```
1: ; ******************************************************
2: ; * Projekt06.asm - Playstation Controller an PIC16F84A *
3: ; ******************************************************
4: ;
5: ;        KOMMT NOCH....
6: ;
7: ;
8: ; ************************************************************
9: ; *                        Changelog 0_1                     *
10: ; ************************************************************
11: ;
12: ; UP_wait250khz hinzugefügt
13: ; UP_wait05s hinzugefügt
14: ;
15: ; main neu aufgebaut um Programmablauf per LED's anzeigen zu lassen
16: ;
17: ;
18: ;*****************************************************************************
19: ;* Bestimmung des Prozessortyps für den Assembler und das Programmiergerät *
20: ;*****************************************************************************
21:
22:                     LIST p=16F84A
23:
24:
25: ;*************************************************************
26: ;* Includedatei für den 16F84A einbinden (vordef. Reg. und Konst.) *
27: ;*************************************************************
28:
29:                     #include <p16f84A.INC>
30:
31: ; Diese Datei enthält Vordefinitionen für wichtige Register und Konstanten.
32: ; (Z.B. gibt es die Konstante PORTB mit der sich ohne Angabe der
33: ; absoluten Adresse H'0006' der Port B des Prozessors ansprechen lässt)
34:
35:
36: ;*********************************************************
37: ;* Konfigurationseinstellungen für IC-Prog vordefinieren *
38: ;*********************************************************
39:
40:                     __CONFIG _PWRTE_ON & _CP_OFF & _HS_OSC & _WDT_OFF
41:
42: ; Hier werden verschiedene Prozessoreigenschaften festgelegt:
43: ; _PWRTE_ON schaltet den Power Up Timer ein, d.h. der Prozessor wartet nach
44: ;           dem Einschalten ca. 70ms mit dem Programmstart, um sicher zu sein,
45: ;           dass alle angeschlossene Peripherie bereit ist.
46: ; _CP_OFF schaltet die Code-Protection des Prozesors aus. Damit ist das im
    Prozessor
47: ;         befindliche Programm jederzeit auslesbar und überschreibbar.
48: ; _HS_OSC spezifiziert einen Quarzoszillator (Highspeed) als Zeitbasis für den
    Prozessor.
49: ; _WDT_OFF schaltet den Watchdog-Timer des Prozesors aus.
50:
51:
52: ;*********************************
53: ;* Register / Variablen  festlegen *
54: ;*********************************
55: ; hier werden Adressen von Registern / Variablen festgelegt. Diese werden
    beginnend
56: ; mit der Adresse H'20' aufsteigend vergeben.
57:
58:
59:          CBLOCK   H'20'
60:                   zaehler_in        ;Zaehler innere Schleife
61:                   zaehler_mid       ;Zaehler mittlere Schleife
```

```
62:                     zaehler_out       ;Zaehler aeussere Schleife
63:
64:                     cnt_get_btn_stats ;Get_Btn_Stats Schleife
65:
66:                     wait250khz        ;250kHz zaehler
67:
68:                     wait05s
69:                     wait05s_1
70:                     wait05s_2
71:
72:
73:           ENDC
74:
75:           in_cnt            EQU D'0'          ;Startwert inner Schleife
76:           mid_cnt           EQU D'0'          ;Startwert mittlere ""
77:           out_cnt           EQU D'1'          ;Startwert äussere ""
78:
79:           get_btn_stats_cnt EQU D'7'          ;Startwert Get_Btn_Stats Schleife
80:
81:
82: ;************************
83: ;* Konstanten festlegen *
84: ;************************
85:
86:
87:
88: ; *********************************************************************
89: ; * Definition von einzelnen Bits in einem Register / in einer Variable *
90: ; *********************************************************************
91:
92: ;#####################################################################
93: #DEFINE   e_data            PORTA, 0          ; Data            Eingang
94: #DEFINE   a_command         PORTA, 1          ; Command         Ausgang
95: #DEFINE   a_clock           PORTA, 2          ; Takt            Ausgang
96: #DEFINE   a_att             PORTA, 3          ; ATT             Ausgang
97: ;#####################################################################
98: #DEFINE   a_test            PORTB, 0          ; Tets LED        Ausgang
99: ;#####################################################################
100: #DEFINE   a1                PORTB, 0          ; BCD-Decoder     Ausgang
101: #DEFINE   b1                PORTB, 1          ; BCD-Decoder     Ausgang
102: #DEFINE   c1                PORTB, 2          ; BCD-Decoder     Ausgang
103: #DEFINE   d1                PORTB, 3          ; BCD-Decoder     Ausgang
104: ;#####################################################################
105: #DEFINE   a2                PORTB, 4          ; BCD-Decoder     Ausgang
106: #DEFINE   b2                PORTB, 5          ; BCD-Decoder     Ausgang
107: #DEFINE   c2                PORTB, 6          ; BCD-Decoder     Ausgang
108: #DEFINE   d2                PORTB, 7          ; BCD-Decoder     Ausgang
109: ;#####################################################################
110: #DEFINE   bank1             STATUS, RP0
111: ;#####################################################################
112:
113:
114:
115: ; In diesem Beispiel steht das Wort ir_sensor für Bit 0 von Port A und das Wort
     motor
116: ; für das Bit 0 von Port B.
117:
118:
119: ;*****************
120: ;* Programmstart *
121: ;*****************
122:
123:           ORG      H'00'            ; Das Programm wird ab Speicherstelle 0 in
     den Speicher geschrieben
```

```
124:          GOTO     init              ; Springe zur Grundinitialisierung der Ports
    A und B
125:
126:
127: ;*******************
128: ;* Initialisierung *
129: ;*******************
130:
131: init      BSF      bank1             ; wechsle zu Registerbank 1 (spezielle
    Register)
132:
133:          MOVLW    B'00000001'
134:          MOVWF    TRISA             ; RA0 Eingang (RA1 bis RA3 sind Ausgänge)
135:          MOVLW    B'00000000'
136:          MOVWF    TRISB             ; RB0 bis RB7 sind Ausgänge
137:
138:          BCF      bank1             ; wechsle zu Registerbank 0 (normaler
    Speicherbereich)
139:
140:          CLRF     PORTA             ; Port A löschen
141:          CLRF     PORTB             ; Port B löschen
142:
143: ; Die Register TRISA und TRISB legen fest, welche Bits in den jeweiligen Ports
    Ein- bzw.
144: ; Ausgänge sind. Eine '1' an der entsprechenden Stelle setzt das Bit des Ports
    als Ein-
145: ; gang eine '0' setzt das Bit als Ausgang.
146:
147:
148:
149: ;******************
150: ;* Hauptprogramm *
151: ;******************
152:
153: ;#########################################################################
154: ;#DEFINE e_data            PORTA, 0          ; Data            Eingang #
155: ;#DEFINE a_command         PORTA, 1          ; Command         Ausgang #
156: ;#DEFINE a_clock           PORTA, 2          ; Takt            Ausgang #
157: ;#DEFINE a_att             PORTA, 3          ; ATT             Ausgang #
158: ;#########################################################################
159: ;#DEFINE a_test            PORTB, 0          ; Tets LED        Ausgang #
160: ;#########################################################################
161: ;#DEFINE a1                PORTB, 0          ; BCD-Decoder     Ausgang #
162: ;#DEFINE b1                PORTB, 1          ; BCD-Decoder     Ausgang #
163: ;#DEFINE c1                PORTB, 2          ; BCD-Decoder     Ausgang #
164: ;#DEFINE d1                PORTB, 3          ; BCD-Decoder     Ausgang #
165: ;#########################################################################
166: ;#DEFINE a2                PORTB, 4          ; BCD-Decoder     Ausgang #
167: ;#DEFINE b2                PORTB, 5          ; BCD-Decoder     Ausgang #
168: ;#DEFINE c2                PORTB, 6          ; BCD-Decoder     Ausgang #
169: ;#DEFINE d2                PORTB, 7          ; BCD-Decoder     Ausgang #
170: ;#########################################################################
171:
172:
173: main
174:          CALL     UP_wait05s
175:          BSF      a_att                      ;Controller ignoriert alle Daten
176:
177:          BSF      a1;;;;;;;;;
178:          CALL     UP_wait05s
179:          CALL     UP_Start                   ;H'01'
180:          BCF      a1;;;;;;;;;
181:
182:          BSF      b1;;;;;;;;;
```

```
183:          CALL     UP_wait05s
184:          CALL     UP_GetType                ;H'42'
185:          BCF      b1;;;;;;;;;
186:
187:          BSF      c1;;;;;;;;;
188:          CALL     UP_wait05s
189:          CALL     UP_Status                 ;H'00'
190:          BCF      c1;;;;;;;;;
191:
192:          BSF      d1;;;;;;;;;
193:          CALL     UP_wait05s
194:          CALL     UP_linke_btns             ;H'00'
195:          BCF      d1;;;;;;;;;
196:
197:          BSF      a2;;;;;;;;;
198:          CALL     UP_wait05s
199:          CALL     UP_rechte_btns            ;H'00'
200:          BCF      a2;;;;;;;;;
201:
202:          BSF      b2;;;;;;;;;
203:          CALL     UP_wait05s
204:          CALL     UP_sende0                 ;#########################
205:          CALL     UP_sende0                 ;Hier kommen die Joysticks
206:          CALL     UP_sende0                 ;die abgefragt werden aber
207:          CALL     UP_sende0                 ;nicht gespeichert
208:          BCF      b2;;;;;;;;;
209:
210:
211:
212:
213:          GOTO     main
214:
215:
216: ;******************
217: ;* Unterprogramme *
218: ;******************
219:
220: ;PSX_TxRx:
221: ;  FOR idx = 0 TO 7
222: ;    PsxCmd = psxOut.LOWBIT(idx)                setup command bit
223: ;    PsxClk = ClockMode                         clock the bit (low)
224: ;    psxIn.LOWBIT(idx) = PsxDat                 get data bit
225: ;    PsxClk = ~ClockMode                        r(high)
226: ;  NEXT
227:
228: ;#############################################################################
229: UP_Start
230:          ;H'01' (Bitfolge 00000001) muss im LSB-Verfahren über
231:          ;a_command gesendet werden.
232:
233:          BCF      a_att                     ; ATT auf LOW das der Controller
234:                                             ; die Daten annimmt
235:
236:          ;Bit 1 Senden
237:          BSF      a_command         ; "1" senden
238:          BCF      a_clock           ; Clock LOW
239:          ;CALL UP_wait250khz
240:          ;BTFSC   e_data            ; e_data in Register speichern
241:          BSF      a_clock           ; Clock HIGH
242:
243:
244:          ;Bit 0 Senden
245:          BCF      a_command         ; "0" senden
246:          BCF      a_clock           ; Clock LOW
```

```
247:          ;CALL UP_wait250khz
248:          ;BTFSC   e_data           ; e_data in Register speichern
249:          BSF      a_clock          ; Clock HIGH
250:
251:
252:          ;Bit 0 Senden
253:          BCF      a_command        ; "0" senden
254:          BCF      a_clock          ; Clock LOW
255:          ;CALL UP_wait250khz
256:          ;BTFSC   e_data           ; e_data in Register speichern
257:          BSF      a_clock          ; Clock HIGH
258:
259:
260:          ;Bit 0 Senden
261:          BCF      a_command        ; "0" senden
262:          BCF      a_clock          ; Clock LOW
263:          ;CALL UP_wait250khz
264:          ;BTFSC   e_data           ; e_data in Register speichern
265:          BSF      a_clock          ; Clock HIGH
266:
267:
268:          ;Bit 0 Senden
269:          BCF      a_command        ; "0" senden
270:          BCF      a_clock          ; Clock LOW
271:          ;CALL UP_wait250khz
272:          ;BTFSC   e_data           ; e_data in Register speichern
273:          BSF      a_clock          ; Clock HIGH
274:
275:
276:          ;Bit 0 Senden
277:          BCF      a_command        ; "0" senden
278:          BCF      a_clock          ; Clock LOW
279:          ;CALL UP_wait250khz
280:          ;BTFSC   e_data           ; e_data in Register speichern
281:          BSF      a_clock          ; Clock HIGH
282:
283:
284:          ;Bit 0 Senden
285:          BCF      a_command        ; "0" senden
286:          BCF      a_clock          ; Clock LOW
287:          ;CALL UP_wait250khz
288:          ;BTFSC   e_data           ; e_data in Register speichern
289:          BSF      a_clock          ; Clock HIGH
290:
291:
292:          ;Bit 0 Senden
293:          BCF      a_command        ; "0" senden
294:          BCF      a_clock          ; Clock LOW
295:          ;CALL UP_wait250khz
296:          ;BTFSC   e_data           ; e_data in Register speichern
297:          BSF      a_clock          ; Clock HIGH
298:
299:
300: RETURN
301: ;*****************************************************************************
302: UP_GetType
303:          ;H'42' (Bitfolge 01000010) muss im LSB-Verfahren über
304:          ;a_command gesendet werden.
305:
306:          ;Bit 0 Senden
307:          BCF      a_command        ; "1" senden
308:          BCF      a_clock          ; Clock LOW
309:          ;CALL UP_wait250khz
310:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
```

```
311:          BSF      a_clock          ; Clock HIGH
312:
313:
314:          ;Bit 1 Senden
315:          BSF      a_command        ; "0" senden
316:          BCF      a_clock          ; Clock LOW
317:          ;CALL UP_wait250khz
318:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
319:          BSF      a_clock          ; Clock HIGH
320:
321:
322:          ;Bit 0 Senden
323:          BCF      a_command        ; "0" senden
324:          BCF      a_clock          ; Clock LOW
325:          ;CALL UP_wait250khz
326:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
327:          BSF      a_clock          ; Clock HIGH
328:
329:
330:          ;Bit 0 Senden
331:          BCF      a_command        ; "0" senden
332:          BCF      a_clock          ; Clock LOW
333:          ;CALL UP_wait250khz
334:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
335:          BSF      a_clock          ; Clock HIGH
336:
337:
338:          ;Bit 0 Senden
339:          BCF      a_command        ; "0" senden
340:          BCF      a_clock          ; Clock LOW
341:          ;CALL UP_wait250khz
342:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
343:          BSF      a_clock          ; Clock HIGH
344:
345:
346:          ;Bit 0 Senden
347:          BCF      a_command        ; "0" senden
348:          BCF      a_clock          ; Clock LOW
349:          ;CALL UP_wait250khz
350:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
351:          BSF      a_clock          ; Clock HIGH
352:
353:
354:          ;Bit 1 Senden
355:          BSF      a_command        ; "0" senden
356:          BCF      a_clock          ; Clock LOW
357:          ;CALL UP_wait250khz
358:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
359:          BSF      a_clock          ; Clock HIGH
360:
361:
362:          ;Bit 0 Senden
363:          BCF      a_command        ; "0" senden
364:          BCF      a_clock          ; Clock LOW
365:          ;CALL UP_wait250khz
366:          ;BTFSC   e_data           ; e_data in Register speichern - controller
     type
367:          BSF      a_clock          ; Clock HIGH
```

```
368:
369:
370: RETURN
371: ;#############################################################################
372: UP_Status
373:           ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
374:           ;a_command gesendet werden.
375:
376:           ;Bit 0 Senden
377:           BCF       a_command          ; "1" senden
378:           BCF       a_clock            ; Clock LOW
379:           ;CALL UP_wait250khz
380:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
381:           BSF       a_clock            ; Clock HIGH
382:
383:           ;Bit 0 Senden
384:           BCF       a_command          ; "0" senden
385:           BCF       a_clock            ; Clock LOW
386:           ;CALL UP_wait250khz
387:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
388:           BSF       a_clock            ; Clock HIGH
389:
390:           ;Bit 0 Senden
391:           BCF       a_command          ; "0" senden
392:           BCF       a_clock            ; Clock LOW
393:           ;CALL UP_wait250khz
394:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
395:           BSF       a_clock            ; Clock HIGH
396:
397:           ;Bit 0 Senden
398:           BCF       a_command          ; "0" senden
399:           BCF       a_clock            ; Clock LOW
400:           ;CALL UP_wait250khz
401:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
402:           BSF       a_clock            ; Clock HIGH
403:
404:           ;Bit 0 Senden
405:           BCF       a_command          ; "0" senden
406:           BCF       a_clock            ; Clock LOW
407:           ;CALL UP_wait250khz
408:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
409:           BSF       a_clock            ; Clock HIGH
410:
411:           ;Bit 0 Senden
412:           BCF       a_command          ; "0" senden
413:           BCF       a_clock            ; Clock LOW
414:           ;CALL UP_wait250khz
415:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
416:           BSF       a_clock            ; Clock HIGH
417:
418:           ;Bit 0 Senden
419:           BCF       a_command          ; "0" senden
420:           BCF       a_clock            ; Clock LOW
421:           ;CALL UP_wait250khz
422:           ;BTFSC    e_data             ; e_data in Register speichern - controller
     type
423:           BSF       a_clock            ; Clock HIGH
424:
```

```
425:          ;Bit 0 Senden
426:          BCF       a_command          ; "0" senden
427:          BCF       a_clock            ; Clock LOW
428:          ;CALL UP_wait250khz
429:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
430:          BSF       a_clock            ; Clock HIGH
431:
432: RETURN
433: ;#########################################################################
434: UP_linke_btns
435:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
436:          ;a_command gesendet werden.
437:
438:          ; Pfeiltaste nach links <-
439:          ;######################
440:          ;Bit 0 Senden
441:          BCF       a_command          ; "0" senden
442:          BCF       a_clock            ; Clock LOW
443:          ;CALL UP_wait250khz
444:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
445:          BSF       c1
446:          BSF       a_clock            ; Clock HIGH
447:
448:          ; Pfeiltaste nach unten \/
449:          ;######################
450:          ;Bit 0 Senden
451:          BCF       a_command          ; "0" senden
452:          BCF       a_clock            ; Clock LOW
453:          ;CALL UP_wait250khz
454:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
455:          BSF       a1
456:          BSF       a_clock            ; Clock HIGH
457:
458:          ; Pfeiltaste nach rechts ->
459:          ;######################
460:          ;Bit 0 Senden
461:          BCF       a_command          ; "0" senden
462:          BCF       a_clock            ; Clock LOW
463:          ;CALL UP_wait250khz
464:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
465:          BSF       b1
466:          BSF       a_clock            ; Clock HIGH
467:
468:          ; Pfeiltaste nach oben /\
469:          ;######################
470:          ;Bit 0 Senden
471:          BCF       a_command          ; "0" senden
472:          BCF       a_clock            ; Clock LOW
473:          ;CALL UP_wait250khz
474:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
475:          BSF       d1
476:          BSF       a_clock            ; Clock HIGH
477:
478:          ; Start
479:          ;######################
480:          ;Bit 0 Senden
481:          BCF       a_command          ; "0" senden
482:          BCF       a_clock            ; Clock LOW
483:          ;CALL UP_wait250khz
```

```
484:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
485:          BSF      a_clock          ; Clock HIGH
486:
487:          ; Joy-R
488:          ;#####################
489:          ;Bit 0 Senden
490:          BCF      a_command        ; "0" senden
491:          BCF      a_clock          ; Clock LOW
492:          ;CALL UP_wait250khz
493:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
494:          BSF      a_clock          ; Clock HIGH
495:
496:          ; Joy-L
497:          ;#####################
498:          ;Bit 0 Senden
499:          BCF      a_command        ; "0" senden
500:          BCF      a_clock          ; Clock LOW
501:          ;CALL UP_wait250khz
502:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
503:          BSF      a_clock          ; Clock HIGH
504:
505:          ; Select
506:          ;#####################
507:          ;Bit 0 Senden
508:          BCF      a_command        ; "0" senden
509:          BCF      a_clock          ; Clock LOW
510:          ;CALL UP_wait250khz
511:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
512:          BSF      a_clock          ; Clock HIGH
513:
514: RETURN
515: ;#########################################################################
516: UP_rechte_btns
517:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
518:          ;a_command gesendet werden.
519:
520:          ; [] - Quadrat Taste
521:          ;#####################
522:          ;Bit 0 Senden
523:          BCF      a_command        ; "0" senden
524:          BCF      a_clock          ; Clock LOW
525:          ;CALL UP_wait250khz
526:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
527:          BSF      a_clock          ; Clock HIGH
528:
529:          ; X - X Taste
530:          ;#####################
531:          ;Bit 0 Senden
532:          BCF      a_command        ; "0" senden
533:          BCF      a_clock          ; Clock LOW
534:          ;CALL UP_wait250khz
535:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
536:          BSF      a_clock          ; Clock HIGH
537:
538:          ; O - Kreis Taste
539:          ;#####################
540:          ;Bit 0 Senden
541:          BCF      a_command        ; "0" senden
```

```
542:          BCF      a_clock          ; Clock LOW
543:          ;CALL UP_wait250khz
544:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
545:          BSF      a_clock          ; Clock HIGH
546:
547:          ; /.\ - Dreick Taste
548:          ;#####################
549:          ;Bit 0 Senden
550:          BCF      a_command        ; "0" senden
551:          BCF      a_clock          ; Clock LOW
552:          ;CALL UP_wait250khz
553:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
554:          BSF      a_clock          ; Clock HIGH
555:
556:          ; R1 - Schultertaste
557:          ;#####################
558:          ;Bit 0 Senden
559:          BCF      a_command        ; "0" senden
560:          BCF      a_clock          ; Clock LOW
561:          ;CALL UP_wait250khz
562:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
563:          BSF      a_clock          ; Clock HIGH
564:
565:          ; L1 - Schultertaste
566:          ;#####################
567:          ;Bit 0 Senden
568:          BCF      a_command        ; "0" senden
569:          BCF      a_clock          ; Clock LOW
570:          ;CALL UP_wait250khz
571:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
572:          BSF      a_clock          ; Clock HIGH
573:
574:          ; R2 - Schultertaste
575:          ;#####################
576:          ;Bit 0 Senden
577:          BCF      a_command        ; "0" senden
578:          BCF      a_clock          ; Clock LOW
579:          ;CALL UP_wait250khz
580:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
581:          BSF      a_clock          ; Clock HIGH
582:
583:          ; L2 - Schultertaste
584:          ;#####################
585:          ;Bit 0 Senden
586:          BCF      a_command        ; "0" senden
587:          BCF      a_clock          ; Clock LOW
588:          ;CALL UP_wait250khz
589:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
590:          BSF      a_clock          ; Clock HIGH
591:
592: RETURN
593: ;*************************************************************************
594: UP_sende0
595:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
596:          ;a_command gesendet werden.
597:
598:          ;Bit 0 Senden
599:          BCF      a_command        ; "0" senden
```

```
600:          BCF      a_clock          ; Clock LOW
601:          ;CALL UP_wait250khz
602:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
603:          BSF      a_clock          ; Clock HIGH
604:
605:          ;Bit 0 Senden
606:          BCF      a_command        ; "0" senden
607:          BCF      a_clock          ; Clock LOW
608:          ;CALL UP_wait250khz
609:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
610:          BSF      a_clock          ; Clock HIGH
611:
612:          ;Bit 0 Senden
613:          BCF      a_command        ; "0" senden
614:          BCF      a_clock          ; Clock LOW
615:          ;CALL UP_wait250khz
616:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
617:          BSF      a_clock          ; Clock HIGH
618:
619:          ;Bit 0 Senden
620:          BCF      a_command        ; "0" senden
621:          BCF      a_clock          ; Clock LOW
622:          ;CALL UP_wait250khz
623:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
624:          BSF      a_clock          ; Clock HIGH
625:
626:          ;Bit 0 Senden
627:          BCF      a_command        ; "0" senden
628:          BCF      a_clock          ; Clock LOW
629:          ;CALL UP_wait250khz
630:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
631:          BSF      a_clock          ; Clock HIGH
632:
633:          ;Bit 0 Senden
634:          BCF      a_command        ; "0" senden
635:          BCF      a_clock          ; Clock LOW
636:          ;CALL UP_wait250khz
637:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
638:          BSF      a_clock          ; Clock HIGH
639:
640:          ;Bit 0 Senden
641:          BCF      a_command        ; "0" senden
642:          BCF      a_clock          ; Clock LOW
643:          ;CALL UP_wait250khz
644:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
645:          BSF      a_clock          ; Clock HIGH
646:
647:          ;Bit 0 Senden
648:          BCF      a_command        ; "0" senden
649:          BCF      a_clock          ; Clock LOW
650:          ;CALL UP_wait250khz
651:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
652:          BSF      a_clock          ; Clock HIGH
653:
654: RETURN
655: ;#############################################################################
```

```
656: UP_sende0_schleife
657:           MOVLW     get_btn_stats_cnt          ;Startwert innere Schleife
658:           MOVWF     cnt_get_btn_stats
659:
660: loop_btn_stats
661:                     BCF       a_command          ; "0" senden
662:                     BCF       a_clock            ; Clock LOW
663:                     ;CALL UP_wait250khz
664:                     ;BTFSC    e_data             ; e_data in Register speichern -
     controller type
665:                     BSF       a_clock            ; Clock HIGH
666:
667:                     DECFSZ    cnt_get_btn_stats, F       ; innnere Schleife
668:                     GOTO      loop_btn_stats
669:
670: RETURN
671: ;*******************************************************************************
672: UP_wait
673:           MOVLW     out_cnt   ;Startwert äussere Schleife
674:           MOVWF     zaehler_out
675:
676: loop_out
677:           MOVLW     mid_cnt   ;Startwert mittlere Schleife
678:           MOVWF     zaehler_mid
679:
680: loop_mid
681:           MOVLW     in_cnt    ;Startwert innere Schleife
682:           MOVWF     zaehler_in
683:
684: loop_in
685:           DECFSZ    zaehler_in, F; innnere Schleife
686:           GOTO      loop_in
687:
688:           DECFSZ    zaehler_mid, F; mittlere Schleife
689:           GOTO      loop_mid
690:
691:           DECFSZ    zaehler_out, F; äussere Schleife
692:           GOTO      loop_out
693:
694: RETURN
695: ;*******************************************************************************
696: UP_wait250khz
697:                               ;19 cycles
698:           movlw     0x06
699:           movwf     wait250khz
700: Delay_0
701:           decfsz    wait250khz, f
702:           goto      Delay_0
703:
704:                               ;1 cycle
705:           nop
706:
707: RETURN
708: ;*******************************************************************************
709: UP_wait05s
710:                               ;2499999 cycles
711:           movlw     0x16
712:           movwf     wait05s
713:           movlw     0x74
714:           movwf     wait05s_1
715:           movlw     0x06
716:           movwf     wait05s_2
717: Delay_05
718:           decfsz    wait05s, f
```

```
719:            goto      $+2
720:            decfsz    wait05s_1, f
721:            goto      $+2
722:            decfsz    wait05s_2, f
723:            goto      Delay_05
724:
725:                              ;1 cycle
726:            nop
727: RETURN
728: ;#########################################################################################
729: END
730:
```