     1
_ZoneClocks         res     1

```