```
{{

*******************************************************************
 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.

 Z1  Z2  Z3  Z4  Z5  Z6         In Zone1, servo's  0- 7 are active
/‾\_/‾\_/‾\_/‾\_/‾\_/‾\_        In Zone2, servo's  8-15 are active
|3mS|3mS|3mS|3mS|3mS|3mS|       In Zone3, servo's 16-23 are active
|←──────19.998mS───────→|       In Zone4, servo's 24-31 are active
 ↑↑  ↑↑  ↑↑  ↑↑  ↑↑  ↑↑         In Zone5, servo's 32-39 are active
 \_ 1-2mS servo pulse _/        In Zone6, servo's 40-47 are active

Note: ALL 74xx573'a 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.


                                           ┌───────────┐
                           GND >─┬────────┤           │    Note: 1) GND has a common
connection with
                                 ╪ 100µF  │   Hobby   │             the Propeller's VSS
connection.
                           +5V >─┴────────┤           │          2) If leads are long (> 3ft)
use a 4.7K
                      *4.7K               │   Servo   │             resistor as shown, and a
100µF capacitor
3.3V I/O to Servo >──/\/\/────────────────┤           │             across GND and +5V of the
servo connector.
                                           └───────────┘


Propeller 144 Servo Pinout:

                                 V       V               Data
        ┌────── Zones ──────┐   d   X   X   s  ┌────────── Group #3 ────────┐
        Z5  Z4  Z3  Z2  Z1  Z0  d   0   1   s  D0  D1  D2  D3  D4  D5  D6  D7
 │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │
P40 P39 P38 P37 P36 P35 P34 P33 P32 P31 P30 P29 P28 P27 P26 P25 P24 P23 P22 P21


                                40-Pin DIP Propeller


P1  P2  P3  P4  P5  P6  P7  P8  P9  P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20
 │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │
D7  D6  D5  D4  D3  D2  D1  D0  V   B   R   V  D7  D6  D5  D4  D3  D2  D1  D0
└───────── Group #1 ──────────┘   s   O   E   d  └──────── Group #2 ─────────┘
               Data                s   E   S   d              Data
                                       n   n

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
        ZonePeriod = 3_333                                                       '3.333mS (
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)                                                            'Set Servo
value
      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
                        rdlong  _IOdirection,           Index                    'Get IO
direction of pins
                        add     Index,                  #4                       'Increment to
address of next argument
                        rdlong  _ZoneClocks,            Index                    'Get Number of
Clocks per Zone
                        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,                   _IOdirection            'Set IO
directions corresponding to selected Bank
'-----------------------------------------------------------------------------------------------
--------------------------------------------------
MainServoLoop           mov     ZoneEnable,             #1                      'Initialize
Zone value
                        mov     ZoneOffset,             #0                      'Initialize
Zone Offset
                        mov     ZoneCount,              #6                      'Initialize
number of Zones
ZoneCore                mov     SyncPoint,              cnt                     'Create a Sync
Point with the system counter for current Zone
                        mov     temp,                   SyncPoint               'No "Glitch"...
 detect cnt rollover
                        add     temp,                   _ZoneClocks             'If a rollover
was to occur, at this point temp would be less than _ZoneClocks
                        sub     temp,                   _ZoneClocks           wc
              if_C      jmp     #ZoneCore                                       '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
                        mov     temp,                   ZoneEnable              'Send zone
data to port
                        shl     temp,                   #1
                        and     temp,                   #%001111110
                        mov     temp2,                  ServoPulseData
                        shr     temp2,                  #23
                        and     temp2,                  #%000000001
                        add     temp,                   temp2
                        movi    outa,                   temp
                        mov     temp,                   ServoPulseData          'Send servo
data to port
                        mov     outa,                   temp
                        mov     temp,                   _ZoneClocks             'Determine if
Zone is complete...
                        add     temp,                   SyncPoint
                        sub     temp,                   cnt                 wc
              if_NC     jmp     #ZoneLoop                                       '...if the "C
Flag" is not set stay in the current Zone
                        add     ZoneOffset,             #32                     'Increment
Zone Offset pointer
                        shl     ZoneEnable,             #1                      'Left Shift
Zone enable data.
                        djnz    ZoneCount,              #ZoneCore               'Decrement
ZoneCount; Jump to ZoneLoop if not "0"
                        jmp     #MainServoLoop
'-----------------------------------------------------------------------------------------------
--------------------------------------------------
CheckServos             mov     tempCount,              #8
                        mov     tempIndex,              ServoAddress
                        add     tempIndex,              ZoneOffset
        ServoLoop       rdlong  ServoWidth,             tempIndex               'Get Servo Data
                        add     ServoWidth,             SyncPoint               'Determine
system counter location where pulse should end
                        sub     ServoWidth,             cnt                 wc 'subtract
system counter from ServoWidth ; write result in C flag
                        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
'-----------------------------------------------------------------------------------------------
--------------------------------------------------
_IOdirection            res     1
_ZoneClocks             res     1
```