```
1: ; ******************************************************
2: ; * Projekt06.asm - Playstation Controller an PIC16F84A *
3: ; ******************************************************
4: ;
5: ;        KOMMT NOCH....
6: ;
7: ;
8: ; ************************************************************
9: ; *                        Changelog 0_2                     *
10: ; ************************************************************
11: ;
12: ; main neu aufgebaut (Kommentiert)
13: ;
14: ; Clockrate neu aufgebaut um 250 khz einzuhalten
15: ;
16: ;
17: ;*****************************************************************************
18: ;* Bestimmung des Prozessortyps für den Assembler und das Programmiergerät *
19: ;*****************************************************************************
20:
21:                     LIST p=16F84A
22:
23:
24: ;*****************************************************************
25: ;* Includedatei für den 16F84A einbinden (vordef. Reg. und Konst.) *
26: ;*****************************************************************
27:
28:                     #include <p16f84A.INC>
29:
30: ; Diese Datei enthält Vordefinitionen für wichtige Register und Konstanten.
31: ; (Z.B. gibt es die Konstante PORTB mit der sich ohne Angabe der
32: ; absoluten Adresse H'0006' der Port B des Prozessors ansprechen lässt)
33:
34:
35: ;*********************************************************
36: ;* Konfigurationseinstellungen für IC-Prog vordefinieren *
37: ;*********************************************************
38:
39:                     __CONFIG _PWRTE_ON & _CP_OFF & _HS_OSC & _WDT_OFF
40:
41: ; Hier werden verschiedene Prozessoreigenschaften festgelegt:
42: ; _PWRTE_ON schaltet den Power Up Timer ein, d.h. der Prozessor wartet nach
43: ;           dem Einschalten ca. 70ms mit dem Programmstart, um sicher zu sein,
44: ;           dass alle angeschlossene Peripherie bereit ist.
45: ; _CP_OFF schaltet die Code-Protection des Prozesors aus. Damit ist das im
    Prozessor
46: ;        befindliche Programm jederzeit auslesbar und überschreibbar.
47: ; _HS_OSC spezifiziert einen Quarzoszillator (Highspeed) als Zeitbasis für den
    Prozessor.
48: ; _WDT_OFF schaltet den Watchdog-Timer des Prozesors aus.
49:
50:
51: ;***********************************
52: ;* Register / Variablen  festlegen *
53: ;***********************************
54: ; hier werden Adressen von Registern / Variablen festgelegt. Diese werden
    beginnend
55: ; mit der Adresse H'20' aufsteigend vergeben.
56:
57:
58:          CBLOCK   H'20'
59:                   zaehler_in        ;Zaehler innere Schleife
60:                   zaehler_mid       ;Zaehler mittlere Schleife
61:                   zaehler_out       ;Zaehler aeussere Schleife
```

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

```
124:
125:
126: ;*******************
127: ;* Initialisierung *
128: ;*******************
129:
130: init     BSF      bank1              ; wechsle zu Registerbank 1 (spezielle
     Register)
131:
132:          MOVLW    B'00000001'
133:          MOVWF    TRISA              ; RA0 Eingang (RA1 bis RA3 sind Ausgänge)
134:          MOVLW    B'00000000'
135:          MOVWF    TRISB              ; RB0 bis RB7 sind Ausgänge
136:
137:          BCF      bank1              ; wechsle zu Registerbank 0 (normaler
     Speicherbereich)
138:
139:          CLRF     PORTA              ; Port A löschen
140:          CLRF     PORTB              ; Port B löschen
141:
142: ; Die Register TRISA und TRISB legen fest, welche Bits in den jeweiligen Ports
     Ein- bzw.
143: ; Ausgänge sind. Eine '1' an der entsprechenden Stelle setzt das Bit des Ports
     als Ein-
144: ; gang eine '0' setzt das Bit als Ausgang.
145:
146:
147:
148: ;******************
149: ;* Hauptprogramm *
150: ;******************
151:
152: ;#########################################################################
153: ;#DEFINE e_data            PORTA, 0          ; Data            Eingang #
154: ;#DEFINE a_command         PORTA, 1          ; Command         Ausgang #
155: ;#DEFINE a_clock           PORTA, 2          ; Takt            Ausgang #
156: ;#DEFINE a_att             PORTA, 3          ; ATT             Ausgang #
157: ;#########################################################################
158: ;#DEFINE a_test            PORTB, 0          ; Tets LED        Ausgang #
159: ;#########################################################################
160: ;#DEFINE a1                PORTB, 0          ; BCD-Decoder     Ausgang #
161: ;#DEFINE b1                PORTB, 1          ; BCD-Decoder     Ausgang #
162: ;#DEFINE c1                PORTB, 2          ; BCD-Decoder     Ausgang #
163: ;#DEFINE d1                PORTB, 3          ; BCD-Decoder     Ausgang #
164: ;#########################################################################
165: ;#DEFINE a2                PORTB, 4          ; BCD-Decoder     Ausgang #
166: ;#DEFINE b2                PORTB, 5          ; BCD-Decoder     Ausgang #
167: ;#DEFINE c2                PORTB, 6          ; BCD-Decoder     Ausgang #
168: ;#DEFINE d2                PORTB, 7          ; BCD-Decoder     Ausgang #
169: ;#########################################################################
170:
171:
172: main
173:          CALL     UP_wait05s
174:          BCF      a_att              ; ATT auf LOW das der Controller
175:                                      ; die Daten annimmt
176:
177:          BSF      a1;;;;;;;;;
178:          CALL     UP_wait05s
179:          CALL     UP_Start           ;Empfangen:  H'XX'
180:                                      ;-----------------------------------
181:                                      ;Senden:     H'01' Startbefehl
182:          BCF      a1;;;;;;;;;
183:
```

```
184:          BSF      b1;;;;;;;;;
185:          CALL     UP_wait05s
186:          CALL     UP_GetType          ;Empfangen:  H'41'=Digital
187:                                       ;ODER        H'23'=NegCon
188:                                       ;ODER        H'73'=Analogue Red LED
189:                                       ;ODER        H'53'=Analogue Green LED
190:                                       ;-----------------------------------
191:                                       ;Senden:     H'42' Datenanfrage
192:          BCF      b1;;;;;;;;;
193:
194:          BSF      c1;;;;;;;;;
195:          CALL     UP_wait05s
196:          CALL     UP_Status           ;Empfangen:  H'5A'=Status:READY
197:                                       ;-----------------------------------
198:                                       ;Senden:     H'00' Idle
199:          BCF      c1;;;;;;;;;
200:
201:          BSF      d1;;;;;;;;;
202:          CALL     UP_wait05s
203:          CALL     UP_linke_btns       ;Empfangen:  H'XX'
204:                                       ;-----------------------------------
205:                                       ;Senden:     H'00' Idle
206:          BCF      d1;;;;;;;;;
207:
208:          BSF      a2;;;;;;;;;
209:          CALL     UP_wait05s
210:          CALL     UP_rechte_btns      ;Empfangen:  H'XX'
211:                                       ;-----------------------------------
212:                                       ;Senden:     H'00' Idle
213:          BCF      a2;;;;;;;;;
214:
215:          BSF      b2;;;;;;;;;
216:          CALL     UP_wait05s
217:          CALL     UP_sende0           ;#########################
218:          CALL     UP_sende0           ;Hier kommen die Joysticks
219:          CALL     UP_sende0           ;die abgefragt werden aber
220:          CALL     UP_sende0           ;nicht gespeichert
221:          BCF      b2;;;;;;;;;
222:
223:
224:
225:
226:          GOTO     main
227:
228:
229: ;******************
230: ;* Unterprogramme *
231: ;******************
232:
233: ;PSX_TxRx:
234: ;  FOR idx = 0 TO 7
235: ;    PsxCmd = psxOut.LOWBIT(idx)                 setup command bit
236: ;    PsxClk = ClockMode                          clock the bit (low)
237: ;    psxIn.LOWBIT(idx) = PsxDat                  get data bit
238: ;    PsxClk = ~ClockMode                         r(high)
239: ;  NEXT
240:
241: ;#############################################################################
242: UP_Start
243:          ;H'01' (Bitfolge 00000001) muss im LSB-Verfahren über
244:          ;a_command gesendet werden.
245:
246:          ;Bit 1 Senden
247:          CALL     UP_wait250khz
```

```
248:          BSF       a_command          ; "1" senden
249:          BCF       a_clock            ; Clock LOW
250:          CALL      UP_wait250khz
251:          ;BTFSC    e_data             ; e_data in Register speichern
252:          BSF       a_clock            ; Clock HIGH
253:
254:          ;Bit 0 Senden
255:          CALL      UP_wait250khz
256:          BCF       a_command          ; "0" senden
257:          BCF       a_clock            ; Clock LOW
258:          CALL      UP_wait250khz
259:          ;BTFSC    e_data             ; e_data in Register speichern
260:          BSF       a_clock            ; Clock HIGH
261:
262:          ;Bit 0 Senden
263:          CALL      UP_wait250khz
264:          BCF       a_command          ; "0" senden
265:          BCF       a_clock            ; Clock LOW
266:          CALL      UP_wait250khz
267:          ;BTFSC    e_data             ; e_data in Register speichern
268:          BSF       a_clock            ; Clock HIGH
269:
270:          ;Bit 0 Senden
271:          CALL      UP_wait250khz
272:          BCF       a_command          ; "0" senden
273:          BCF       a_clock            ; Clock LOW
274:          CALL      UP_wait250khz
275:          ;BTFSC    e_data             ; e_data in Register speichern
276:          BSF       a_clock            ; Clock HIGH
277:
278:          ;Bit 0 Senden
279:          CALL      UP_wait250khz
280:          BCF       a_command          ; "0" senden
281:          BCF       a_clock            ; Clock LOW
282:          CALL      UP_wait250khz
283:          ;BTFSC    e_data             ; e_data in Register speichern
284:          BSF       a_clock            ; Clock HIGH
285:
286:          ;Bit 0 Senden
287:          CALL      UP_wait250khz
288:          BCF       a_command          ; "0" senden
289:          BCF       a_clock            ; Clock LOW
290:          CALL      UP_wait250khz
291:          ;BTFSC    e_data             ; e_data in Register speichern
292:          BSF       a_clock            ; Clock HIGH
293:
294:          ;Bit 0 Senden
295:          CALL      UP_wait250khz
296:          BCF       a_command          ; "0" senden
297:          BCF       a_clock            ; Clock LOW
298:          CALL      UP_wait250khz
299:          ;BTFSC    e_data             ; e_data in Register speichern
300:          BSF       a_clock            ; Clock HIGH
301:
302:          ;Bit 0 Senden
303:          CALL      UP_wait250khz
304:          BCF       a_command          ; "0" senden
305:          BCF       a_clock            ; Clock LOW
306:          CALL      UP_wait250khz
307:          ;BTFSC    e_data             ; e_data in Register speichern
308:          BSF       a_clock            ; Clock HIGH
309:
310: RETURN
311: ;*****************************************************************************
```

```
312: UP_GetType
313:           ;H'42' (Bitfolge 01000010) muss im LSB-Verfahren über
314:           ;a_command gesendet werden.
315:
316:           ;Bit 0 Senden
317:           CALL     UP_wait250khz
318:           BCF      a_command          ; "0" senden
319:           BCF      a_clock            ; Clock LOW
320:           CALL     UP_wait250khz
321:           ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
322:           BSF      a_clock            ; Clock HIGH
323:
324:
325:           ;Bit 1 Senden
326:           CALL     UP_wait250khz
327:           BSF      a_command          ; "1" senden
328:           BCF      a_clock            ; Clock LOW
329:           CALL     UP_wait250khz
330:           ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
331:           BSF      a_clock            ; Clock HIGH
332:
333:
334:           ;Bit 0 Senden
335:           CALL     UP_wait250khz
336:           BCF      a_command          ; "0" senden
337:           BCF      a_clock            ; Clock LOW
338:           CALL     UP_wait250khz
339:           ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
340:           BSF      a_clock            ; Clock HIGH
341:
342:
343:           ;Bit 0 Senden
344:           CALL     UP_wait250khz
345:           BCF      a_command          ; "0" senden
346:           BCF      a_clock            ; Clock LOW
347:           CALL     UP_wait250khz
348:           ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
349:           BSF      a_clock            ; Clock HIGH
350:
351:
352:           ;Bit 0 Senden
353:           CALL     UP_wait250khz
354:           BCF      a_command          ; "0" senden
355:           BCF      a_clock            ; Clock LOW
356:           CALL     UP_wait250khz
357:           ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
358:           BSF      a_clock            ; Clock HIGH
359:
360:
361:           ;Bit 0 Senden
362:           CALL     UP_wait250khz
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:          ;Bit 1 Senden
371:          CALL     UP_wait250khz
372:          BSF      a_command          ; "1" senden
373:          BCF      a_clock            ; Clock LOW
374:          CALL     UP_wait250khz
375:          ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
376:          BSF      a_clock            ; Clock HIGH
377:
378:
379:          ;Bit 0 Senden
380:          CALL     UP_wait250khz
381:          BCF      a_command          ; "0" senden
382:          BCF      a_clock            ; Clock LOW
383:          CALL     UP_wait250khz
384:          ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
385:          BSF      a_clock            ; Clock HIGH
386:
387:
388: RETURN
389: ;#########################################################################
390: UP_Status
391:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
392:          ;a_command gesendet werden.
393:
394:          ;Bit 0 Senden
395:          CALL     UP_wait250khz
396:          BCF      a_command          ; "0" senden
397:          BCF      a_clock            ; Clock LOW
398:          CALL     UP_wait250khz
399:          ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
400:          BSF      a_clock            ; Clock HIGH
401:
402:          ;Bit 0 Senden
403:          CALL     UP_wait250khz
404:          BCF      a_command          ; "0" senden
405:          BCF      a_clock            ; Clock LOW
406:          CALL     UP_wait250khz
407:          ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
408:          BSF      a_clock            ; Clock HIGH
409:
410:          ;Bit 0 Senden
411:          CALL     UP_wait250khz
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:          CALL     UP_wait250khz
420:          BCF      a_command          ; "0" senden
421:          BCF      a_clock            ; Clock LOW
422:          CALL     UP_wait250khz
423:          ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
424:          BSF      a_clock            ; Clock HIGH
425:
426:          ;Bit 0 Senden
427:          CALL     UP_wait250khz
```

```
428:          BCF       a_command          ; "0" senden
429:          BCF       a_clock            ; Clock LOW
430:          CALL      UP_wait250khz
431:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
432:          BSF       a_clock            ; Clock HIGH
433:
434:          ;Bit 0 Senden
435:          CALL      UP_wait250khz
436:          BCF       a_command          ; "0" senden
437:          BCF       a_clock            ; Clock LOW
438:          CALL      UP_wait250khz
439:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
440:          BSF       a_clock            ; Clock HIGH
441:
442:          ;Bit 0 Senden
443:          CALL      UP_wait250khz
444:          BCF       a_command          ; "0" senden
445:          BCF       a_clock            ; Clock LOW
446:          CALL      UP_wait250khz
447:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
448:          BSF       a_clock            ; Clock HIGH
449:
450:          ;Bit 0 Senden
451:          CALL      UP_wait250khz
452:          BCF       a_command          ; "0" senden
453:          BCF       a_clock            ; Clock LOW
454:          CALL      UP_wait250khz
455:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
456:          BSF       a_clock            ; Clock HIGH
457:
458: RETURN
459: ;*****************************************************************************
460: UP_linke_btns
461:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
462:          ;a_command gesendet werden.
463:
464:          ; Pfeiltaste nach links <-
465:          ;######################
466:          ;Bit 0 Senden
467:          CALL      UP_wait250khz
468:          BCF       a_command          ; "0" senden
469:          BCF       a_clock            ; Clock LOW
470:          CALL      UP_wait250khz
471:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
472:          BSF       c1
473:          BSF       a_clock            ; Clock HIGH
474:
475:          ; Pfeiltaste nach unten \/
476:          ;######################
477:          ;Bit 0 Senden
478:          CALL      UP_wait250khz
479:          BCF       a_command          ; "0" senden
480:          BCF       a_clock            ; Clock LOW
481:          CALL      UP_wait250khz
482:          BTFSC     e_data             ; e_data in Register speichern - controller
    type
483:          BSF       a1
484:          BSF       a_clock            ; Clock HIGH
485:
```

```
486:          ; Pfeiltaste nach rechts ->
487:          ;#####################
488:          ;Bit 0 Senden
489:          CALL     UP_wait250khz
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      b1
495:          BSF      a_clock            ; Clock HIGH
496:
497:          ; Pfeiltaste nach oben /\
498:          ;#####################
499:          ;Bit 0 Senden
500:          CALL     UP_wait250khz
501:          BCF      a_command          ; "0" senden
502:          BCF      a_clock            ; Clock LOW
503:          CALL     UP_wait250khz
504:          BTFSC    e_data             ; e_data in Register speichern - controller
     type
505:          BSF      d1
506:          BSF      a_clock            ; Clock HIGH
507:
508:          ; Start
509:          ;#####################
510:          ;Bit 0 Senden
511:          CALL     UP_wait250khz
512:          BCF      a_command          ; "0" senden
513:          BCF      a_clock            ; Clock LOW
514:          CALL     UP_wait250khz
515:          ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
516:          BSF      a_clock            ; Clock HIGH
517:
518:          ; Joy-R
519:          ;#####################
520:          ;Bit 0 Senden
521:          CALL     UP_wait250khz
522:          BCF      a_command          ; "0" senden
523:          BCF      a_clock            ; Clock LOW
524:          CALL     UP_wait250khz
525:          ;BTFSC   e_data             ; e_data in Register speichern - controller
     type
526:          BSF      a_clock            ; Clock HIGH
527:
528:          ; Joy-L
529:          ;#####################
530:          ;Bit 0 Senden
531:          CALL     UP_wait250khz
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:          ; Select
539:          ;#####################
540:          ;Bit 0 Senden
541:          CALL     UP_wait250khz
542:          BCF      a_command          ; "0" senden
543:          BCF      a_clock            ; Clock LOW
544:          CALL     UP_wait250khz
```

```
545:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
546:          BSF      a_clock          ; Clock HIGH
547:
548: RETURN
549: ;#############################################################################
550: UP_rechte_btns
551:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
552:          ;a_command gesendet werden.
553:
554:          ; [] - Quadrat Taste
555:          ;#####################
556:          ;Bit 0 Senden
557:          CALL     UP_wait250khz
558:          BCF      a_command        ; "0" senden
559:          BCF      a_clock          ; Clock LOW
560:          CALL     UP_wait250khz
561:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
562:          BSF      a_clock          ; Clock HIGH
563:
564:          ; X - X Taste
565:          ;#####################
566:          ;Bit 0 Senden
567:          CALL     UP_wait250khz
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:          ; O - Kreis Taste
575:          ;#####################
576:          ;Bit 0 Senden
577:          CALL     UP_wait250khz
578:          BCF      a_command        ; "0" senden
579:          BCF      a_clock          ; Clock LOW
580:          CALL     UP_wait250khz
581:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
582:          BSF      a_clock          ; Clock HIGH
583:
584:          ; /.\ - Dreick Taste
585:          ;#####################
586:          ;Bit 0 Senden
587:          CALL     UP_wait250khz
588:          BCF      a_command        ; "0" senden
589:          BCF      a_clock          ; Clock LOW
590:          CALL     UP_wait250khz
591:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
592:          BSF      a_clock          ; Clock HIGH
593:
594:          ; R1 - Schultertaste
595:          ;#####################
596:          ;Bit 0 Senden
597:          CALL     UP_wait250khz
598:          BCF      a_command        ; "0" senden
599:          BCF      a_clock          ; Clock LOW
600:          CALL     UP_wait250khz
601:          ;BTFSC   e_data           ; e_data in Register speichern - controller
    type
602:          BSF      a_clock          ; Clock HIGH
```

```
603:
604:            ; L1 - Schultertaste
605:            ;#####################
606:            ;Bit 0 Senden
607:            CALL     UP_wait250khz
608:            BCF      a_command          ; "0" senden
609:            BCF      a_clock            ; Clock LOW
610:            CALL     UP_wait250khz
611:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
612:            BSF      a_clock            ; Clock HIGH
613:
614:            ; R2 - Schultertaste
615:            ;#####################
616:            ;Bit 0 Senden
617:            CALL     UP_wait250khz
618:            BCF      a_command          ; "0" senden
619:            BCF      a_clock            ; Clock LOW
620:            CALL     UP_wait250khz
621:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
622:            BSF      a_clock            ; Clock HIGH
623:
624:            ; L2 - Schultertaste
625:            ;#####################
626:            ;Bit 0 Senden
627:            CALL     UP_wait250khz
628:            BCF      a_command          ; "0" senden
629:            BCF      a_clock            ; Clock LOW
630:            CALL     UP_wait250khz
631:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
632:            BSF      a_clock            ; Clock HIGH
633:
634: RETURN
635: ;#############################################################################
636: UP_sende0
637:            ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
638:            ;a_command gesendet werden.
639:
640:            ;Bit 0 Senden
641:            CALL     UP_wait250khz
642:            BCF      a_command          ; "0" senden
643:            BCF      a_clock            ; Clock LOW
644:            CALL     UP_wait250khz
645:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
646:            BSF      a_clock            ; Clock HIGH
647:
648:            ;Bit 0 Senden
649:            CALL     UP_wait250khz
650:            BCF      a_command          ; "0" senden
651:            BCF      a_clock            ; Clock LOW
652:            CALL     UP_wait250khz
653:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
654:            BSF      a_clock            ; Clock HIGH
655:
656:            ;Bit 0 Senden
657:            CALL     UP_wait250khz
658:            BCF      a_command          ; "0" senden
659:            BCF      a_clock            ; Clock LOW
660:            CALL     UP_wait250khz
661:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
```

```
662:          BSF      a_clock           ; Clock HIGH
663:
664:          ;Bit 0 Senden
665:          CALL     UP_wait250khz
666:          BCF      a_command         ; "0" senden
667:          BCF      a_clock           ; Clock LOW
668:          CALL     UP_wait250khz
669:          ;BTFSC   e_data            ; e_data in Register speichern - controller
      type
670:          BSF      a_clock           ; Clock HIGH
671:
672:          ;Bit 0 Senden
673:          CALL     UP_wait250khz
674:          BCF      a_command         ; "0" senden
675:          BCF      a_clock           ; Clock LOW
676:          CALL     UP_wait250khz
677:          ;BTFSC   e_data            ; e_data in Register speichern - controller
      type
678:          BSF      a_clock           ; Clock HIGH
679:
680:          ;Bit 0 Senden
681:          CALL     UP_wait250khz
682:          BCF      a_command         ; "0" senden
683:          BCF      a_clock           ; Clock LOW
684:          CALL     UP_wait250khz
685:          ;BTFSC   e_data            ; e_data in Register speichern - controller
      type
686:          BSF      a_clock           ; Clock HIGH
687:
688:          ;Bit 0 Senden
689:          CALL     UP_wait250khz
690:          BCF      a_command         ; "0" senden
691:          BCF      a_clock           ; Clock LOW
692:          CALL     UP_wait250khz
693:          ;BTFSC   e_data            ; e_data in Register speichern - controller
      type
694:          BSF      a_clock           ; Clock HIGH
695:
696:          ;Bit 0 Senden
697:          CALL     UP_wait250khz
698:          BCF      a_command         ; "0" senden
699:          BCF      a_clock           ; Clock LOW
700:          CALL     UP_wait250khz
701:          ;BTFSC   e_data            ; e_data in Register speichern - controller
      type
702:          BSF      a_clock           ; Clock HIGH
703:
704: RETURN
705: ;#########################################################################
706: UP_sende0_schleife
707:          MOVLW    get_btn_stats_cnt          ;Startwert innere Schleife
708:          MOVWF    cnt_get_btn_stats
709:
710: loop_btn_stats
711:                   CALL     UP_wait250khz
712:                   BCF      a_command         ; "0" senden
713:                   BCF      a_clock           ; Clock LOW
714:                   CALL     UP_wait250khz
715:                   ;BTFSC   e_data            ; e_data in Register speichern -
      controller type
716:                   BSF      a_clock           ; Clock HIGH
717:
718:                   DECFSZ   cnt_get_btn_stats, F       ; innnere Schleife
719:                   GOTO     loop_btn_stats
```

```
720:
721: RETURN
722: ;*****************************************************************************
723: UP_wait
724:          MOVLW     out_cnt   ;Startwert äussere Schleife
725:          MOVWF     zaehler_out
726:
727: loop_out
728:          MOVLW     mid_cnt   ;Startwert mittlere Schleife
729:          MOVWF     zaehler_mid
730:
731: loop_mid
732:          MOVLW     in_cnt    ;Startwert innere Schleife
733:          MOVWF     zaehler_in
734:
735: loop_in
736:          DECFSZ    zaehler_in, F; innnere Schleife
737:          GOTO      loop_in
738:
739:          DECFSZ    zaehler_mid, F; mittlere Schleife
740:          GOTO      loop_mid
741:
742:          DECFSZ    zaehler_out, F; äussere Schleife
743:          GOTO      loop_out
744:
745: RETURN
746: ;*****************************************************************************
747: UP_wait250khz
748:                               ;19 cycles
749:          movlw     0x06
750:          movwf     wait250khz
751: Delay_0
752:          decfsz    wait250khz, f
753:          goto      Delay_0
754:
755:                               ;1 cycle
756:          nop
757:
758: RETURN
759: ;*****************************************************************************
760: UP_wait05s
761:                               ;2499999 cycles
762:          movlw     0x16
763:          movwf     wait05s
764:          movlw     0x74
765:          movwf     wait05s_1
766:          movlw     0x06
767:          movwf     wait05s_2
768: Delay_05
769:          decfsz    wait05s, f
770:          goto      $+2
771:          decfsz    wait05s_1, f
772:          goto      $+2
773:          decfsz    wait05s_2, f
774:          goto      Delay_05
775:
776:                               ;1 cycle
777:          nop
778: RETURN
779: ;*****************************************************************************
780: END
781:
```