```
1: ; ******************************************************
2: ; * Projekt06.asm - Playstation Controller an PIC16F84A *
3: ; ******************************************************
4: ;
5: ;        KOMMT NOCH....
6: ;
7: ;
8: ; ************************************************************
9: ; *                        Changelog 0_3                     *
10: ; ************************************************************
11: ;
12: ; UP_Bit0 hinzugefügt
13: ; UP_Bit1 hinzugefügt
14: ;
15: ; UP_Start an UP_BitX angepasst (Codeverkürzung - Abarbeitungszeit=LÄNGER)
16: ; UP_Sende0 an UP_BitX angepasst                    "
17: ; UP_Sende0_schleife an UP_BitX angepasst          "
18: ;
19: ;
20: ;*****************************************************************************
21: ;* Bestimmung des Prozessortyps für den Assembler und das Programmiergerät *
22: ;*****************************************************************************
23:
24:                  LIST p=16F84A
25:
26:
27: ;*****************************************************************
28: ;* Includedatei für den 16F84A einbinden (vordef. Reg. und Konst.) *
29: ;*****************************************************************
30:
31:                  #include <p16f84A.INC>
32:
33: ; Diese Datei enthält Vordefinitionen für wichtige Register und Konstanten.
34: ; (Z.B. gibt es die Konstante PORTB mit der sich ohne Angabe der
35: ; absoluten Adresse H'0006' der Port B des Prozessors ansprechen lässt)
36:
37:
38: ;*********************************************************
39: ;* Konfigurationseinstellungen für IC-Prog vordefinieren *
40: ;*********************************************************
41:
42:                  __CONFIG _PWRTE_ON & _CP_OFF & _HS_OSC & _WDT_OFF
43:
44: ; Hier werden verschiedene Prozessoreigenschaften festgelegt:
45: ; _PWRTE_ON schaltet den Power Up Timer ein, d.h. der Prozessor wartet nach
46: ;           dem Einschalten ca. 70ms mit dem Programmstart, um sicher zu sein,
47: ;           dass alle angeschlossene Peripherie bereit ist.
48: ; _CP_OFF schaltet die Code-Protection des Prozesors aus. Damit ist das im
    Prozessor
49: ;        befindliche Programm jederzeit auslesbar und überschreibbar.
50: ; _HS_OSC spezifiziert einen Quarzoszillator (Highspeed) als Zeitbasis für den
    Prozessor.
51: ; _WDT_OFF schaltet den Watchdog-Timer des Prozesors aus.
52:
53:
54: ;*********************************
55: ;* Register / Variablen  festlegen *
56: ;*********************************
57: ; hier werden Adressen von Registern / Variablen festgelegt. Diese werden
    beginnend
58: ; mit der Adresse H'20' aufsteigend vergeben.
59:
60:
61:          CBLOCK   H'20'
```

```
 62:                     zaehler_in          ;Zaehler innere Schleife
 63:                     zaehler_mid         ;Zaehler mittlere Schleife
 64:                     zaehler_out         ;Zaehler aeussere Schleife
 65:
 66:                     cnt_get_btn_stats ;Get_Btn_Stats Schleife
 67:
 68:                     wait250khz          ;250kHz zaehler
 69:
 70:                     wait05s
 71:                     wait05s_1
 72:                     wait05s_2
 73:
 74:
 75:           ENDC
 76:
 77:           in_cnt              EQU D'0'          ;Startwert inner Schleife
 78:           mid_cnt             EQU D'0'          ;Startwert mittlere ""
 79:           out_cnt             EQU D'1'          ;Startwert äussere ""
 80:
 81:           get_btn_stats_cnt EQU D'7'          ;Startwert Get_Btn_Stats Schleife
 82:
 83:
 84: ;************************
 85: ;* Konstanten festlegen *
 86: ;************************
 87:
 88:
 89:
 90: ; ********************************************************************
 91: ; * Definition von einzelnen Bits in einem Register / in einer Variable *
 92: ; ********************************************************************
 93:
 94: ;#####################################################################
 95: #DEFINE   e_data             PORTA, 0          ; Data            Eingang
 96: #DEFINE   a_command          PORTA, 1          ; Command         Ausgang
 97: #DEFINE   a_clock            PORTA, 2          ; Takt            Ausgang
 98: #DEFINE   a_att              PORTA, 3          ; ATT             Ausgang
 99: ;#####################################################################
100: #DEFINE   a_test             PORTB, 0          ; Tets LED        Ausgang
101: ;#####################################################################
102: #DEFINE   a1                 PORTB, 0          ; BCD-Decoder     Ausgang
103: #DEFINE   b1                 PORTB, 1          ; BCD-Decoder     Ausgang
104: #DEFINE   c1                 PORTB, 2          ; BCD-Decoder     Ausgang
105: #DEFINE   d1                 PORTB, 3          ; BCD-Decoder     Ausgang
106: ;#####################################################################
107: #DEFINE   a2                 PORTB, 4          ; BCD-Decoder     Ausgang
108: #DEFINE   b2                 PORTB, 5          ; BCD-Decoder     Ausgang
109: #DEFINE   c2                 PORTB, 6          ; BCD-Decoder     Ausgang
110: #DEFINE   d2                 PORTB, 7          ; BCD-Decoder     Ausgang
111: ;#####################################################################
112: #DEFINE   bank1              STATUS, RP0
113: ;#####################################################################
114:
115:
116:
117: ; In diesem Beispiel steht das Wort ir_sensor für Bit 0 von Port A und das Wort
     motor
118: ; für das Bit 0 von Port B.
119:
120:
121: ;*****************
122: ;* Programmstart *
123: ;*****************
124:
```

```
125:          ORG       H'00'              ; Das Programm wird ab Speicherstelle 0 in
    den Speicher geschrieben
126:          GOTO      init               ; Springe zur Grundinitialisierung der Ports
    A und B
127:
128:
129: ;*******************
130: ;* Initialisierung *
131: ;*******************
132:
133: init      BSF       bank1              ; wechsle zu Registerbank 1 (spezielle
    Register)
134:
135:          MOVLW     B'00000001'
136:          MOVWF     TRISA              ; RA0 Eingang (RA1 bis RA3 sind Ausgänge)
137:          MOVLW     B'00000000'
138:          MOVWF     TRISB              ; RB0 bis RB7 sind Ausgänge
139:
140:          BCF       bank1              ; wechsle zu Registerbank 0 (normaler
    Speicherbereich)
141:
142:          CLRF      PORTA              ; Port A löschen
143:          CLRF      PORTB              ; Port B löschen
144:
145: ; Die Register TRISA und TRISB legen fest, welche Bits in den jeweiligen Ports
    Ein- bzw.
146: ; Ausgänge sind. Eine '1' an der entsprechenden Stelle setzt das Bit des Ports
    als Ein-
147: ; gang eine '0' setzt das Bit als Ausgang.
148:
149:
150:
151: ;******************
152: ;* Hauptprogramm *
153: ;******************
154:
155: ;#####################################################################
156: ;#DEFINE e_data             PORTA, 0          ; Data             Eingang #
157: ;#DEFINE a_command          PORTA, 1          ; Command          Ausgang #
158: ;#DEFINE a_clock            PORTA, 2          ; Takt             Ausgang #
159: ;#DEFINE a_att              PORTA, 3          ; ATT              Ausgang #
160: ;#####################################################################
161: ;#DEFINE a_test             PORTB, 0          ; Tets LED         Ausgang #
162: ;#####################################################################
163: ;#DEFINE a1                 PORTB, 0          ; BCD-Decoder      Ausgang #
164: ;#DEFINE b1                 PORTB, 1          ; BCD-Decoder      Ausgang #
165: ;#DEFINE c1                 PORTB, 2          ; BCD-Decoder      Ausgang #
166: ;#DEFINE d1                 PORTB, 3          ; BCD-Decoder      Ausgang #
167: ;#####################################################################
168: ;#DEFINE a2                 PORTB, 4          ; BCD-Decoder      Ausgang #
169: ;#DEFINE b2                 PORTB, 5          ; BCD-Decoder      Ausgang #
170: ;#DEFINE c2                 PORTB, 6          ; BCD-Decoder      Ausgang #
171: ;#DEFINE d2                 PORTB, 7          ; BCD-Decoder      Ausgang #
172: ;#####################################################################
173:
174:
175: main
176:          CALL      UP_wait05s
177:          BCF       a_att              ; ATT auf LOW das der Controller
178:                                       ; die Daten annimmt
179:
180:          BSF       a1;;;;;;;;;
181:          CALL      UP_wait05s
182:          CALL      UP_Start           ;Empfangen:  H'XX'
```

```
183:                                    ;-----------------------------------
184:                                    ;Senden:     H'01' Startbefehl
185:          BCF       a1;;;;;;;;;
186:
187:          BSF       b1;;;;;;;;;
188:          CALL      UP_wait05s
189:          CALL      UP_GetType      ;Empfangen:  H'41'=Digital
190:                                    ;ODER        H'23'=NegCon
191:                                    ;ODER        H'73'=Analogue Red LED
192:                                    ;ODER        H'53'=Analogue Green LED
193:                                    ;-----------------------------------
194:                                    ;Senden:     H'42' Datenanfrage
195:          BCF       b1;;;;;;;;;
196:
197:          BSF       c1;;;;;;;;;
198:          CALL      UP_wait05s
199:          CALL      UP_Status       ;Empfangen:  H'5A'=Status:READY
200:                                    ;-----------------------------------
201:                                    ;Senden:     H'00' Idle
202:          BCF       c1;;;;;;;;;
203:
204:          BSF       d1;;;;;;;;;
205:          CALL      UP_wait05s
206:          CALL      UP_linke_btns   ;Empfangen:  H'XX'
207:                                    ;-----------------------------------
208:                                    ;Senden:     H'00' Idle
209:          BCF       d1;;;;;;;;;
210:
211:          BSF       a2;;;;;;;;;
212:          CALL      UP_wait05s
213:          CALL      UP_rechte_btns  ;Empfangen:  H'XX'
214:                                    ;-----------------------------------
215:                                    ;Senden:     H'00' Idle
216:          BCF       a2;;;;;;;;;
217:
218:          BSF       b2;;;;;;;;;
219:          CALL      UP_wait05s
220:          CALL      UP_sende0       ;#########################
221:          CALL      UP_sende0       ;Hier kommen die Joysticks
222:          CALL      UP_sende0       ;die abgefragt werden aber
223:          CALL      UP_sende0       ;nicht gespeichert
224:          BCF       b2;;;;;;;;;
225:
226:
227:
228:
229:          GOTO      main
230:
231:
232: ;******************
233: ;* Unterprogramme *
234: ;******************
235:
236: ;PSX_TxRx:
237: ;  FOR idx = 0 TO 7
238: ;    PsxCmd = psxOut.LOWBIT(idx)                 setup command bit
239: ;    PsxClk = ClockMode                          clock the bit (low)
240: ;    psxIn.LOWBIT(idx) = PsxDat                  get data bit
241: ;    PsxClk = ~ClockMode                         r(high)
242: ;  NEXT
243:
244: ;#############################################################################
245: UP_Start
246:          ;H'01' (Bitfolge 00000001) muss im LSB-Verfahren über
```

```
247:          ;a_command gesendet werden.
248:
249: ;0
250:          ;Bit 1 Senden
251:          CALL     UP_Bit1
252: ;1
253:          ;Bit 0 Senden
254:          CALL     UP_Bit0
255: ;2
256:          ;Bit 0 Senden
257:          CALL     UP_Bit0
258: ;3
259:          ;Bit 0 Senden
260:          CALL     UP_Bit0
261: ;4
262:          ;Bit 0 Senden
263:          CALL     UP_Bit0
264: ;5
265:          ;Bit 0 Senden
266:          CALL     UP_Bit0
267: ;6
268:          ;Bit 0 Senden
269:          CALL     UP_Bit0
270: ;7
271:          ;Bit 0 Senden
272:          CALL     UP_Bit0
273:
274: RETURN
275: ;###############################################################################
276: UP_GetType
277:          ;H'42' (Bitfolge 01000010) muss im LSB-Verfahren über
278:          ;a_command gesendet werden.
279:
280: ;0
281:          ;Bit 0 Senden
282:          CALL     UP_Bit0
283: ;1
284:          ;Bit 1 Senden
285:          CALL     UP_Bit1
286: ;2
287:          ;Bit 0 Senden
288:          CALL     UP_Bit0
289: ;3
290:          ;Bit 0 Senden
291:          CALL     UP_Bit0
292: ;4
293:          ;Bit 0 Senden
294:          CALL     UP_Bit0
295: ;5
296:          ;Bit 0 Senden
297:          CALL     UP_Bit0
298: ;6
299:          ;Bit 1 Senden
300:          CALL     UP_Bit1
301: ;7
302:          ;Bit 0 Senden
303:          CALL     UP_Bit0
304:
305:
306: RETURN
307: ;###############################################################################
308: UP_Status
309:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
310:          ;a_command gesendet werden.
```

```
311:
312: ;0
313:           ;Bit 0 Senden
314:           CALL      UP_Bit0
315: ;1
316:           ;Bit 0 Senden
317:           CALL      UP_Bit0
318: ;2
319:           ;Bit 0 Senden
320:           CALL      UP_Bit0
321: ;3
322:           ;Bit 0 Senden
323:           CALL      UP_Bit0
324: ;4
325:           ;Bit 0 Senden
326:           CALL      UP_Bit0
327: ;5
328:           ;Bit 0 Senden
329:           CALL      UP_Bit0
330: ;6
331:           ;Bit 0 Senden
332:           CALL      UP_Bit0
333: ;7
334:           ;Bit 0 Senden
335:           CALL      UP_Bit0
336:
337: RETURN
338: ;*****************************************************************************
339: UP_linke_btns
340:           ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
341:           ;a_command gesendet werden.
342:
343:           ; Pfeiltaste nach links <-
344:           ;#####################
345:           ;Bit 0 Senden
346:           CALL      UP_wait250khz
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       c1
352:           BSF       a_clock            ; Clock HIGH
353:
354:           ; Pfeiltaste nach unten \/
355:           ;#####################
356:           ;Bit 0 Senden
357:           CALL      UP_wait250khz
358:           BCF       a_command          ; "0" senden
359:           BCF       a_clock            ; Clock LOW
360:           CALL      UP_wait250khz
361:           BTFSC     e_data             ; e_data in Register speichern - controller
    type
362:           BSF       a1
363:           BSF       a_clock            ; Clock HIGH
364:
365:           ; Pfeiltaste nach rechts ->
366:           ;#####################
367:           ;Bit 0 Senden
368:           CALL      UP_wait250khz
369:           BCF       a_command          ; "0" senden
370:           BCF       a_clock            ; Clock LOW
371:           CALL      UP_wait250khz
372:           BTFSC     e_data             ; e_data in Register speichern - controller
    type
```

```
373:          BSF       b1
374:          BSF       a_clock           ; Clock HIGH
375:
376:          ; Pfeiltaste nach oben /\
377:          ;#####################
378:          ;Bit 0 Senden
379:          CALL      UP_wait250khz
380:          BCF       a_command         ; "0" senden
381:          BCF       a_clock           ; Clock LOW
382:          CALL      UP_wait250khz
383:          BTFSC     e_data            ; e_data in Register speichern - controller
    type
384:          BSF       d1
385:          BSF       a_clock           ; Clock HIGH
386:
387:          ; Start
388:          ;#####################
389:          ;Bit 0 Senden
390:          CALL      UP_wait250khz
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:          ; Joy-R
398:          ;#####################
399:          ;Bit 0 Senden
400:          CALL      UP_wait250khz
401:          BCF       a_command         ; "0" senden
402:          BCF       a_clock           ; Clock LOW
403:          CALL      UP_wait250khz
404:          ;BTFSC    e_data            ; e_data in Register speichern - controller
    type
405:          BSF       a_clock           ; Clock HIGH
406:
407:          ; Joy-L
408:          ;#####################
409:          ;Bit 0 Senden
410:          CALL      UP_wait250khz
411:          BCF       a_command         ; "0" senden
412:          BCF       a_clock           ; Clock LOW
413:          CALL      UP_wait250khz
414:          ;BTFSC    e_data            ; e_data in Register speichern - controller
    type
415:          BSF       a_clock           ; Clock HIGH
416:
417:          ; Select
418:          ;#####################
419:          ;Bit 0 Senden
420:          CALL      UP_wait250khz
421:          BCF       a_command         ; "0" senden
422:          BCF       a_clock           ; Clock LOW
423:          CALL      UP_wait250khz
424:          ;BTFSC    e_data            ; e_data in Register speichern - controller
    type
425:          BSF       a_clock           ; Clock HIGH
426:
427: RETURN
428: ;#########################################################################
429: UP_rechte_btns
430:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
431:          ;a_command gesendet werden.
```

```
432:
433:            ; [] - Quadrat Taste
434:            ;#####################
435:            ;Bit 0 Senden
436:            CALL     UP_wait250khz
437:            BCF      a_command          ; "0" senden
438:            BCF      a_clock            ; Clock LOW
439:            CALL     UP_wait250khz
440:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
441:            BSF      a_clock            ; Clock HIGH
442:
443:            ; X - X Taste
444:            ;#####################
445:            ;Bit 0 Senden
446:            CALL     UP_wait250khz
447:            BCF      a_command          ; "0" senden
448:            BCF      a_clock            ; Clock LOW
449:            CALL     UP_wait250khz
450:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
451:            BSF      a_clock            ; Clock HIGH
452:
453:            ; O - Kreis Taste
454:            ;#####################
455:            ;Bit 0 Senden
456:            CALL     UP_wait250khz
457:            BCF      a_command          ; "0" senden
458:            BCF      a_clock            ; Clock LOW
459:            CALL     UP_wait250khz
460:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
461:            BSF      a_clock            ; Clock HIGH
462:
463:            ; /.\ - Dreick Taste
464:            ;#####################
465:            ;Bit 0 Senden
466:            CALL     UP_wait250khz
467:            BCF      a_command          ; "0" senden
468:            BCF      a_clock            ; Clock LOW
469:            CALL     UP_wait250khz
470:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
471:            BSF      a_clock            ; Clock HIGH
472:
473:            ; R1 - Schultertaste
474:            ;#####################
475:            ;Bit 0 Senden
476:            CALL     UP_wait250khz
477:            BCF      a_command          ; "0" senden
478:            BCF      a_clock            ; Clock LOW
479:            CALL     UP_wait250khz
480:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
481:            BSF      a_clock            ; Clock HIGH
482:
483:            ; L1 - Schultertaste
484:            ;#####################
485:            ;Bit 0 Senden
486:            CALL     UP_wait250khz
487:            BCF      a_command          ; "0" senden
488:            BCF      a_clock            ; Clock LOW
489:            CALL     UP_wait250khz
490:            ;BTFSC   e_data             ; e_data in Register speichern - controller
    type
```

```
491:          BSF      a_clock           ; Clock HIGH
492:
493:          ; R2 - Schultertaste
494:          ;#####################
495:          ;Bit 0 Senden
496:          CALL     UP_wait250khz
497:          BCF      a_command         ; "0" senden
498:          BCF      a_clock           ; Clock LOW
499:          CALL     UP_wait250khz
500:          ;BTFSC   e_data            ; e_data in Register speichern - controller
    type
501:          BSF      a_clock           ; Clock HIGH
502:
503:          ; L2 - Schultertaste
504:          ;#####################
505:          ;Bit 0 Senden
506:          CALL     UP_wait250khz
507:          BCF      a_command         ; "0" senden
508:          BCF      a_clock           ; Clock LOW
509:          CALL     UP_wait250khz
510:          ;BTFSC   e_data            ; e_data in Register speichern - controller
    type
511:          BSF      a_clock           ; Clock HIGH
512:
513: RETURN
514: ;#############################################################################
515: UP_sende0
516:          ;H'00' (Bitfolge 00000000) muss im LSB-Verfahren über
517:          ;a_command gesendet werden.
518: ;0
519:          ;Bit 0 Senden
520:          CALL     UP_Bit0
521: ;1
522:          ;Bit 0 Senden
523:          CALL     UP_Bit0
524: ;2
525:          ;Bit 0 Senden
526:          CALL     UP_Bit0
527: ;3
528:          ;Bit 0 Senden
529:          CALL     UP_Bit0
530: ;4
531:          ;Bit 0 Senden
532:          CALL     UP_Bit0
533: ;5
534:          ;Bit 0 Senden
535:          CALL     UP_Bit0
536: ;6
537:          ;Bit 0 Senden
538:          CALL     UP_Bit0
539: ;7
540:          ;Bit 0 Senden
541:          CALL     UP_Bit0
542:
543: RETURN
544: ;#############################################################################
545: UP_sende0_schleife
546:          MOVLW    get_btn_stats_cnt          ;Startwert innere Schleife
547:          MOVWF    cnt_get_btn_stats
548:
549: loop_btn_stats
550:                   CALL     UP_Bit0
551:
552:                   DECFSZ   cnt_get_btn_stats, F       ; innnere Schleife
```

```
553:                    GOTO     loop_btn_stats
554:
555: RETURN
556: ;#########################################################################
557: UP_wait
558:            MOVLW    out_cnt  ;Startwert äussere Schleife
559:            MOVWF    zaehler_out
560:
561: loop_out
562:            MOVLW    mid_cnt  ;Startwert mittlere Schleife
563:            MOVWF    zaehler_mid
564:
565: loop_mid
566:            MOVLW    in_cnt   ;Startwert innere Schleife
567:            MOVWF    zaehler_in
568:
569: loop_in
570:            DECFSZ   zaehler_in, F; innnere Schleife
571:            GOTO     loop_in
572:
573:            DECFSZ   zaehler_mid, F; mittlere Schleife
574:            GOTO     loop_mid
575:
576:            DECFSZ   zaehler_out, F; äussere Schleife
577:            GOTO     loop_out
578:
579: RETURN
580: ;#########################################################################
581: UP_wait250khz
582:                              ;19 cycles
583:            movlw    0x06
584:            movwf    wait250khz
585: Delay_0
586:            decfsz   wait250khz, f
587:            goto     Delay_0
588:
589:                              ;1 cycle
590:            nop
591:
592: RETURN
593: ;#########################################################################
594: UP_wait05s
595:                              ;2499999 cycles
596:            movlw    0x16
597:            movwf    wait05s
598:            movlw    0x74
599:            movwf    wait05s_1
600:            movlw    0x06
601:            movwf    wait05s_2
602: Delay_05
603:            decfsz   wait05s, f
604:            goto     $+2
605:            decfsz   wait05s_1, f
606:            goto     $+2
607:            decfsz   wait05s_2, f
608:            goto     Delay_05
609:
610:                              ;1 cycle
611:            nop
612: RETURN
613: ;#########################################################################
614: UP_Bit0
615:            ;Bit 0 Senden
616:            CALL     UP_wait250khz
```

```
617:          BCF       a_command          ; "0" senden
618:          BCF       a_clock            ; Clock LOW
619:          CALL      UP_wait250khz
620:          ;BTFSC    e_data             ; e_data in Register speichern - controller
    type
621:          BSF       a_clock            ; Clock HIGH
622: RETURN
623: ;#########################################################################
624: UP_Bit1
625:          ;Bit 1 Senden
626:          CALL      UP_wait250khz
627:          BSF       a_command          ; "1" 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: RETURN
633: ;#########################################################################
634: END
635:
```