

```

1
2  /*****
3  * By CCSC compiler
4  * File Name: 20121010_MK204_TalkingTester.c
5  * Description: check Volt & Amp with MCP3421 and play ATP3011
6  * RA5 out to ATP3011 RESET
7  * RA4 out to ATP3011 RXD
8  * RA2 in to PLAY, active low
9  * RA1 in to Volt/Amp, low means Amp
10 * RA0 in to x1 / x10 Volt,
11 low means x10
12 * RC5 out to LED, active high
13 * RC4 in to ATP3011 play satatus
14 * RC1 out to MCP3421 SCL
15 * RC0 in/out to MCP3421 SDA
16 * RC2 AD input for repeat period setting potentiometer
17 * COPYRIGHT 2012 MYCOMKITS.COM, owned by CNET LIMITED
18 * 当プログラムの著作権は、製作者「マイコンキットドットコム運営 有限会社クネット」に帰属します。
19 * 著作権を放棄していませんが、当プログラムを使った学習の中でプログラムを自由に変更してお使いください。
20 *****/
21 //
22 #include <16f1825.h>
23 #DEVICE ADC=8
24 #fuses INTRC IO, NOWDT, MCLR, NOBROWNOUT, PUT, NOLVP, NOPROTECT, NODEBUG
25 #fuses NOPCPD, NOCLKOUT, NOWRT, NOIESO, NOFCMEN, PLL_SW, NOSTVREN
26 #use delay(CLOCK = 4000000)
27 #use rs232(BAUD = 9600, XMIT = PIN_A4)
28 #use I2C(Master, SLOW, SDA = PIN_C1, SCL = PIN_C0, ADDRESS=0xA0)
29 //
30 // initial data on EEPROM
31 // #ROM getenv("EEPROM_ADDRESS") = {0x7F, 0x08, 0xF5, 0xC3, 0x7F, 0x08, 0x1E, 0xB8}
32 #ROM getenv("EEPROM_ADDRESS") = {0x7F, 0x08, 0xF5, 0xC3, 0x7F, 0, 0, 0} //
33 // #define Bmode 0x00 //port B all output
34 #define Amode 0x0F //port A
35 #define Cmode 0x14 //port C
36 #byte OSCCON = 0x099
37 #byte WPUA = 0x20C
38 #byte OPTION_REG = 0x095
39
40 // variables
41 int low_buffer = 0;
42 int high_buffer = 0;
43 long adc_in = 0; //16 bits
44 long preadc_in = 0;
45 float value = 0; //actual value that is converted
46 int mmode = 0; // 1=amp, 2=voltx1, 3=voltx10
47 int en_repeat = 0; // 1 = enable auto repeat every 5 sec
48 float thres;
49 int repeat_time = 50;
50 int minusflag = 0; // 1 means minus value
51 int value_setting = 50; // max 256
52 int pre_value = 50;
53 int fast_repeat = 0; // 1 means repeating fast without unit
54 float volt10_coef = 1.07; // x10 volt coefficient
55 float amp_coef = 1.00; // amp coefficient
56 //float amp_coef = 1.04; // amp coefficient
57 float dummy;
58 long int j;
59 int factory_v_data = 0;
60 int factory_a_data = 0;
61 float talked_value = 0;
62 //
63 // prototyping
64 #separate
65 void initializing();
66 #separate
67 void readadc();
68 #separate
69 void tellvalue();
70 #separate
71 void tell31();
72 #separate
73 void tell13();
74 #separate
75 void get_setting();
76 #separate
77 void init_coef();
78 #separate
79 void WRITE_FLOAT_EEPROM();
80 #separate

```

```

81 void READ_FLOAT_EEPROM();
82 //
83 //
84 //write_float_to_eeprom;
85 //
86 #separate
87 void WRITE_FLOAT_EEPROM(int addr, float data)
88 {
89     int8 i;
90     for (i = 0; i < 4; i++)
91     {
92         write_eeprom(addr + i, *((int8*)&data + i));
93         delay_ms(20);
94     }
95 }
96 //
97 //read_float_from_eeprom;
98 //
99 #separate
100 float READ_FLOAT_EEPROM(int addr)
101 {
102     int8 i;
103     float data=0;
104     for (i = 0; i < 4; i++)
105     {
106         *((int8*)&data + i) = read_eeprom(addr + i);
107         delay_ms(20);
108     }
109     return(data);
110 }
111 //
112 // timer 1 sub function
113 // wait time count down the timer value at every 100msec.
114 #int timer1
115 void isr1(void)
116 {
117     set_timer1(0x7FFF); // setting timer 1. almost 100msec
118     if(repeat_time > 0) repeat_time = repeat_time -1;
119     else repeat_time = 0;
120 }
121 //
122 // main function
123 void main()
124 {
125     initializing();
126     //
127     // main loop
128     while(1)
129     {
130         //
131         get_setting(); // get repeat value from potentiometer
132         //check mode sw
133         if(input(PIN_A1) == 0) // check mode sw, A1 shows amp
134         {
135             if(mmode != 1)
136             {
137                 //
138                 delay_ms(1000);
139                 while(input(PIN_C4)==0) {} //busy
140                 disable_interrupts(GLOBAL);
141                 printf("den'ryuusokuteimo'-dodesu");
142                 putc(0x0D);
143                 //
144                 i2c_start();
145                 i2c_write(0xD0); //write mode
146                 //16bit, continuous conversion, PGA = 8V/V
147                 i2c_write(0x1B); // mode 1, amp mode
148                 i2c_stop();
149                 //
150                 enable_interrupts(GLOBAL);
151             }
152             mmode = 1; // amp mode
153             thres = 232; // 8*0x7FFF/1000, 1% of full range, actual adc value
154         }
155         else if(input(PIN_A0) == 0) // check mode sw, A0 shows x10 volt
156         {
157             if(mmode != 3)
158             {
159                 //
160                 delay_ms(1000);

```

```

161         while(input(PIN_C4)==0) {} //busy
162         disable_interrupts(GLOBAL);
163         printf("den'a'tusokuteimo'-do/ nijyu'-bo'rutore'njidesu");
164         putc(0x0D);
165         //
166         i2c_start();
167         i2c_write(0xD0); //write mode
168         //16bit, continuous conversion, PGA = 1V/V
169         i2c_write(0x18); // mode 3, volt x10 mode
170         i2c_stop();
171         enable_interrupts(GLOBAL);
172     }
173     mmode = 3; // volt x10 mode
174     thres = 327; // 32764/100, 1% of full range, actual adc value
175 }
176 else
177 {
178     if(mmode != 2)
179     {
180         //
181         delay_ms(1000);
182         while(input(PIN_C4)==0) {} //busy
183         disable_interrupts(GLOBAL);
184         printf("den'atusokuteimo'-do/ nibo'rutore'njidesu");
185         putc(0x0D);
186         //
187         i2c_start();
188         i2c_write(0xD0); //write mode
189         //16bit, continuous conversion, PGA = 1V/V
190         i2c_write(0x18); // mode 2, volt x1 mode
191         i2c_stop();
192         enable_interrupts(GLOBAL);
193     }
194     mmode = 2; // volt x1 mode
195     thres = 327; // 32764/100, 1% of full range, actual adc value
196 }
197 //
198 // get value from MCP3421 and play it by one of 3 ways
199 //while(input(PIN_C4)==0) {} //busy
200 readadc();
201 // play it, if over threshold
202 if((adc_in > (preadc_in + thres)) || (adc_in < (preadc_in - thres)))
203 {
204     // play
205     tellvalue();
206     delay_ms(1000);
207     readadc();
208     tellvalue();
209 }
210 // play it, if repeat time = 0
211 else if((repeat_time == 0) && (en_repeat == 1)) // repeat time will be decremented by
int1
212 {
213     // play
214     tellvalue();
215     repeat_time = value_setting; // 5sec = 50x100msec repeat cycle
216 }
217 // play it, if pressing PLAY switch
218 else if(input(PIN_A2) == 0)
219 {
220     // play
221     tellvalue();
222 }
223 }
224 //
225 }
226 //
227 #separate
228 void initializing()
229 {
230     OSCCON = 0xEA; // set 4MHz
231     //
232     set_tris_a(Amode); //
233     set_tris_c(Cmode); //
234     //
235     //PORT A PULLUPS(true); // pull up
236     OPTION_REG = OPTION_REG & 0x7F; // set 0 to WPUEN
237     WPUA = 0x3F; // pull up all A port
238     //
239     // A/D converter initialize AN6, RC2

```

```

240     setup_adc_ports(sAN6 | VSS_VDD);    // gnd to 5v
241     setup_adc(ADC_CLOCK_DIV_32);
242     //
243     // initialize timer 1 for repeat play
244     //
245     setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
246     set_timer1(0x7FFF);    // setting almost 104.4msec
247     enable_interrupts(INT_TIMER1);    // enable timer 1
248     //
249     //first, configure ADC
250     output_high(PIN_C0);
251     output_high(PIN_C1);
252     //
253     i2c_start();
254     i2c_write(0xD0); //write mode
255     //16bit, continuous conversion, PGA = 8V/V
256     if(input(PIN_A1) == 0) i2c_write(0x1B); // mode 1, amp mode
257     //16bit, continuous conversion, PGA = 1V/V
258     else i2c_write(0x18);
259     //i2c_write(0x1C); // for 18 bits
260     i2c_stop();
261     //
262     delay_us(100);
263     //
264     // reset ATP3011
265     output_high(PIN_A5);
266     output_low(PIN_A5);
267     delay_ms(500);
268     output_high(PIN_A5);
269     delay_ms(100);
270     //
271     printf("?");    //set baud rate
272     delay_ms(100);
273     //
274     init_coeff();    // initialize volt10 and amp coefficient
275     //
276     printf("to-kinngu/te'suta-/kido'-shima'shita");
277    putc(0x0D);
278     //
279     delay_ms(2000);
280     //
281     // LED off
282     output_low(PIN_C5);
283     //
284     // get initial value for repeat time
285     set_adc_channel(6); //must wait 65u, AN6, RC2
286     delay_us(100);
287     pre_value = read_adc();
288     repeat_time = 0;
289     //
290     enable_interrupts(GLOBAL);
291     //
292 }
293 //
294 #separate
295 void readadc()
296 {
297     //
298     disable_interrupts(GLOBAL);
299     //
300     i2c_start();
301     i2c_write(0xD1); //Read mode
302     high_buffer = i2c_read(1);
303     low_buffer = i2c_read(0);
304     i2c_stop();
305     //
306     enable_interrupts(GLOBAL);
307     //
308     // adc_value 16 bit length
309     adc_in = high_buffer;
310     adc_in <<= 8;
311     adc_in = adc_in + low_buffer;
312     if((adc_in & 0x8000) == 0) minusflag = 0;
313     else
314     {
315         minusflag = 1;
316         adc_in = 1-adc_in;    // get complement
317     }
318     //
319     value = adc_in; // value is float, adc_in is long int

```

```

320     //
321 }
322 //
323 #separate
324 void tellvalue()
325 {
326     //
327     preadc_in = adc_in; // keep talked value
328     // LED on
329     output_high(PIN_C5);
330     //
331     if(mmode == 1)
332     {
333         // amp mode
334         value = value*2.559/0x7FFF; //value is 0 to 26214
335         //
336         if(value > 2.04)
337         {
338             disable_interrupts(GLOBAL);
339             printf("#K");
340             putc(0x0D);
341             delay_ms(1000);
342             printf("chu'ui/ saida'itio/koeteima'su");
343             putc(0x0D);
344             enable_interrupts(GLOBAL);
345         }
346         else
347         {
348             if(minusflag == 1)
349             {
350                 value = value * amp_coef; // adjust for internal impedance
351             }
352             else value = value * amp_coef; // adjust for internal impedance
353             // adjust for internal impedance
354             if(value < 1.0)
355             {
356                 value = value * 1000;
357                 if(minusflag == 1) printf("mainasu");
358                 tell11();
359                 disable_interrupts(GLOBAL);
360                 printf("miri'anpea");
361                 putc(0x0D);
362                 enable_interrupts(GLOBAL);
363             }
364             else
365             {
366                 if(minusflag == 1) printf("mainasu");
367                 // over or equal 1.0
368                 tell13();
369                 disable_interrupts(GLOBAL);
370                 printf("anpea");
371                 putc(0x0D);
372                 enable_interrupts(GLOBAL);
373             }
374         }
375     }
376     else if(mmode == 2)
377     {
378         // volt x1 mode
379         value = value*2.048/0x7FFF; //value is 0 to 32767
380         //
381         if(value > 2.04)
382         {
383             disable_interrupts(GLOBAL);
384             printf("#K");
385             putc(0x0D);
386             delay_ms(1000);
387             printf("chu'ui/ saida'itio/koeteima'su");
388             putc(0x0D);
389             enable_interrupts(GLOBAL);
390         }
391         else
392         {
393             if(value < 1)
394             {
395                 value = value * 1000;
396                 if(minusflag == 1) printf("mainasu");
397                 tell11();
398                 disable_interrupts(GLOBAL);
399                 printf("miri'boruto");

```

```

400         putc(0x0D);
401         enable_interrupts(GLOBAL);
402     }
403     else
404     {
405         if(minusflag == 1) printf("mainasu");
406         // over or equal 1.0
407         tell13();
408         disable_interrupts(GLOBAL);
409         printf("bo'ruto");
410         putc(0x0D);
411         enable_interrupts(GLOBAL);
412     }
413 }
414 }
415 else if(mmode == 3)
416 {
417     // volt x10 mode
418     float tempo;
419     value = value*2.048/0x7FFF; //value is 0 to 32767
420     //
421     if(value > 2.04)
422     {
423         disable_interrupts(GLOBAL);
424         printf("#K");
425         putc(0x0D);
426         delay_ms(1000);
427         printf("chu'ui/ saida'itio/koeteima'su");
428         putc(0x0D);
429         enable_interrupts(GLOBAL);
430     }
431     else
432     {
433         if(minusflag == 1)
434         {
435             disable_interrupts(GLOBAL);
436             printf("mainasu");
437             enable_interrupts(GLOBAL);
438             // adjust for internal impedance
439             value = 10 * value * volt10_coef; // adjust for internal impedance
440         }
441         else value = 10 * value * volt10_coef; // adjust for internal impedance
442         //
443         if(value < 1)
444         {
445             value = value * 1000;
446             tell131();
447             disable_interrupts(GLOBAL);
448             printf("miri'boruto");
449             putc(0x0D);
450             enable_interrupts(GLOBAL);
451         }
452         else
453         {
454             // over or equal 1.0
455             disable_interrupts(GLOBAL);
456             putc(0x3C);
457             printf("NUMK VAL=");
458             printf("%2.2f", value);
459             putc(0x3E);
460             printf("bo'ruto");
461             putc(0x0D);
462             enable_interrupts(GLOBAL);
463         }
464     }
465 }
466 //
467 delay_ms(2000);
468 while(input(PIN_C4)==0) {} //busy
469 // LED off
470 output_low(PIN_C5);
471 }
472 //
473 //
474 #separate
475 void tell131()
476 {
477     disable_interrupts(GLOBAL);
478     putc(0x3C);
479     printf("NUMK VAL=");

```

```

480     printf("%3.1f",value);
481     putc(0x3E);
482     enable_interrupts(GLOBAL);
483 }
484 //
485 #separate
486 void tell13()
487 {
488     disable_interrupts(GLOBAL);
489     putc(0x3C);
490     printf("NUMK VAL=");
491     printf("%1.3f",value);
492     putc(0x3E);
493     enable_interrupts(GLOBAL);
494     //
495 }
496 //
497 // get repeat time value from potentiometer
498 #separate
499 void get_setting()
500 {
501     set_adc_channel(6); //must wait 65u, AN6, RC2
502     delay_us(100);
503     value_setting=read_adc(); // 0 to 256, 0V to 5V
504     if((value_setting > pre_value + 5) || (value_setting < pre_value - 5)) repeat_time = 0;
505     pre_value = value_setting; // 0 to 256, 0V to 5V
506     if(pre_value < 5) pre_value = 5;
507     if(pre_value > 250) pre_value = 250;
508     if(value_setting < 245)
509     {
510         en_repeat = 1;
511         if(value_setting < 15)
512         {
513             fast_repeat = 1;
514             value_setting = 0;
515         }
516         else fast_repeat = 0;
517     }
518     else en_repeat = 0;
519     //
520     //
521 }
522 // load stored coefficient data for x10 voltage and amp mode
523 #separate
524 void init_coeff()
525 {
526     if(input(PIN_A2) == 1) // if pressing PLAY sw, then calibrate
527     {
528         volt10_coef = READ_FLOAT_EEPROM(0); // load stored data
529         amp_coef = READ_FLOAT_EEPROM(4); // load stored data
530     }
531     else
532     {
533         if(input(PIN_A1) == 0) // mode 1, amp mode
534         {
535             printf("kousei/puro'guramu/kidousima'sita");
536             putc(0x0D);
537             delay_ms(3000);
538             while(input(PIN_C4)==0) {} //wait during busy
539             if(input(PIN_A2) == 0)
540             {
541                 printf("pureibo'tankara/yubi'o/hana'sitekudasai");
542                 putc(0x0D);
543                 delay_ms(4000);
544             }
545             while(input(PIN_C4)==0) {} //wait during busy
546             while(input(PIN_A2) == 0) {} // wait play switch
547             //
548             while(input(PIN_C4)==0) {} //busy
549             printf("den'ryuusokuteimo'-dodesu");
550             putc(0x0D);
551             delay_ms(4000);
552             while(input(PIN_C4)==0) {} //busy
553             printf("tya'imuga/naru'maeni/ pure'ibotan/o/osu'to");
554             printf(" /syukka'jino/de-tani/modo'sima'su");
555             putc(0x0D);
556             delay_ms(2000);
557             while(input(PIN_C4)==0) {} //busy
558             //
559             for(j=0; j<3000; j=j+1)

```

```

560 {
561     if(input(PIN_A2) == 0) // wait play switch
562     {
563         while(input(PIN_C4)==0) {} //busy
564         printf("#K");
565         putc(0x0D);
566         delay_ms(1000);
567         printf("syukka'jino/ de'-tao/ sette'isima'sita");
568         putc(0x0D);
569         delay_ms(2000);
570         while(input(PIN_C4)==0) {} //busy
571         //volt10 coef = 1.07;
572         amp_coef = 1.00;
573         WRITE_FLOAT_EEPROM(4, amp_coef);
574         factory_a_data = 1;
575         break;
576     }
577     delay_ms(1);
578 }
579 //delay_ms(3000);
580 printf("#J");
581 putc(0x0D);
582 delay_ms(1000);
583 if(factory_a_data == 0)
584 {
585     printf("hyakumiri/ anpe'ao /setu'zokusi / pu're'ibo'tan;o /osite;/kuda'sai");
586     putc(0x0D);
587     while(input(PIN_A2) == 1) {} // wait
588     if(input(PIN_A2) == 0) // wait play switch
589     {
590         readadc();
591         amp_coef = 1280.59/value; // should be around 1.1
592         WRITE_FLOAT_EEPROM(4, amp_coef);
593         printf("#K");
594         putc(0x0D);
595         delay_ms(1000);
596         while(input(PIN_C4)==0) {} //wait during busy
597         printf("kouse'isima'sita");
598         putc(0x0D);
599         delay_ms(2000);
600         while(input(PIN_C4)==0) {} //wait during busy
601     }
602 }
603
604 }
605 else if(input(PIN_A0) == 0) // mode 3, x10 volt mode
606 {
607     printf("kousei/puro'guramu/kidousima'sita");
608     putc(0x0D);
609     delay_ms(3000);
610     while(input(PIN_C4)==0) {} //wait during busy
611     if(input(PIN_A2) == 0)
612     {
613         printf("pureibo'tankara/yubi'o/hana'sitekudasai");
614         putc(0x0D);
615         delay_ms(4000);
616     }
617     while(input(PIN_C4)==0) {} //wait during busy
618     //
619     while(input(PIN_A2) == 0) {} // wait play switch
620     //
621     printf("den'a'tusokuteimo'-do/ nijyu'-bo'rutore'njidesu");
622     putc(0x0D);
623     delay_ms(4000);
624     while(input(PIN_C4)==0) {} //wait during busy
625     printf("tya'imuga/naru'maeni/ pure'ibotan/o/osu'to");
626     printf("syukka'jino/de-tani/modo'sima'su");
627     putc(0x0D);
628     delay_ms(2000);
629     while(input(PIN_C4)==0) {} //wait during busy
630     //
631     for(j=0; j<3000; j=j+1)
632     {
633         if(input(PIN_A2) == 0) // wait play switch
634         {
635             printf("#K");
636             putc(0x0D);
637             delay_ms(1000);
638             while(input(PIN_C4)==0) {} //wait during busy
639             printf("syukka'jino/ de'-tao/ sette'isima'sita");

```



```

640         putc(0x0D);
641         delay_ms(2000);
642         while(input(PIN_C4)==0) {} //wait during busy
643         volt10_coef = 1.07;
644         factory_v_data = 1;
645         WRITE_FLOAT_EEPROM(0, volt10_coef);
646         break;
647     }
648     delay_ms(1);
649 }
650 //delay_ms(3000);
651 printf("#J");
652 putc(0x0D);
653 delay_ms(1000);
654 if(factory_v_data == 0)
655 {
656     while(input(PIN_C4)==0) {} //wait during busy
657     printf("gobo'rutoo /setu'zokusi/ pu're'ibo'tan;o /osite;/kuda'sai");
658     putc(0x0D);
659     while(input(PIN_A2) == 1) {} // wait
660     if(input(PIN_A2) == 0) // wait play switch
661     {
662         readadc();
663         volt10_coef = 8003.66/value; // should be around 1.1
664         WRITE_FLOAT_EEPROM(0, volt10_coef);
665         printf("#K");
666         putc(0x0D);
667         delay_ms(1000);
668         while(input(PIN_C4)==0) {} //wait during busy
669         printf("kouse'isima'sita");
670         putc(0x0D);
671         delay_ms(2000);
672         while(input(PIN_C4)==0) {} //wait during busy
673     }
674 }
675 }
676 else // 2 volt range, it doesn't need any calibration
677 {
678     printf("den'atusokuteimo'-do/ nibo'rutore'njidesu");
679     putc(0x0D);
680     delay_ms(3000);
681     while(input(PIN_C4)==0) {} //wait during busy
682     printf("konore'njidewa/ kou'seiwa dekimase'n");
683     putc(0x0D);
684     delay_ms(5000);
685     while(input(PIN_C4)==0) {} //wait during busy
686     if(input(PIN_A2) == 0)
687     {
688         printf("pureibo'tankara/yubi'o/hana'sitekudasai");
689         putc(0x0D);
690         delay_ms(2000);
691         while(input(PIN_A2) == 0) {} // stay here
692     }
693 }
694 }
695 }
696 //
697
698

```