```
1               ;GPS INS autonomous vehicle control system
2               ;Alexey Morozov
3               ;
4               ;This program makes AVR Atmega 128 output control signals
5               ;in response to inputs from GPS and an IMU
6               ;GPS should be connected to USART 1, LCD to USART 0
7               ;Accelerometer X should be connected to port A/D pin 1
8               ;ac-r Y to pin 2 and Rateout (turn/rotation) to pin 3
9               ;
10              ;The Clock Fuse on the chip should be set to 8 MHz
11              ;  (otherwise the baud rate code for the LCD will be wrong).
12
13
14
15  .include "m128def.inc"  ;Includes the 128 definitions file
16  .device ATmega128       ;Compiles the rest of the code for ATmega 128
17
18  .org 0x0000             ;Places the following code from address 0x0000
19
20
21  ;;;;;;;;;The interrupt vectors table (not quite used right now)
22
23  jmp RESET               ; Reset Handler
24  jmp EXT_INT0            ; IRQ0 Handler
25  jmp EXT_INT1            ; IRQ1 Handler
26  jmp EXT_INT2            ; IRQ2 Handler
27  jmp EXT_INT3            ; IRQ3 Handler
28  jmp EXT_INT4            ; IRQ4 Handler
29  jmp EXT_INT5            ; IRQ5 Handler
30  jmp EXT_INT6            ; IRQ6 Handler
31  jmp EXT_INT7            ; IRQ7 Handler
32  jmp TIM2_COMP           ; Timer2 Compare Handler
33  jmp TIM2_OVF            ; Timer2 Overflow Handler
34  jmp TIM1_CAPT           ; Timer1 Capture Handler
35  jmp TIM1_COMPA          ; Timer1 CompareA Handler
36  jmp TIM1_COMPB          ; Timer1 CompareB Handler
37  jmp TIM1_OVF            ; Timer1 Overflow Handler
38  jmp TIM0_COMP           ; Timer0 Compare Handler
39  jmp TIM0_OVF            ; Timer0 Overflow Handler
40  jmp SPI_STC             ; SPI Transfer Complete Handler
41  jmp USART0_RXC          ; USART0 RX Complete Handler
42  jmp USART0_DRE          ; USART0,UDR Empty Handler
43  jmp USART0_TXC          ; USART0 TX Complete Handler
44  jmp ADChandler          ; ADC Conversion Complete Handler
45  jmp EE_RDY              ; EEPROM Ready Handler
46  jmp ANA_COMP            ; Analog Comparator Handler
47  jmp TIM1_COMPC          ; Timer1 CompareC Handler
48  jmp TIM3_CAPT           ; Timer3 Capture Handler
49  jmp TIM3_COMPA          ; Timer3 CompareA Handler
50  jmp TIM3_COMPB          ; Timer3 CompareB Handler
51  jmp TIM3_COMPC          ; Timer3 CompareC Handler
52  jmp TIM3_OVF            ; Timer3 Overflow Handler
53  jmp USART1_RXC          ; USART1 RX Complete Handler
54  jmp USART1_DRE          ; USART1,UDR Empty Handler
55  jmp USART1_TXC          ; USART1 TX Complete Handler
56  jmp TWI                 ; Two-wire Serial Interface Interrupt Handler
57  jmp SPM_RDY             ; SPM Ready Handler
58
59
60      .set GPSsentence = 0x100        ;That's where every GPS sentence
61                                      ;  will be written one by one  (one
62                                      ;  written, read, analyzed and
63                                      ;  overwritten by the next one (in
64                                      ;  that sequence)).
```

```
65
66        .set LCDdoneFlag = 0x190        ;A service flag location for
67                                        ;  LCD-send routine
68
69        .set GUTCtimeASCII = 0x200      ;That's where Time is stored (ASCII
70                                        ;  chars)
71        .set GStatusASCII = 0x20C       ;That's where Status of GPS fix is
72                                        ;  stored
73        .set GLatitudeASCII = 0x210     ;That's where Latitude is stored
74        .set GAhemisphASCII = 0x21C     ;That's where Lat' hemisphere. is
75                                        ;  stored
76        .set GLongitudeASCII = 0x220    ;That's where Longitude is stored
77        .set GOhemisphASCII = 0x22D     ;That's where Long' hemisphere. is
78                                        ;  stored
79        .set GSpeedKnotsASCII = 0x230   ;That's where Speed is stored
80        .set GHeadingASCII = 0x238      ;That's where Heading is stored
81
82        .set LCDstopper = 0x250         ;This place is to catch LCD if it
83                                        ;  went out of range
84
85                                        ;R16,R17,R18,R19 are always a
86                                        ;  general purpose registers, don't
87                                        ;  use them for anything important.
88
89                                        ;The rest of the used and named
90                                        ;  registers is described here
91        .def counter = r21
92
93
94
95    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Main program
96
97    RESET:
98
99        ldi r16, high(RAMEND)   ;Set stack pointer to top of RAM
100       out SPH, r16
101       ldi r16, low(RAMEND)
102       out SPL, r16
103
104
105   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Set up GPS
106
107       ldi r16, 12             ;Set baud rate, 38400 bps (12, 0x0c)
108       sts UBRR1L, r16
109
110       ldi r16, 0x90           ;Enable the receiver and the receive
111       sts UCSR1B,r16          ;  complete interrupt (RxCIEn)
112
113
114   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Set up LCD
115
116       ldi r16, 51             ;Set baud rate, 9600 bps (51, 0x33)
117       out UBRR0L, r16
118
119       ldi r16, 0x08           ;Enable the TX-r
120       out UCSR0B,r16
121
122
123   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Set up LED array
124
125       ldi r16, 0xff           ;Make PORTC an all-output port
126       out DDRC, r16
127
128
```

```
129  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;Set up Z register for GPS sentence-writing
130                               ;  (to the data space)
131
132      ldi r31, high(GPSsentence)
133      ldi r30, low(GPSsentence)
134      push r30                 ;Save the Z register's current target
135      push r31                 ;  onto the stack
136
137      clr r16                  ;The LCD done flag is responsible for
138      sts LCDdoneFlag, r16     ;  stopping the transmission once a
139                               ;  comma is encountered. Initially
140                               ;  it's cleared.
141
142      ldi r16,','              ;To prevent the LCD from accidentally
143      sts LCDstopper, r16      ;  outputting the whole data space to
144      sts LCDstopper+1, r16    ;  the screen, these three commas are
145      sts LCDstopper+2, r16    ;  set in place (after the data)
146
147      ldi counter,3
148
149  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
150
151      sei                      ;Enable Interrupts
152
153
154  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;Start the send process
155
156  MainLoop:
157
158  XStatusThree:
159
160      cpi counter, 3           ;These next four partitions represent a
161      brne XLatitudeTwo        ;  switch statement that selects what data
162                               ;  will be outputted next
163      ldi r27, high(GStatusASCII)
164      ldi r26, low(GStatusASCII)
165      rjmp LCDWriteLoop
166
167
168  XLatitudeTwo:
169
170      cpi counter, 2
171      brne XLongitudeOne
172
173
174      ldi r27, high(GLatitudeASCII)
175      ldi r26, low(GLatitudeASCII)
176      rjmp LCDWriteLoop
177
178
179  XLongitudeOne:
180
181      cpi counter, 1
182      brne XSpeedKnotsZero
183
184
185      ldi r27, high(GLongitudeASCII)
186      ldi r26, low(GLongitudeASCII)
187      rjmp LCDWriteLoop
188
189
190  XSpeedKnotsZero:
191
192      ;cpi counter, 0
```

```
193      ;brne LCDWriteLoop
194
195
196      ldi r27, high(GSpeedKnotsASCII)
197      ldi r26, low(GSpeedKnotsASCII)
198
199
200
201  LCDWriteLoop:               ;This is a send-value-to-LCD loop
202
203      ldi r16, 0xff           ;Microprocessor shall slow down (wait
204      ldi r20, 5              ;  for the LCD screen) by a factor of 1024
205
206  SlowDownLoop:
207
208      dec r16                 ;Slow down loop: decrease r16 register
209      nop                     ;  until it reaches zero, then repeat
210      nop                     ;Do nothing
211      nop
212      nop
213      nop
214      nop
215      tst r16
216      brne SlowDownLoop       ;If not at zero yet, keep decreasing
217
218      dec r20
219      nop
220      nop
221      nop
222      nop
223      nop
224      nop
225      tst r20
226      brne SlowDownLoop
227
228
229  TransmitCheck:
230
231      in r16,UCSR0A           ;Check if the USART0 is ready to transmit
232      andi r16,0x20           ;Select only the Data Register Empty bit
233      tst r16                 ;See if the result is zero: if it was zero,
234                              ;  the transaction is not complete yet, so
235                              ;  wait (loop again); if it was not zero,
236                              ;  the transaction is complete, so send the
237                              ;  data
238      breq TransmitCheck      ;That's the branch (on equaling zero)
239
240      lds r16, 0x190          ;This part of code checks if flag was set
241      tst r16                 ;  to end the transmission, see
242      brmi EndTransmissionPiece; EndTransmissionPiece
243
244
245  Transmit:
246
247      ld r16,X+               ;LOAD the letter
248      cpi r16, ','            ;See if it is a comma
249
250      brne TransmitCont
251      ldi r16,0x80            ;If it was a comma, set the flag and move
252      sts LCDdoneFlag,r16     ;  on to the next piece of data (f.e. speed)
253
254      ldi r16,254             ;Prepare the LCD for moving on to the next
255                              ;  type of data by entering a control char
256
```

```
257  TransmitCont:
258      out UDR0,r16            ;SEND the letter
259
260
261  Transmitted:
262
263      rjmp LCDWriteLoop       ;When done, repeat
264
265
266
267
268  EndTransmissionPiece:       ;This code moves the LCD cursor position
269                              ;  to a new place every time a comma is
270                              ;  encountered
271
272
273  StatusThree:                ;This set of four pieces of code is
274                              ;  responcible for making sure that the next
275                              ;  displayed value doesn't overlap the
276                              ;  previous one
277
278      cpi counter, 3
279      brne LatitudeTwo
280
281      ldi r16, (128+2)
282      rjmp PreExitLCDWrite
283
284
285  LatitudeTwo:
286
287      cpi counter, 2
288      brne LongitudeOne
289
290      ldi r16, (128+12)
291      rjmp PreExitLCDWrite
292
293
294  LongitudeOne:
295
296      cpi counter, 1
297      brne SpeedKnotsZero
298
299      ldi r16, (128+72)
300      rjmp PreExitLCDWrite
301
302
303  SpeedKnotsZero:
304
305      ldi r16, 128
306
307
308  PreExitLCDWrite:
309
310      out UDR0,r16            ;Send it to the LCD
311      dec counter             ;Decrement the data type counter
312
313      brne ExitLCDWrite       ;Exit if not to the last piece of data yet
314
315  PreClearLCD:
316      ldi counter,3           ;If the counter is exhausted, refill it
317
318
319
320  ClearLCD:
```

```
321      ldi r16, 0xff          ;Microprocessor shall slow down (wait
322      ldi r20, 0xff          ;  for the LCD screen) by a factor of 1024
323
324  SlowDownLoop2:
325
326      dec r16                ;Slow down loop: decrease counter register
327      nop                    ;  until it reaches zero, repeat
328      nop
329      nop
330      nop
331      nop
332      nop                    ;Do nothing
333      tst r16
334      brne SlowDownLoop2     ;If not at zero yet, keep decreasing
335
336      dec r20                ;Slow down loop: decrease counter register
337      nop                    ;  until it reaches zero, repeat
338      nop
339      nop
340      nop
341      nop
342      nop                    ;Do nothing
343      tst r20
344      brne SlowDownLoop2     ;If not at zero yet, keep decreasing
345
346  TransmitCheck2:
347
348      in r16,UCSR0A          ;Check if the USART0 is ready to transmit
349      andi r16,0x20          ;Select only the Data Register Empty bit
350      tst r16
351      breq TransmitCheck2    ;That's the branch (on equaling zero)
352
353      ldi r16, 254
354      out UDR0,r16
355
356      ldi r16, 0xff          ;Microprocessor shall slow down (wait
357      ldi r20, 0xff          ;  for the LCD screen) by a factor of 1024
358
359  SlowDownLoop3:
360
361      dec r16                ;Slow down loop: decrease counter register
362      nop                    ;  until it reaches zero, repeat
363      nop
364      nop
365      nop
366      nop
367      nop                    ;Do nothing
368      tst r16
369      brne SlowDownLoop3     ;If not at zero yet, keep decreasing
370
371      dec r20                ;Slow down loop: decrease counter register
372      nop                    ;  until it reaches zero, repeat
373      nop
374      nop
375      nop
376      nop
377      nop                    ;Do nothing
378      tst r20
379      brne SlowDownLoop3     ;If not at zero yet, keep decreasing
380
381  TransmitCheck3:
382
383      in r16,UCSR0A          ;Check if the USART0 is ready to transmit
384      andi r16,0x20          ;Select only the Data Register Empty bit
```

```
385      tst r16
386      breq TransmitCheck3     ;That's the branch (on equaling zero)
387
388      ldi r16, 1
389      out UDR0,r16
390
391
392  ExitLCDWrite:
393      clr r16                 ;Clear the flag
394      sts LCDdoneFlag, r16
395
396      rjmp MainLoop           ;Start this sending process again
397
398
399
400  ;Analog to digital conversion
401
402
403
404  ;Kalman Filtering
405
406
407
408  ;Outputting the on-off decision
409
410
411
412  ;Outputting the steer decision
413
414
415
416  ;Repeat
417
418
419
420
421  ;;;;;;;;;;NOT used yet, but these are the labels for interrupt handlers
422
423  EXT_INT0:           ; IRQ0 Handler
424  EXT_INT1:           ; IRQ1 Handler
425  EXT_INT2:           ; IRQ2 Handler
426  EXT_INT3:           ; IRQ3 Handler
427  EXT_INT4:           ; IRQ4 Handler
428  EXT_INT5:           ; IRQ5 Handler
429  EXT_INT6:           ; IRQ6 Handler
430  EXT_INT7:           ; IRQ7 Handler
431  TIM2_COMP:          ; Timer2 Compare Handler
432  TIM2_OVF:           ; Timer2 Overflow Handler
433  TIM1_CAPT:          ; Timer1 Capture Handler
434  TIM1_COMPA:         ; Timer1 CompareA Handler
435  TIM1_COMPB:         ; Timer1 CompareB Handler
436  TIM1_OVF:           ; Timer1 Overflow Handler
437  TIM0_COMP:          ; Timer0 Compare Handler
438  TIM0_OVF:           ; Timer0 Overflow Handler
439  SPI_STC:            ; SPI Transfer Complete Handler
440  USART0_RXC:         ; USART0 RX Complete Handler
441  USART0_DRE:         ; USART0,UDR Empty Handler
442  USART0_TXC:         ; USART0 TX Complete Handler
443  ADChandler:         ; ADC Conversion Complete Handler
444  EE_RDY:             ; EEPROM Ready Handler
445  ANA_COMP:           ; Analog Comparator Handler
446  TIM1_COMPC:         ; Timer1 CompareC Handler
447  TIM3_CAPT:          ; Timer3 Capture Handler
448  TIM3_COMPA:         ; Timer3 CompareA Handler
```

```
449  TIM3_COMPB:          ; Timer3 CompareB Handler
450  TIM3_COMPC:          ; Timer3 CompareC Handler
451  TIM3_OVF:            ; Timer3 Overflow Handler
452
453  USART1_RXC:          ; USART1 RX Complete Handler
454
455      pop r18                  ;Get the return address from the stack
456      pop r17
457      pop r31                  ;Get the last known Zregister
458      pop r30
459
460      push r16
461
462      in r16, SREG             ;Since interrupts don't save the SREG, let's
463      push r16                 ;  do that for them
464
465      ldi r16, 0b01101011      ;Blink an LED (attached to Portc) to signify
466      out PORTC, r16           ;  that the GPS reception has begun
467
468
469      lds r16,UDR1             ;Receive data (a character) from USART1
470                               ;  (from GPS)
471      cpi r16, '$'             ;See if received character (char) is a
472                               ;  dollar sign (24 in hex) (stands for the
473                               ;  start of the NMEA message).
474      breq DollarSign          ;If it is, go to the
475                               ;  start-of-sentence-handler (DollarSign)
476      cpi r16, 0xD             ;See if received character (char) is a
477                               ;  carriage return sign (D in hex)
478                               ;  (stands for the "end of message" more or
479                               ;  less).
480      breq CarriageReturnSign  ;If it is, go to the end-of-sentence-handler
481                               ;  (CarriageReturnSign)
482
483  NormalChar:                  ;If it was neither beginning of sentence,
484                               ;  nor end, just process it
485
486      rjmp StoreGPSchar        ;Just proceed to store this character to the
487                               ;  memory
488
489  DollarSign:
490
491      ldi r31, high(GPSsentence)  ;Reset the Z register to the beginning
492      ldi r30, low(GPSsentence)   ;  of the data space to signify that a
493                                  ;  new sentence has started
494      rjmp StoreGPSchar           ;And then store it to the memory
495
496  CarriageReturnSign:
497
498      rcall GPSsentenceReader  ;If we got here, the sentence must have
499                               ;  ended, so go ahead and read it.
500      ;rjmp StoreGPSchar       ;Rjmp is not needed, since the next thing is
501                               ;  store anyways.
502
503  StoreGPSchar:
504
505      st Z+, r16               ;Store this character at the next available
506                               ;  location in data space (pointed to by Z
507                               ;  register and post-incremented)
508
509      ldi r16, 0               ;Turn off the LED once all data was received
510      out PORTC, r16
511
512      pop r16                  ;Now we put back the SREG that was stored on
```

```
513      out SREG,r16            ;  the stack in the beginning the interrupt.
514
515      pop r16
516
517      push r30                ;Recover the Z register's condition to its
518      push r31                ;  original state
519      push r17                ;Recover the return address
520      push r18
521      reti                    ;Return from Interrupt when done
522
523  GPSsentenceReader:          ;This subroutine reads recent GPS sentence,
524                              ;  takes out various pieces (velocity,
525                              ;  heading, position...), sorts them and
526                              ;  stores them (ASCII representation) in
527                              ;  separate locations.
528
529      push r30                ;Save the Z register's current target
530      push r31                ;  address onto the stack
531      push r28                ;Save the Y register's current target
532      push r29                ;  address onto the stack
533      push r19                ;Save r19 register onto the stack
534
535      ldi r31, high(GPSsentence)  ;Reset the Z register to the beginning
536      ldi r30, low(GPSsentence)   ;  of the data space to signify that we
537                                  ;  are reading the new sentence
538
539
540      ld r19, Z+
541      cpi r19, '$'            ;See if received character (char) is a
542                              ;  dollar sign (24 in hex) (stands for the
543                              ;  start of the NMEA message).
544      brne GPSsrDone
545
546      ld r19, Z+
547      cpi r19, 'G'            ;See if it's G (next char in message)
548      brne GPSsrDone
549
550      ld r19, Z+
551      cpi r19, 'P'            ;See if it's P
552      brne GPSsrDone
553
554      ld r19, Z+
555      cpi r19, 'R'            ;See if it's R (for RMC message - this
556                              ;  message is all I care about right now).
557      brne GPSsrDone
558
559      inc r30
560      inc r30
561      inc r30
562
563      ldi r29, high(GUTCtimeASCII)    ;The Y register will tell the
564      ldi r28, low(GUTCtimeASCII)     ;  micro processor where to store
565                                      ;  the bits of information.
566  StoreGUTCtimeASCII:
567
568      ld r19, Z+
569      st Y+, r19
570      cpi r19, ','
571      brne StoreGUTCtimeASCII
572
573      ldi r29, high(GStatusASCII)
574      ldi r28, low(GStatusASCII)
575
576  StoreGStatusASCII:
```

```
577
578       ld r19, Z+
579       st Y+, r19
580       cpi r19, ','
581       brne StoreGStatusASCII
582
583       ldi r29, high(GLatitudeASCII)
584       ldi r28, low(GLatitudeASCII)
585
586   StoreGLatitudeASCII:
587
588       ld r19, Z+
589       st Y+, r19
590       cpi r19, ','
591       brne StoreGLatitudeASCII
592
593       ldi r29, high(GAhemisphASCII)
594       ldi r28, low(GAhemisphASCII)
595
596   StoreGAhemisphASCII:
597
598       ld r19, Z+
599       st Y+, r19
600       cpi r19, ','
601       brne StoreGAhemisphASCII
602
603       ldi r29, high(GLongitudeASCII)
604       ldi r28, low(GLongitudeASCII)
605
606   StoreGLongitudeASCII:
607
608       ld r19, Z+
609       st Y+, r19
610       cpi r19, ','
611       brne StoreGLongitudeASCII
612
613       ldi r29, high(GOhemisphASCII)
614       ldi r28, low(GOhemisphASCII)
615
616   StoreGOhemisphASCII:
617
618       ld r19, Z+
619       st Y+, r19
620       cpi r19, ','
621       brne StoreGOhemisphASCII
622
623       ldi r29, high(GSpeedKnotsASCII)
624       ldi r28, low(GSpeedKnotsASCII)
625
626   StoreGSpeedKnotsASCII:
627
628       ld r19, Z+
629       st Y+, r19
630       cpi r19, ','
631       brne StoreGSpeedKnotsASCII
632
633       ldi r29, high(GHeadingASCII)
634       ldi r28, low(GHeadingASCII)
635
636   StoreGHeadingASCII:
637
638       ld r19, Z+
639       st Y+, r19
640       cpi r19, ','
```

```
641      brne StoreGHeadingASCII
642
643
644  GPSsrDone:
645      pop r19                 ;Recover the r19 register from the stack
646      pop r29                 ;Recover the Y register's condition to its
647      pop r28                 ;  original state
648      pop r31                 ;Recover the Z register's condition to its
649      pop r30                 ;  original state
650
651      ret                     ;Return from the subroutine
652
653
654  USART1_DRE:          ; USART1,UDR Empty Handler
655  USART1_TXC:          ; USART1 TX Complete Handler
656  TWI:                 ; Two-wire Serial Interface Interrupt Handler
657  SPM_RDY:             ; SPM Ready Handler
658  jmp RESET
659
```