

CON

```

1  _CLKMODE = XTAL1 + PLL16X      'Set to ext crystal, 16x PLL, 80MHz Clock
2  _XINFREQ = 5_000_000          'Frequency on XIN pin is 5 MHz
3
4
5  '----- Hardware bedingte Konstanten -----
6  x_0 = 60.0                    ' Mittelpunkt des Arbeitsradius in x
7  y_0 = 0.0                     ' Mittelpunkt des Arbeitsradius in y
8  h_0 = 80.0                    ' Systemhöhe
9
10 l_should = 20.0                ' Abstand Servos Schulter/Hüfte x'
11 l_should_2 = 400.0            ' x'^2
12
13 l_fuss = 120.0                 ' Länge vom Fuss (b)
14 b_2 = 14400.0                 ' (l_fuss)^2
15
16 l_schenk = 72.0                ' Länge vom Oberschenkel
17 a_2 = 5184.0                  ' (l_schenk)^2
18
19 a_2_min_b_2 = -9216.0          ' a^2 - b^2          !!! mit xxx.0 angeben
20 a_2_plus_b_2 = 19584.0        ' a^2 + b^2          !!! mit xxx.0 angeben
21 ax2 = 144.0                   ' a * 2              !!! mit xxx.0 angeben
22 axbx2 = 17280.0               ' a * b * 2         !!! mit xxx.0 angeben
23
24 wink_fuss_off = 17.0           ' Winkeländerung bedingt durch Fuss-Krümmung (35mm) in Grad
!!! mit xxx.0 angeben
25
26
27 '----- Servo-Offests -----
28 y_off_A = 20.0
29 y_off_B = 0.0
30 y_off_C = -20.0
31 y_off_D = -20.0
32 y_off_E = 0.0
33 y_off_F = 20.0
34 x_off_A = 90.0
35 x_off_B = 90.0
36 x_off_C = 90.0
37 x_off_D = -90.0
38 x_off_E = -90.0
39 x_off_F = -90.0
40
41

```

Var

```

42 '-----Servo-Puls-Erzeugung-----
43 LONG new_peri_r,PinStart_r,pin_r,high_pulse_r[9]      'Reihenfolge nicht ändern da
44 Pointer in Move-Servo darauf zu greifen
45 LONG new_peri_l,PinStart_l,pin_l,high_pulse_l[9]      'DO NOT TOUCH
46
47 '-----Winkelberechnungen-----
48 LONG x_delta,y_delta,h_delta
49 LONG x_neu, y_neu, h_neu
50 LONG radius, c_1, c_2
51 LONG wink_fuss, wink_schult, wink_schult_temp, wink_hueft
52
53 LONG periode, target_pos
54 byte i,j,x
55
56 '-----Pulslängen-----
57 LONG A[3],B[3],C[3],D[3],E[3],F[3]

```

```

58 '-----Geradlinige Bewegung-----
59 LONG phi, step_length,step_high,high_start
60
61
62 LONG anzahl_steps,step_aktuell,step_delta,x_min[6],y_min[6],x_max[6],y_max[6]
63 LONG step_x[6],step_y[6] ,step_position,step_low,step_high,aktuel_step_pos,ak_st_pos_f,
anzahl_steps_f ,phase,repeat_step
64
65 LONG stack_1[20],stack_2[30]
66
67 LONG l_A,l_B,l_C,l_D,l_E,l_F
68 LONG x_phi,y_phi,large_r,center_x_f,center_y_f
69 LONG y_offset, x_offset
70 LONG delta_winkel, schritt_winkel, winkel_pro_step
71 LONG phi_A,phi_B,phi_C,phi_D,phi_E,phi_F
72 LONG ri_A,ri_B,ri_C,ri_D,ri_E,ri_F
73 OBJ
74 math : "Float32Full"
75
76 pub main
77 math.start
78 init
79
80 coginit(6,@moveServo,@new_peri_r)
81 coginit(7,@moveServo,@new_peri_l)
82
83 ' move_trans(0.0,3) ' phi[float] und Steps[int] (wenn Steps = 0 dann wird nur
Anfangsposition angefahren)
84 ' move_rotation(0,300,80.0) 'x-Koordinate, y-Koordinate, Epsilon
85
86 '-----Eigenes MAIN Bewegung Drehen --> Gerade --> Drehen -----
-
87 dira[0]~
88 dira[1]~~
89 dira[2]~~
90
91 move_rotation(0,0,0.1)
92
93 repeat
94 outa[1]~~
95 if ina[0] == 1
96 outa[1]~
97 quit
98
99
100 move_trans(90.0,3) ' phi[float] und Steps[int]
101 repeat 100000
102 move_rotation(0,400,270.0)
103 repeat 100000
104 repeat
105 '-----
106
107 pub init 'Initalisieren
108
109 x_delta := 0.0 '!!! mit xxx.0 angeben
110 y_delta := 0.0 '!!! mit xxx.0 angeben
111 h_delta := 0.0 '!!! mit xxx.0 angeben
112
113 A[0]:=1400

```

```

114 A[1]:=1600
115 A[2]:=1500
116
117 B[0]:=1500
118 B[1]:=1550
119 B[2]:=1500
120
121 C[0]:=1500
122 C[1]:=1650
123 C[2]:=1500
124
125 D[0]:=1450
126 D[1]:=1500
127 D[2]:=1450
128
129 E[0]:=1500
130 E[1]:=1550
131 E[2]:=1500
132
133 F[0]:=1450
134 F[1]:=1450
135 F[2]:=1500
136
137 '-----Feste-Parameter-----
138 periode := ((clkfreq/1000000)*20000) 'länge der gesamten periode
139 PinStart_r:=|<16
140 PinStart_l:=|<7
141 dira[7..15]~~
142 pin_l:= dira
143 dira[16..24]~~
144 pin_r:= dira
145
146 setpulse
147
148 step_length := 30.0 'in [mm] --> Radius Ri in float
149 hight_start := 50 'h
150 step_hight := 10 '[mm] Höhe der schritte
151
152 pub move_trans(a1,a2) 'Bewegen Translatorisch
153
154 '-----Geradliniege Bewegung-----
155 phi := a1 '90°/-270° Bewegung nach 12 Uhr, 0°/360° --> 3 Uhr, 180°/-
180° --> 9 uhr, 270°/-90° --> 6 Uhr
156 repeat_step := a2
157 step_position := 0
158
159 anzahl_steps_f:= math.FMul(step_length, 3.0)'1.5) 'anzahl der abschnitte
eines ganzen schrittes(hin und zurück)
160 anzahl_steps:= math.FRound(anzahl_steps) 'umwandlung in integer
161 step_delta := math.FRound(math.FDiv(anzahl_steps_f,6.0))'3.0)) 'step_delta ist die
Phasenverschiebung
162
163 step_low:= math.FRound(math.FMul(step_length,2.0))'1.0))
164 step_high:= math.FRound(math.FMul(step_length,3.0))'1.5))
165
166 step_x[0] := math.cos(math.FDiv(phi, 57.2958))
167 step_y[0] := math.Sin(math.FDiv(phi, 57.2958))
168

```

```

169 x_max[0] := math.FMul(step_length,step_x[0])
170 y_max[0] := math.FMul(step_length,step_y[0])
171 y_min[0] := math.FNeg(y_max[0])
172 x_min[0] := math.FNeg(x_max[0])
173
174 if repeat_step > 0
175   repeat (repeat_step*step_high)
176     move
177     step_position +=1
178     cognew(setpulse,@stack_1)
179 if repeat_step == 0
180   move
181   step_position +=1
182   cognew(setpulse,@stack_1)
183
184 pub move 'Bewegen translatorisch
185
186 phase := 0
187 repeat 6
188
189   if step_position => step_high
190     step_position:=0
191     aktuel_step_pos:= step_position + (phase * step_delta)
192     if aktuel_step_pos => step_high
193       aktuel_step_pos -= step_high
194     ak_st_pos_f:= math.FFloat(aktuel_step_pos)           'umwandlung von
aktuel_step_pos nach float
195
196
197
198   if aktuel_step_pos < step_low
199     h_delta:= math.FFloat(hight_start)
200     '-----Höhenänderung (Fuss runter)-----
201     if aktuel_step_pos < (step_hight / -2)
202       h_delta := math.FFloat(step_hight)
203       h_delta := math.FAdd(h_delta, (math.FMul(ak_st_pos_f , 2.0))) 'durch
verändern des multiplikators lässt sich der speed der höhenänderung beeinflussen
204     '-----
205
206     '-----X-Y-Änderung-----
207     x_delta := x_min[0]
208     y_delta := y_min[0]
209     x_delta := math.FAdd(x_delta,math.FMul(ak_st_pos_f,step_x[0]))
210     y_delta := math.FAdd(y_delta,math.FMul(ak_st_pos_f,step_y[0]))
211     '-----
212
213   if (aktuel_step_pos => step_low) 'and (aktuel_step_pos =< step_high)
214
215     aktuel_step_pos := aktuel_step_pos - step_low
216     ak_st_pos_f := math.FFloat(aktuel_step_pos)
217     h_delta:= math.FFloat(step_hight)
218     '-----Höhenänderung (Fuss runter)-----
219     if aktuel_step_pos < (step_hight / -2)
220       h_delta := math.FFloat(hight_start)
221       h_delta := math.FSub(h_delta, (math.FMul(ak_st_pos_f , 2.0))) 'durch
verändern des multiplikators lässt sich der speed der höhenänderung beeinflussen
222     '-----
223
224     '-----X-Y-Änderung-----

```

```

225     x_delta := x_max[0]
226     y_delta := y_max[0]
227     x_delta := math.FSub(x_delta,math.FMul(ak_st_pos_f,math.FMul(step_x[0],2.0)))
228     y_delta := math.FSub(y_delta,math.FMul(ak_st_pos_f,math.FMul(step_y[0],2.0)))
229     '-----
230     '---Offset-----
231     if (phase == 0) or (phase == 3)
232         y_delta := math.FSub(y_delta,y_off_A)
233
234     if (phase == 2) or (phase == 5)
235         y_delta := math.FSub(y_delta,y_off_C)
236     '-----
237
238     '-----Seite D-F ----spiegeln über x-achse-----
239     if (phase == 1) or (phase == 3) or (phase == 5)
240         y_delta := math.FNeg(y_delta)
241
242     '-----Seite A-C ----spiegeln über x und y-achse-----
243     if (phase == 0) or (phase == 2) or (phase == 4)
244         x_delta := math.FNeg(x_delta)
245         y_delta := math.FNeg(y_delta)
246
247     calc_winkel
248
249     if phase == 0
250         target_pos := math.FMul(wink_fuss, 10.0)
251         target_pos := math.FRound(target_pos)
252         A[2]:= 2400 - target_pos
253
254         target_pos := math.FMul(wink_schult,10.0)
255         target_pos := math.FRound(target_pos)
256         A[1]:= 700 + target_pos '750 + target_pos
257
258         target_pos := math.FMul(wink_hueft,10.0)
259         target_pos := math.FRound(target_pos)
260         A[0]:= 500 + 900 - target_pos
261     if phase == 4
262         target_pos := math.FMul(wink_fuss, 10.0)
263         target_pos := math.FRound(target_pos)
264         B[2]:= 2400 - target_pos
265
266         target_pos := math.FMul(wink_schult,10.0)
267         target_pos := math.FRound(target_pos)
268         B[1]:= 650 + target_pos '750 + target_pos
269
270         target_pos := math.FMul(wink_hueft,10.0)
271         target_pos := math.FRound(target_pos)
272         B[0]:= 600 + 900 - target_pos
273     if phase == 2
274         target_pos := math.FMul(wink_fuss, 10.0)
275         target_pos := math.FRound(target_pos)
276         C[2]:= 2400 - target_pos
277
278         target_pos := math.FMul(wink_schult,10.0)
279         target_pos := math.FRound(target_pos)
280         C[1]:= 750 + target_pos '750 + target_pos
281
282         target_pos := math.FMul(wink_hueft,10.0)
283         target_pos := math.FRound(target_pos)

```

```

284     C[0]:= 600 + 900 - target_pos
285     if phase == 5
286         target_pos := math.FMul(wink_fuss, 10.0)
287         target_pos := math.FRound(target_pos)
288         D[2]:= 550 + target_pos '2350 - target_pos
289
290         target_pos := math.FMul(wink_schult,10.0)
291         target_pos := math.FRound(target_pos)
292         D[1]:= 600 + 1800 - target_pos '750 + target_pos
293
294         target_pos := math.FMul(wink_hueft,10.0)
295         target_pos := math.FRound(target_pos)
296         D[0]:= 550 + 900 + target_pos
297     if phase == 1
298         target_pos := math.FMul(wink_fuss, 10.0)
299         target_pos := math.FRound(target_pos)
300         E[2]:= 600+ target_pos '2400 - target_pos
301
302         target_pos := math.FMul(wink_schult,10.0)
303         target_pos := math.FRound(target_pos)
304         E[1]:= 650 +1800 - target_pos '750 + target_pos
305
306         target_pos := math.FMul(wink_hueft,10.0)
307         target_pos := math.FRound(target_pos)
308         E[0]:= 600 + 900 + target_pos
309     if phase == 3
310         target_pos := math.FMul(wink_fuss, 10.0)
311         target_pos := math.FRound(target_pos)
312         F[2]:= 600+ target_pos '2400 - target_pos
313
314         target_pos := math.FMul(wink_schult,10.0)
315         target_pos := math.FRound(target_pos)
316         F[1]:= 550 + 1800 - target_pos '750 + target_pos
317
318         target_pos := math.FMul(wink_hueft,10.0)
319         target_pos := math.FRound(target_pos)
320         F[0]:= 550 + 900 + target_pos
321     phase += 1
322
323
324
325 pub calc_winkel 'Berechne Winkel Schulter + Fuss + Hüfte
326
327 '----- RADIUS-----
328     x_neu := math.FAdd(x_0 , x_delta)
329     y_neu := math.FAdd(y_0 , y_delta)
330
331 '----- Winkel von der Hüfte-----
332     wink_hueft := math.ATan(math.FDiv(y_neu,x_neu))
333     wink_hueft := math.FMul(wink_hueft,180.0)
334     wink_hueft := math.FDiv(wink_hueft,pi)
335
336 '-----
337     x_neu := math.FMul(x_neu , x_neu)
338     y_neu := math.FMul(y_neu , y_neu)
339     radius := math.FAdd(x_neu , y_neu)
340     radius := math.FSqr(radius)
341     radius := math.FSub(radius, l_should)
342

```

```

343 ----- LÄNGE VOM BEIN-----
344 h_neu := math.FAdd(h_0 , h_delta)
345 c_2 := math.FMul(h_neu , h_neu) ' c_2 = h^2
346 c_2 := math.FAdd(c_2, math.FMul(radius,radius)) 'ACHTUNG: c_2 zum Quadrat -> c_2 =
h^2 + r^2
347 c_1 := math.FSqr(c_2)
348
349 ----- Winkel vom Fuss-----
350 wink_fuss := math.FDiv(math.FSub(a_2_plus_b_2,c_2 ), axbx2)
351 wink_fuss := math.ACos(wink_fuss)
352 wink_fuss := math.FMul(wink_fuss,180.0)
353 wink_fuss := math.FDiv(wink_fuss,pi) 'Fuss-Winkel in grad
354 wink_fuss := math.FAdd(wink_fuss,17.0)
355
356 ----- Winkel von der Schulter-----
357 wink_schult := math.FDiv(math.FAdd(c_2 , a_2_min_b_2),math.FMul(ax2,c_1))
358 wink_schult := math.ACos(wink_schult)
359 wink_schult := math.FMul(wink_schult,180.0)
360 wink_schult := math.FDiv(wink_schult,pi)
361
362 wink_schult_temp := math.FDiv(radius, h_neu)
363 wink_schult_temp := math.ATan(wink_schult_temp)
364 wink_schult_temp := math.FMul(wink_schult_temp,180.0)
365 wink_schult_temp := math.FDiv(wink_schult_temp,pi)
366 wink_schult := math.FAdd(wink_schult, wink_schult_temp) 'Schenkel-Winkel in grad
367
368
369
370
371 pub move_rotation(center_x,center_y,epsilon) ' x, y Position des Punktes M und Drehradius
Epsilon
372
373 if (center_x == 150) or (center_x == -150) 'x-offsets aller beine
374 center_x += 1
375 if (center_y == 1) or (center_y == 165) or (center_y == -165) ' y-offsets aller beine
376 center_y += 1
377 ' -> phi für jedes Bein
378 ' -> Schrittlänge für jedes Bein
379 l_A := math.FSqr(math.FFloat( (( 150 - center_x)*( 150 - center_x)) + (( 165 - center_y)*
( 165 - center_y)) ))
380 l_B := math.FSqr(math.FFloat( (( 150 - center_x)*( 150 - center_x)) + (( 0 - center_y)*
( 0 - center_y)) ))
381 l_C := math.FSqr(math.FFloat( (( 150 - center_x)*( 150 - center_x)) + ((-165 - center_y)*
(-165 - center_y)) ))
382 l_D := math.FSqr(math.FFloat( ((-150 - center_x)*(-150 - center_x)) + ((-165 - center_y)*
(-165 - center_y)) ))
383 l_E := math.FSqr(math.FFloat( ((-150 - center_x)*(-150 - center_x)) + (( 0 - center_y)*
( 0 - center_y)) ))
384 l_F := math.FSqr(math.FFloat( ((-150 - center_x)*(-150 - center_x)) + (( 165 - center_y)*
( 165 - center_y)) ))
385
386 repeat
387 large_r := l_F
388 if (math.FRound(l_A) => math.FRound(l_B)) and (math.FRound(l_A) => math.FRound(l_C))
and (math.FRound(l_A) => math.FRound(l_D)) and (math.FRound(l_A) => math.FRound(l_E)) and (
math.FRound(l_A) => math.FRound(l_F))
389 large_r := l_A
390 quit
391 if (math.FRound(l_B) => math.FRound(l_C)) and (math.FRound(l_B) => math.FRound(l_D))

```

```

392 and (math.FRound(l_B) => math.FRound(l_E)) and (math.FRound(l_B) => math.FRound(l_F))
393     large_r := l_B
394     quit
395 if (math.FRound(l_C) => math.FRound(l_D)) and (math.FRound(l_C) => math.FRound(l_E))
and (math.FRound(l_C) => math.FRound(l_F))
396     large_r := l_C
397     quit
398 if (math.FRound(l_D) => math.FRound(l_E)) and (math.FRound(l_D) => math.FRound(l_F))
399     large_r := l_D
400     quit
401 if (math.FRound(l_E) => math.FRound(l_F))
402     large_r := l_E
403     quit
404     quit
405 '-----Schrittradius-----
406 ri_A := math.FMul(math.FDiv(l_A,large_r),30.0)
407 ri_B := math.FMul(math.FDiv(l_B,large_r),30.0)
408 ri_C := math.FMul(math.FDiv(l_C,large_r),30.0)
409 ri_D := math.FMul(math.FDiv(l_D,large_r),30.0)
410 ri_E := math.FMul(math.FDiv(l_E,large_r),30.0)
411 ri_F := math.FMul(math.FDiv(l_F,large_r),30.0)
412
413 '-----Schrittwinkel-----
414 phi_A := calc_phi( 150.0, 165.0,center_x,center_y)
415 phi_B := calc_phi( 150.0,  1.0,center_x,center_y)
416 phi_C := calc_phi( 150.0,-165.0,center_x,center_y)
417 phi_D := calc_phi(-150.0,-165.0,center_x,center_y)
418 phi_E := calc_phi(-150.0,  1.0,center_x,center_y)
419 phi_F := calc_phi(-150.0, 165.0,center_x,center_y)
420
421
422 anzahl_steps_f:= 90.0
423 anzahl_steps:= math.FRound(anzahl_steps_f)
424 step_delta := math.FRound(math.FDiv(anzahl_steps_f,6.0))'3.0) 'umwandlung in integer
' step_delta ist die
Phasenverschiebung
425 step_position:= 0
426 step_low:= 60 'Anzahl der
427 step_high:= 90
428
429 '-----Parameter jedes einzelnen Fusses-----
430 step_x[0] := math.FDiv(math.FMul(math.cos(phi_A), ri_A),30.0)
431 step_y[0] := math.FDiv(math.FMul(math.sin(phi_A), ri_A),30.0)
432 x_max[0] := math.FMul(30.0,step_x[0])
433 y_max[0] := math.FMul(30.0,step_y[0])
434 y_min[0] := math.FNeg(y_max[0])
435 x_min[0] := math.FNeg(x_max[0])
436
437 step_x[1] := math.FDiv(math.FMul(math.cos(phi_E), ri_E),30.0)
438 step_y[1] := math.FDiv(math.FMul(math.sin(phi_E), ri_E),30.0)
439 x_max[1] := math.FMul(30.0,step_x[1])
440 y_max[1] := math.FMul(30.0,step_y[1])
441 y_min[1] := math.FNeg(y_max[1])
442 x_min[1] := math.FNeg(x_max[1])
443
444 step_x[2] := math.FDiv(math.FMul(math.cos(phi_C), ri_C),30.0)
445 step_y[2] := math.FDiv(math.FMul(math.sin(phi_C), ri_C),30.0)
446 x_max[2] := math.FMul(30.0,step_x[2])
447 y_max[2] := math.FMul(30.0,step_y[2])

```

```

448 y_min[2] := math.FNeg(y_max[2])
449 x_min[2] := math.FNeg(x_max[2])
450
451 step_x[3] := math.FDiv(math.FMul(math.cos(phi_F), ri_F),30.0)
452 step_y[3] := math.FDiv(math.FMul(math.sin(phi_F), ri_F),30.0)
453 x_max[3] := math.FMul(30.0,step_x[3])
454 y_max[3] := math.FMul(30.0,step_y[3])
455 y_min[3] := math.FNeg(y_max[3])
456 x_min[3] := math.FNeg(x_max[3])
457
458 step_x[4] := math.FDiv(math.FMul(math.cos(phi_B), ri_B),30.0)
459 step_y[4] := math.FDiv(math.FMul(math.sin(phi_B), ri_B),30.0)
460 x_max[4] := math.FMul(30.0,step_x[4])
461 y_max[4] := math.FMul(30.0,step_y[4])
462 y_min[4] := math.FNeg(y_max[4])
463 x_min[4] := math.FNeg(x_max[4])
464
465 step_x[5] := math.FDiv(math.FMul(math.cos(phi_D), ri_D),30.0)
466 step_y[5] := math.FDiv(math.FMul(math.sin(phi_D), ri_D),30.0)
467 x_max[5] := math.FMul(30.0,step_x[5])
468 y_max[5] := math.FMul(30.0,step_y[5])
469 y_min[5] := math.FNeg(y_max[5])
470 x_min[5] := math.FNeg(x_max[5])
471
472 if math.FRound(epsilon) < 0
473   i:= 0
474   repeat 6
475     x_max[i]:= math.FNeg(x_max[i])
476     y_max[i]:= math.FNeg(y_max[i])
477     x_min[i]:= math.FNeg(x_min[i])
478     y_min[i]:= math.FNeg(y_min[i])
479     step_x[i] := math.FNeg(step_x[i])
480     step_y[i] := math.FNeg(step_y[i])
481     i+=1
482     epsilon := math.FNeg(epsilon)
483
484 '-----Bewegungsroutine-----
485
486 winkel_pro_step:= math.FDiv(math.ATan(math.FDiv(30.0,large_r)),30.0)
487 winkel_pro_step:= math.FRound(math.FMul((math.FDiv(epsilon,math.FMul(winkel_pro_step, 57.
29578))),1.5))
488 step_position:=0
489 repeat winkel_pro_step'2700
490   phase := 0
491   repeat 6
492     if step_position => step_high
493       step_position:=0
494     aktuel_step_pos:= step_position +(phase * step_delta)
495     if aktuel_step_pos => step_high
496       aktuel_step_pos -= step_high
497     ak_st_pos_f:= math.FFloat(aktuel_step_pos) 'umwandlung von aktuel_step_pos nach
float
498
499
500
501 if aktuel_step_pos < step_low
502   h_delta:= math.FFloat(hight_start)
503   '-----Höhenänderung (Fuss runter)-----

```

```

504     if aktuel_step_pos < (step_hight / -2)
505         h_delta := math.FFloat(step_hight)
506         h_delta := math.FAdd(h_delta, (math.FMul(ak_st_pos_f , 2.0))) ' durch verändern
des multiplikators lässt sich der speed der höhenänderung beeinflussen
'-----
507
508
509     '-----X-Y-Änderung-----
510     x_delta := x_min[phase]
511     y_delta := y_min[phase]
512     x_delta := math.FAdd(x_delta,math.FMul(ak_st_pos_f,step_x[phase]))
513     y_delta := math.FAdd(y_delta,math.FMul(ak_st_pos_f,step_y[phase]))
514     '-----
515
516     if (aktuel_step_pos => step_low) 'and (aktuel_step_pos =< step_high)
517
518     aktuel_step_pos := aktuel_step_pos - step_low
519     ak_st_pos_f := math.FFloat(aktuel_step_pos)
520     h_delta:= math.FFloat(step_hight)
521     '-----Höhenänderung (Fuss runter)-----
522     if aktuel_step_pos < (step_hight / -2)
523         h_delta := math.FFloat(hight_start)
524         h_delta := math.FSub(h_delta, (math.FMul(ak_st_pos_f , 2.0))) ' durch verändern
des multiplikators lässt sich der speed der höhenänderung beeinflussen
'-----
525
526
527     '-----X-Y-Änderung-----
528     x_delta := x_max[phase]
529     y_delta := y_max[phase]
530     x_delta := math.FSub(x_delta,math.FMul(ak_st_pos_f,math.FMul(step_x[phase],2.0)))
531     y_delta := math.FSub(y_delta,math.FMul(ak_st_pos_f,math.FMul(step_y[phase],2.0)))
532     '-----
533
534     '---Offset-----
535     if (phase == 0) or (phase == 3)
536         y_delta := math.FSub(y_delta,y_off_A)
537
538     if (phase == 2) or (phase == 5)
539         y_delta := math.FSub(y_delta,y_off_C)
540     '-----
541
542     '-----Seite D-F ----spiegeln über x-Achse-----
543     if (phase == 1) or (phase == 3) or (phase == 5)
544         y_delta := math.FNeg(y_delta)
545
546     '-----Seite A-C ----spiegeln über x und y-Achse-----
547     if (phase == 0) or (phase == 2) or (phase == 4)
548         x_delta := math.FNeg(x_delta)
549         y_delta := math.FNeg(y_delta)
550
551     calc_winkel
552
553     if phase == 0
554         target_pos := math.FMul(wink_fuss, 10.0)
555         target_pos := math.FRound(target_pos)
556         A[2]:= 2400 - target_pos
557
558         target_pos := math.FMul(wink_schult,10.0)
559         target_pos := math.FRound(target_pos)
560         A[1]:= 700 + target_pos ' 750 + target_pos

```

```

561     target_pos := math.FMul(wink_hueft,10.0)
562     target_pos := math.FRound(target_pos)
563     target_pos := math.FRound(target_pos)
564     A[0]:= 500 + 900 - target_pos
565   if phase == 4
566     target_pos := math.FMul(wink_fuss, 10.0)
567     target_pos := math.FRound(target_pos)
568     B[2]:= 2400 - target_pos
569
570     target_pos := math.FMul(wink_schult,10.0)
571     target_pos := math.FRound(target_pos)
572     B[1]:= 650 + target_pos '750 + target_pos
573
574     target_pos := math.FMul(wink_hueft,10.0)
575     target_pos := math.FRound(target_pos)
576     B[0]:= 600 + 900 - target_pos
577   if phase == 2
578     target_pos := math.FMul(wink_fuss, 10.0)
579     target_pos := math.FRound(target_pos)
580     C[2]:= 2400 - target_pos
581
582     target_pos := math.FMul(wink_schult,10.0)
583     target_pos := math.FRound(target_pos)
584     C[1]:= 750 + target_pos '750 + target_pos
585
586     target_pos := math.FMul(wink_hueft,10.0)
587     target_pos := math.FRound(target_pos)
588     C[0]:= 600 + 900 - target_pos
589   if phase == 5
590     target_pos := math.FMul(wink_fuss, 10.0)
591     target_pos := math.FRound(target_pos)
592     D[2]:= 550 + target_pos '2350 - target_pos
593
594     target_pos := math.FMul(wink_schult,10.0)
595     target_pos := math.FRound(target_pos)
596     D[1]:= 600 + 1800 - target_pos '750 + target_pos
597
598     target_pos := math.FMul(wink_hueft,10.0)
599     target_pos := math.FRound(target_pos)
600     D[0]:= 550 + 900 + target_pos
601   if phase == 1
602     target_pos := math.FMul(wink_fuss, 10.0)
603     target_pos := math.FRound(target_pos)
604     E[2]:= 600+ target_pos'2400 - target_pos
605
606     target_pos := math.FMul(wink_schult,10.0)
607     target_pos := math.FRound(target_pos)
608     E[1]:= 650 +1800 - target_pos '750 + target_pos
609
610     target_pos := math.FMul(wink_hueft,10.0)
611     target_pos := math.FRound(target_pos)
612     E[0]:= 600 + 900 + target_pos
613   if phase == 3
614     target_pos := math.FMul(wink_fuss, 10.0)
615     target_pos := math.FRound(target_pos)
616     F[2]:= 600+ target_pos'2400 - target_pos
617
618     target_pos := math.FMul(wink_schult,10.0)
619     target_pos := math.FRound(target_pos)

```

```

620     F[1]:= 550 + 1800 - target_pos '750 + target_pos
621
622     target_pos := math.FMul(wink_hueft,10.0)
623     target_pos := math.FRound(target_pos)
624     F[0]:= 550 + 900 + target_pos
625     phase += 1
626
627     step_position +=1
628     cognew(setpulse,@stack_1)
629
630
631 pub calc_phi(x_off,y_off,center_x,center_y)
632
633     if (center_x < math.FRound(x_off)) and (center_y < math.FRound(y_off))
634         x_phi := math.FSub(x_off,math.FFloat(center_x))
635         y_phi := math.FSub(y_off,math.FFloat(center_y))
636         return math.FSub(3.141593 , (math.ATan(math.FDiv(x_phi,y_phi))))
637     if (center_x < math.FRound(x_off)) and (center_y > math.FRound(y_off))
638         x_phi := math.FSub(x_off,math.FFloat(center_x))
639         y_phi := math.FSub(math.FFloat(center_y),y_off)
640         return math.FSub(1.570796 , (math.ATan(math.FDiv(y_phi,x_phi))))
641     if (center_x > math.FRound(x_off)) and (center_y > math.FRound(y_off))
642         x_phi := math.FSub(math.FFloat(center_x),x_off)
643         y_phi := math.FSub(math.FFloat(center_y),y_off)
644         return math.FSub(6.283185 , (math.ATan(math.FDiv(x_phi,y_phi))))
645     if (center_x > math.FRound(x_off)) and (center_y < math.FRound(y_off))
646         x_phi := math.FSub(math.FFloat(center_x),x_off)
647         y_phi := math.FSub(y_off,math.FFloat(center_y))
648         return math.FSub(4.71239 , (math.ATan(math.FDiv(y_phi,x_phi))))
649
650
651 pub setpulse 'Bewegen der Servos, berechnen Impulslänge
652     new_peri_r:= periode
653     new_peri_l:= periode
654
655     i:=0
656     x:=0
657     repeat 3
658         high_pulse_l[i] := ((clkfreq/ 1000000) * F[x])           'aktuelle pulselänge
659         new_peri_l-=high_pulse_l[i]
660         i+=1
661         x+=1
662     x:=0
663     repeat 3
664         high_pulse_l[i] := ((clkfreq/ 1000000) * E[x])           'aktuelle pulselänge
665         new_peri_l-=high_pulse_l[i]
666         i+=1
667         x+=1
668     x:=0
669     repeat 3
670         high_pulse_l[i] := ((clkfreq/ 1000000) * D[x])           'aktuelle pulselänge
671         new_peri_l-=high_pulse_l[i]
672         i+=1
673         x+=1
674     '-----RECHTE SEITE-----
675     i:=0
676     x:=2
677     repeat 3
678         high_pulse_r[i] := ((clkfreq/ 1000000) * C[x])           'aktuelle pulselänge

```

```

679     new_peri_r-=high_pulse_r[i]
680     i+=1
681     x-=1
682     x:=2
683     repeat 3
684         high_pulse_r[i] := ((clkfreq/ 1000000) * B[x])           'aktuelle pulselänge
685         new_peri_r-=high_pulse_r[i]
686         i+=1
687         x-=1
688         x:=2
689         repeat 3
690             high_pulse_r[i] := ((clkfreq/ 1000000) * A[x])       'aktuelle pulselänge
691             new_peri_r-=high_pulse_r[i]
692             i+=1
693             x-=1
694
695 DAT 'Ausgeben der Pulse
696
697     ORG      0 'Begin at Cog RAM addr 0
698 moveServo  mov      Index,          par          'die adresse des
cograms wird nach index geschrieben
699             rdlong   _newperi_r,      Index
700             add      Index,          #4
701             rdlong   p0,              Index
702
703             add      Index,          #4
704             rdlong   _pin_re,        Index
705             add      Index,          #4
706             rdlong   r_high0,        Index          'an index steht
zoneclock, schreibe es nach _z..
707             add      Index,          #4          'Increment Index
to next Pointer
708             rdlong   r_high1,        Index
709             add      Index,          #4          'Increment Index
to next Pointer
710             rdlong   r_high2,        Index
711             add      Index,          #4          'Increment Index
to next Pointer
712             rdlong   r_high3,        Index
713             add      Index,          #4          'Increment Index
to next Pointer
714             rdlong   r_high4,        Index
715             add      Index,          #4          'Increment Index
to next Pointer
716             rdlong   r_high5,        Index
717             add      Index,          #4          'Increment Index
to next Pointer
718             rdlong   r_high6,        Index
719             add      Index,          #4          'Increment Index
to next Pointer
720             rdlong   r_high7,        Index
721             add      Index,          #4          'Increment Index
to next Pointer
722             rdlong   r_high8,        Index
723
724
725             mov     dira, _pin_re 'Set Pin to output
726
727 :loop      mov      Index,          par

```

```

728      rdlong   _newperi_r,      Index
729      add     Index,           #4
730      rdlong   p0,             Index
731      add     Index,           #8
732      rdlong   r_high0,        Index      'an index steht
zoneclock, schreibe es nach _z..
733      add     Index,           #4      'Increment Index
to next Pointer
734      rdlong   r_high1,        Index
735      add     Index,           #4      'Increment Index
to next Pointer
736      rdlong   r_high2,        Index
737      add     Index,           #4      'Increment Index
to next Pointer
738      rdlong   r_high3,        Index
739      add     Index,           #4      'Increment Index
to next Pointer
740      rdlong   r_high4,        Index
741      add     Index,           #4      'Increment Index
to next Pointer
742      rdlong   r_high5,        Index
743      add     Index,           #4      'Increment Index
to next Pointer
744      rdlong   r_high6,        Index
745      add     Index,           #4      'Increment Index
to next Pointer
746      rdlong   r_high7,        Index
747      add     Index,           #4      'Increment Index
to next Pointer
748      rdlong   r_high8,        Index
749
750
751
752      mov     pinShift,        p0
753      mov     outa,            pinShift
754      mov     temp,            r_high0
755      add     temp,            cnt
756      waitcnt temp,            cnt      'Wait
757      mov     outa,            #0      'Toggle Pin
758      shl    pinShift,        #1
759
760      mov     outa,            pinShift
761      mov     temp,            r_high1
762      add     temp,            cnt
763      waitcnt temp,            cnt      'Wait
764      mov     outa,            #0      'Toggle Pin
765      shl    pinShift,        #1
766
767      mov     outa,            pinShift
768      mov     temp,            r_high2
769      add     temp,            cnt
770      waitcnt temp,            cnt      'Wait
771      mov     outa,            #0      'Toggle Pin
772      shl    pinShift,        #1
773
774      mov     outa,            pinShift
775      mov     temp,            r_high3
776      add     temp,            cnt
777      waitcnt temp,            cnt      'Wait
---
```

```

778      mov      outa,          #0          'Toggle Pin
779      shl      pinShift,     #1
780
781      mov      outa,          pinShift
782      mov      temp,         r_high4
783      add      temp,         cnt
784      waitcnt  temp,         cnt          'Wait
785      mov      outa,          #0          'Toggle Pin
786      shl      pinShift,     #1
787
788      mov      outa,          pinShift
789      mov      temp,         r_high5
790      add      temp,         cnt
791      waitcnt  temp,         cnt          'Wait
792      mov      outa,          #0          'Toggle Pin
793      shl      pinShift,     #1
794
795      mov      outa,          pinShift
796      mov      temp,         r_high6
797      add      temp,         cnt
798      waitcnt  temp,         cnt          'Wait
799      mov      outa,          #0          'Toggle Pin
800      shl      pinShift,     #1
801
802      mov      outa,          pinShift
803      mov      temp,         r_high7
804      add      temp,         cnt
805      waitcnt  temp,         cnt          'Wait
806      mov      outa,          #0          'Toggle Pin
807      shl      pinShift,     #1
808
809      mov      outa,          pinShift
810      mov      temp,         r_high8
811      add      temp,         cnt
812      waitcnt  temp,         cnt          'Wait
813      mov      outa,          #0          'Toggle
Pin
814
815      mov      peri_temp,     _newperi_r
816      add      peri_temp,     cnt
817      waitcnt  peri_temp,     cnt          'Wait
818      jmp      #:loop        'Loop
endlessly
819
820
821
822
823 Index      res      1
824
825
826 r_high0     res      1
827 r_high1     res      1
828 r_high2     res      1
829 r_high3     res      1
830 r_high4     res      1
831 r_high5     res      1
832 r_high6     res      1
833 r_high7     res      1
834 r_high8     res      1

```

835

836

837

838

839

840

841

842

843

```
_newperi_r    res    1
peri_temp     res    1
temp          res    1
_pin_re       res    1

p0            res    1
pinShift      res    1
```