```c
#include "all03.h"

/*
 * MPU Atmel AT89S51/52: parallel mode
 * based on AT89S8253 tool
 *
 * Changelog
 * - 2019.09.21: 1.00 initial release
  *
 * TODO:
 * - add AT89S52
 * - test lock bits functions
 *
 */

#include <bios.h>

/* PARALLEL MODE */

#define _VERSION_       "1.00"

#define MODE_ERASE      0x05
#define MODE_WRITE      0x1E
#define MODE_READ       0x18 // 1C for AT89S8253 , 18 for AT89S51
#define MODE_LOCK_1     0x1F
#define MODE_LOCK_2     0x07
#define MODE_LOCK_3     0x0D
#define MODE_LOCK_RD    0x0B
#define MODE_SIGNATURE  0x00 // 17 for AT89S8253 , 00 for AT89S51

unsigned char _RST, _RDY, _PGM;
unsigned char _VCC, _VPP, _GND, _MODE[5], _DATA[8], _ADDR[16], _SEPARATE;

long _CODESIZE, _DATASIZE;
int _PP[5];

struct DEV {
    char *name;
    long Size, DataSize;        // ROM & EEPROM sizes in bytes
    char separate;              // 1:separated code and data space, 0:combined, data follows code
    int BlkSize, DataBlkSize;   // ROM & EEPROM pagesizes in bytes
    int progpulse[5];           // prog pulse length in us for 0:erase, 1:code, 2:data, 3:lock 4:fuses
    char rst,rdy,pgm;           // Pin numbers for RESET,RDY/BUSY and PROG
    char vcc,vpp,gnd,mode[5];   // Pin numbers for VCC, GND and MODE0..4
    char data[8];               // Pin numbers for d0..7
    char addr[16];              // Pin numbers for a0..15
    unsigned char UBmin, UBmax; // min. and max. Vcc value
    unsigned char Upp;          // Vpp value
};

struct DEV ATMEL_devs[] = { //        C/D  prog   erase  code  data  lock  fuse          ale     ea
//          TYPE       ROMSIZE EESIZE sep blksize *----- progpulse[0-4] ------*  rst rdy pgm vcc vpp gnd * mode[0-4] -*  *----- data[0-7] -----+  +-----------------  addr[0-15] -----------------+  Umn Umx Upp
/* 0*/  { "AT89S51",   0x1000, 0x0000, 1,  1,  1,     1,    1,    1,    1,    1,  9, 10, 30, 40, 31, 20, 27,28,13,16,17, 39,38,37,36,35,34,33,32,  1, 2, 3, 4, 5, 6, 7, 8, 21,22,23,24, 0, 0, 0, 0, 40, 55, 120 }
// Note: EESIZE should be set to 0 in this version for AT89S51
};

struct {
    char name[20];
    int numdevs;
    struct DEV *dev;
} mfr[] = {
    { "ATMEL", sizeof(ATMEL_devs)/sizeof(struct DEV), ATMEL_devs }
};

int mfrno = 0, devno = 0;
```

```
66  long signature;
67  int lock_bits = 0, fuse_bits = 0;
68
69  #define BUFSIZE     0x8000U
70
71  long addr;
72  long DevStart= 0x0000;
73  long Counter = 0x0000;
74
75  void ShowCounter( long val )
76  {
77      struct text_info ti;
78
79      gettextinfo( &ti );
80
81      textattr( ( BLUE << 4 ) + WHITE );
82      locate( 5, 73 ); cprintf( "%04lX", val );
83
84      textattr( ti.attribute );
85      gotoxy( ti.curx, ti.cury );
86  }
87
88  void ShowConfig( void )
89  {
90      struct text_info ti;
91      int idx;
92
93      gettextinfo( &ti );
94
95      textattr( ( BLUE << 4 ) + WHITE );
96      locate( 8,61 );
97      cprintf( "%06.6lX", signature );
98
99      locate( 9,61 );
100     //cprintf( "%01X", lock_bits );
101
102
103     for (idx = 128; idx > 0; idx = idx/2)
104     {
105         cprintf( "%c", (lock_bits & (idx)) ? '1': '0' );
106     }
107
108
109
110     //EEPROM not present
111         //locate( 2, 69 ); cprintf( "%04X", _DATASIZE );
112
113     textattr( ti.attribute );
114     gotoxy( ti.curx, ti.cury );
115 }
116
117 void ShowType( void )
118 {
119     int i;
120
121     if( mfrno < 0 || mfrno > sizeof(mfr) / sizeof(mfr[0]) )
122         mfrno = 0;
123     if( devno < 0 || devno >= mfr[mfrno].numdevs )
124         devno = 0;
125
126     _RST = mfr[mfrno].dev[devno].rst;
127     _RDY = mfr[mfrno].dev[devno].rdy;
128     _PGM = mfr[mfrno].dev[devno].pgm;
129
130     _VCC = mfr[mfrno].dev[devno].vcc;
131     _VPP = mfr[mfrno].dev[devno].vpp;
132     _GND = mfr[mfrno].dev[devno].gnd;
133
134     _CODESIZE = mfr[mfrno].dev[devno].Size;
135     _DATASIZE = mfr[mfrno].dev[devno].DataSize;
136     _SEPARATE = mfr[mfrno].dev[devno].separate;
137
```

```
138         for( i = 0; i < sizeof(_MODE); ++i )
139             _MODE[i]  = mfr[mfrno].dev[devno].mode[i];
140
141         for( i = 0; i < sizeof(_DATA); ++i )
142             _DATA[i]  = mfr[mfrno].dev[devno].data[i];
143
144         for( i = 0; i < sizeof(_ADDR); ++i )
145             _ADDR[i]  = mfr[mfrno].dev[devno].addr[i];
146
147         for( i = 0; i < sizeof(_PP)/2; ++i )
148             _PP[i]  = mfr[mfrno].dev[devno].progpulse[i];
149
150         bufsize = _CODESIZE + _DATASIZE;
151         BufEnd = bufsize - 1;
152
153         textattr( ( BLUE << 4 ) | WHITE );
154
155         locate( 0,40 ); cprintf( "*Mfr.: %s", mfr[mfrno].name );
156         locate( 0,60 ); cprintf( "*TYPE: %s", mfr[mfrno].dev[devno].name );
157
158         locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
159
160         textattr( ( CYAN << 4 ) | WHITE );
161     }
162
163     int getdata( void )
164     {
165         int i, val = 0;
166
167         for( i = sizeof(_DATA)-1; i >= 0; --i )
168             val = ( val << 1 ) | getpin( _DATA[i] );
169
170         return val;
171     }
172
173     void setdata( int val )
174     {
175         int i;
176
177         for( i = 0; i < sizeof(_DATA); ++i )
178         {
179             setpin( _DATA[i], TTLID, val & 1 );
180             val >>= 1;
181         }
182     }
183
184     void setaddr( void )
185     {
186         long a = addr;
187         int i;
188
189         if( _SEPARATE )
190             a %= _CODESIZE;
191
192         for( i = 0; i < sizeof(_ADDR); ++i )
193         {
194             setpin( _ADDR[i], TTLID, (int)(a & 1) );
195             a >>= 1;
196         }
197     }
198
199     void setmode( int md )
200     {
201         int i;
202
203         for( i = 0; i < sizeof(_MODE); ++i )
204         {
205             setpin( _MODE[i], TTLID, (int)(md & 1) );
206             md >>= 1;
207         }
208     }
209
```

```
210  void pulse( int pulselength )
211  {
212      us_delay(2);
213
214      setpin( _PGM, TTLID, 0 );
215      us_delay( pulselength );
216      setpin( _PGM, TTLID, 1 );
217
218      us_delay(2);
219  }
220
221  int wait_busy( int maxwait )
222  {
223      int t;
224
225      maxwait *= 10;
226
227      for( t = 0; t < maxwait; ++t )
228      {
229          if( getpin( _RDY ) )
230              break;
231          dly100u();
232      }
233      return getpin( _RDY ) ? 1 : 0;
234  }
235
236  void power( int voltage )
237  {
238      int i;
239
240      // we dont need VHH
241      setdac( VHHID, 0 );                                          // VHH = 0V
242      for( i = 0; i <= 4; ++i ) setport( VHHENID, i, 0 );          // no VHH
243      setport( VHHENCID,0, 0 );                                    // no VHHC
244      setport( VHHENCID,1, 0 );                                    // no VHHC
245
246      // Can't use Pins 2,3,4,6,8,32..35,37..40 for Vop
247      if( _VPP == 2 || _VPP == 3 || _VPP == 4 || _VPP == 6 || _VPP == 8 || ( _VPP > 31
         && _VPP != 36 ) )
248      {
249          errbeep();
250          textattr( ( RED << 4 ) | WHITE );
251          locate( 41, 23 ); cprintf( "Connect pin 1 to %d", _VPP );
252          textattr( ( CYAN << 4 ) | WHITE );
253          _VPP = 1;                                                // force using Pin 1
254      }
255
256      setport( OTHERENID, 0, 0 );                                  // Pin 20 = GND
257
258      if( voltage )
259      {
260          // Set all pins LOW
261          for( i = 0; i < 5; ++i )
262              setport( TTLID, i, 0 );
263
264          setdac( VCCID, voltage );                    // Vcc = xV
265          setdac( VOPID, mfr[mfrno].dev[devno].Upp ); // Vpp = xV
266          delay( 500 );                               // wait to stabilize
267
268          setpin( _VCC, VCCENID, 1 );                 // set Vcc pin (automatically sets
             TTLID also)
269
270          dly20u();
271
272          setpin( _RST, TTLID, 1 );                   // RST = H
273
274          dly20u();
275
276          setpin( 18, TTLID, 1 );                     // start oscillator on pins 18/19
277          setpin( 19, TTLID, 1 );
278          setport( OTHERENID, 0, 0x30 );              // with 4MHz
279
```

```
280             dly50m();
281
282             setpin( _VPP, TTLID, 1 );               // set EA = H
283             setpin( _PGM, TTLID, 1 );               // set PGM = H
284
285             setdata( 0xFF );                        // set D0..7 = H
286             setpin( _RDY, TTLID, 1 );               // set RDY = H (HiZ)
287
288             dly20u();
289
290             setpin( _VPP, VOPENID, 1 );             // set EA = Vpp
291
292             dly20u();
293         }
294         else
295         {
296             setpin( _VPP, VOPENID, 0 );             // remove Vpp
297             setpin( _VPP, TTLID, 0 );
298
299             dly20u();
300
301             setport( OTHERENID, 0, 0 );             // osc off
302
303             dly20u();
304
305             setpin( _RST, TTLID, 0 );               // pull reset low
306
307             dly20u();
308
309             // Set all pins LOW except VCC
310             for( i = 1; i < 40; ++i )
311                 if( i != _VCC )
312                     setpin( i, TTLID, 0 );
313
314             dly20u();
315
316             setpin( _VCC, VCCENID, 0 );             // remove Vcc
317             setpin( _VCC, TTLID, 0 );
318
319             setdac( VOPID, 0 );                     // Vpp = 0V
320             setdac( VCCID, 0 );                     // Vcc = 0V
321         }
322     }
323
324     int program( void )
325     {
326         long end = _CODESIZE + _DATASIZE;
327         int i, blksize;
328
329         setmode( MODE_WRITE );
330         blksize = mfr[mfrno].dev[devno].BlkSize;
331         for( addr = BufStart; addr < end; )
332         {
333             /*
334             if( _SEPARATE && addr == _CODESIZE )        // change area
335             {
336                 setmode( MODE_WRITE_EE );
337                 blksize = mfr[mfrno].dev[devno].DataBlkSize;
338             }
339             */
340
341             if( ( addr & 0xFF ) == 0 )
342                 ShowCounter( addr );
343
344             disable();
345             for( i = 0; i < blksize; ++i )
346             {
347                 setaddr();
348                 setdata( buffer[addr] );
349                 pulse( _PP[ (addr > _CODESIZE) ? 2 : 1 ] );
350                 ++addr;
351             }
```

```
352             enable();
353
354             dly1m();                                    // chip should have started
                programming after 1ms
355             if( !wait_busy( 10*blksize ) )              // wait for completion (max. 10ms
                per byte)
356                 return 0;
357         }
358
359         ShowCounter( addr );
360         return 1;
361     }
362
363     int read_verify_check( int md )
364     {
365         long end = _CODESIZE + _DATASIZE;
366         int val;
367
368         setdata( 0xFF );        // release pin drivers on data pins
369
370         setmode( MODE_READ );
371         for( addr = BufStart; addr < end; ++addr )
372         {
373             /*
374             if( _SEPARATE && addr == _CODESIZE )    // change area
375                 setmode( MODE_READ_EE );
376             */
377
378             setaddr();
379             if( ( addr & 0xFF ) == 0 ) ShowCounter( addr );
380
381             val = getdata();
382
383             switch( md )
384             {
385             case 0:     // read
386                 Chks += ( buffer[addr] = val );
387                 break;
388
389             case 1:     // verify
390                 if( buffer[addr] != val )
391                 {
392                     ShowCounter( addr );
393                     return 0;
394                 }
395                 break;
396
397             case 2:     // blank check
398                 if( val != 0xFF )
399                 {
400                     ShowCounter( addr );
401                     return 0;
402                 }
403                 break;
404             }
405         }
406
407         ShowCounter( addr );
408         return 1;
409     }
410
411     int write_config( void )
412     {
413             if( lock_bits & 0x01 ) {
414             setmode( MODE_LOCK_1 );
415             pulse( _PP[3] );
416             if( !wait_busy(50) )
417                 return 0;
418         }
419
420             if( lock_bits & 0x02 ) {
421             setmode( MODE_LOCK_2 );
```

```
422             pulse( _PP[3] );
423             if( !wait_busy(50) )
424                 return 0;
425         }
426
427
428             if( lock_bits & 0x04 ) {
429             setmode( MODE_LOCK_3 );
430             pulse( _PP[3] );
431             if( !wait_busy(50) )
432                 return 0;
433         }
434
435
436         /*
437
438         setdata( fuse_bits & 0x0F );
439         setmode( MODE_FUSE );
440         pulse( _PP[4] );
441         return wait_busy(50);
442         */
443         return 1;
444
445     }
446
447     int read_config( void )
448     {
449         long signature_temp;
450
451         setdata( 0xFF );
452
453
454         setmode( MODE_LOCK_RD );
455         lock_bits = (getdata() >> 2) & 7;
456     /*
457         setmode( MODE_FUSE_RD );
458         fuse_bits = getdata() & 0x0F;
459     */
460         setmode( MODE_SIGNATURE );
461         addr = 0x0200L;
462         setaddr();
463         signature = getdata()&(0xFF);
464         addr = 0x0100L;
465         setaddr();
466         signature += (getdata() << 8) & 0xFF00;
467         addr = 0x0000L;
468         setaddr();
469         signature_temp = getdata();
470         signature += ( signature_temp << 16) & 0xFF0000L;
471
472
473
474         return 1;
475     }
476
477
478     int flash_check( void )
479     {
480         int done;
481
482         textattr( ( CYAN << 4 ) | WHITE );
483         _window( 12,40, 23,79 );
484
485         textattr( ( BLUE << 4 ) | WHITE );
486         locate( 12, 45 ); cprintf( "   BLANK CHECK device:" );
487
488         for(;;)
489         {
490             textattr( ( CYAN << 4 ) | WHITE );
491             locate( 13, 41 ); cprintf( "Ready to check (Y/<CR>)? " );
492
493             for( done = 0; !done; )
```

```
494             {
495                 switch( getch() )
496                 {
497                 case 0:
498                     getch();
499                     break;
500 
501                 case '\n':
502                 case '\r':
503                 case 0x1B:
504                     return;
505 
506                 case 'y':
507                 case 'Y':
508                     done = 1;
509                 }
510             }
511 
512             clscrn( 14,41, 22,78 );
513 
514             setport( USERBITS, 0, 0 );
515 
516             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
517             locate( 14, 41 ); cprintf( "Blank checking now... " );
518 
519             power(50); done = read_verify_check(2); power(0);
520 
521             textattr( ( CYAN << 4 ) | WHITE );
522             locate( 14, 41 ); cprintf( "Blank checking now... " );
523 
524             locate( 15, 41 );
525             if( done )
526             {
527                 ShowCounter( BufEnd );
528                 putchar( 7 );
529                 cprintf( " OK !" );
530                 setport( USERBITS, 0, 8 );
531             }
532             else
533             {
534                 errbeep();
535                 textattr( ( RED << 4 ) | WHITE );
536                 cprintf( "Blank check error at %04lX", addr );
537                 textattr( ( CYAN << 4 ) | WHITE );
538             }
539         }
540     }
541 
542     void flash_program( void )
543     {
544         int done;
545 
546         textattr( ( CYAN << 4 ) | WHITE );
547         _window( 12,40, 23,79 );
548 
549         textattr( ( BLUE << 4 ) | WHITE );
550         locate( 12, 45 ); cprintf( "   PROGRAM :" );
551 
552         for(;;)
553         {
554             textattr( ( CYAN << 4 ) | WHITE );
555             locate( 13, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
556 
557             for( done = 0; !done; )
558             {
559                 switch( getch() )
560                 {
561                 case 0:
562                     getch();
563                     break;
564 
565                 case '\n':
```

```
566             case '\r':
567             case 0x1B:
568                 return;
569
570             case 'y':
571             case 'Y':
572                 done = 1;
573             }
574         }
575
576         clscrn( 14,41, 22,78 );
577
578         setport( USERBITS, 0, 0 );
579
580         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
581         locate( 14, 41 ); cprintf( "Programming now... " );
582
583         power(50); done = program(); power(0);
584
585         textattr( ( CYAN << 4 ) | WHITE );
586         locate( 14, 41 ); cprintf( "Programming now... " );
587
588         locate( 15, 41 );
589         if( done )
590         {
591             ShowCounter( BufEnd );
592             putchar( 7 );
593             setport( USERBITS, 0, 8 );
594             cprintf( " OK !" );
595         }
596         else
597         {
598             errbeep();
599             textattr( ( RED << 4 ) | WHITE );
600             cprintf( "Program error ! at %04lX", addr );
601             textattr( ( CYAN << 4 ) | WHITE );
602         }
603     }
604 }
605
606 void flash_protect( void )
607 {
608     int done;
609
610     textattr( ( CYAN << 4 ) | WHITE );
611     _window( 12,40, 23,79 );
612
613     textattr( ( BLUE << 4 ) | WHITE );
614     locate( 12, 42 ); cprintf( " Program protect bits: " );
615
616     for(;;)
617     {
618         textattr( ( CYAN << 4 ) | WHITE );
619         locate( 13, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
620
621         for( done = 0; !done; )
622         {
623             switch( getch() )
624             {
625             case 0:
626                 getch();
627                 break;
628
629             case '\n':
630             case '\r':
631             case 0x1B:
632                 return;
633
634             case 'y':
635             case 'Y':
636                 done = 1;
637             }
```

```
638             }
639
640             clscrn( 14,41, 22,78 );
641
642             setport( USERBITS, 0, 0 );
643
644             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
645             locate( 14, 41 ); cprintf( "Programming now... " );
646
647             power(50); done = write_config(); power(0);
648
649             textattr( ( CYAN << 4 ) | WHITE );
650             locate( 14, 41 ); cprintf( "Programming now... " );
651
652             locate( 15, 41 );
653             if( done )
654             {
655                 ShowCounter( BufEnd );
656                 putchar( 7 );
657                 setport( USERBITS, 0, 8 );
658                 cprintf( " OK !" );
659             }
660             else
661             {
662                 errbeep();
663                 textattr( ( RED << 4 ) | WHITE );
664                 cprintf( "Program error ! at %04lX", addr );
665                 textattr( ( CYAN << 4 ) | WHITE );
666             }
667         }
668     }
669
670     void flash_read( void )
671     {
672         int done;
673
674         textattr( ( CYAN << 4 ) | WHITE );
675         _window( 12,40, 23,79 );
676
677         textattr( ( BLUE << 4 ) | WHITE );
678         locate( 12, 45 ); cprintf( "   READ to buffer :" );
679
680         for(;;)
681         {
682             textattr( ( CYAN << 4 ) | WHITE );
683             locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
684
685             for( done = 0; !done; )
686             {
687                 switch( getch() )
688                 {
689                 case 0:
690                     getch();
691                     break;
692
693                 case '\n':
694                 case '\r':
695                 case 0x1B:
696                     return;
697
698                 case 'y':
699                 case 'Y':
700                     done = 1;
701
702                 case 'e':
703                 case 'E':
704                 case 'o':
705                 case 'O':
706                     done = 1;
707                 }
708             }
709
```

```
710             clscrn( 14,41, 22,78 );
711
712             setport( USERBITS, 0, 0 );
713
714             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
715             locate( 14, 41 ); cprintf( "Reading now... " );
716
717             Chks = 0;
718             power(50);
719             read_verify_check(0);
720             read_config();
721             power(0);
722
723             textattr( ( CYAN << 4 ) | WHITE );
724             locate( 14, 41 ); cprintf( "Reading now... " );
725
726             locate( 15, 41 );
727             putchar( 7 );
728             cprintf( " OK !" );
729
730             textattr( ( BLUE << 4 ) | WHITE );
731             locate( 4, 41 ); cprintf( "       Check Sum  : %04X", Chks );
732
733             ShowConfig();
734         }
735     }
736
737     void flash_verify( void )
738     {
739         int done;
740
741         textattr( ( CYAN << 4 ) | WHITE );
742         _window( 12,40, 23,79 );
743
744         textattr( ( BLUE << 4 ) | WHITE );
745         locate( 12, 45 ); cprintf( "   VERIFY with buffer :" );
746
747         for(;;)
748         {
749             textattr( ( CYAN << 4 ) | WHITE );
750             locate( 13, 41 ); cprintf( "Ready to verify (Y/Even/Odd/<CR>)? " );
751
752             for( done = 0; !done; )
753             {
754                 switch( getch() )
755                 {
756                 case 0:
757                     getch();
758                     break;
759
760                 case '\n':
761                 case '\r':
762                 case 0x1B:
763                     return;
764
765                 case 'y':
766                 case 'Y':
767                     done = 1;
768
769                 case 'e':
770                 case 'E':
771                 case 'o':
772                 case 'O':
773                     done = 1;
774                 }
775             }
776
777             clscrn( 14,41, 22,78 );
778
779             setport( USERBITS, 0, 0 );
780
781             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
```

```
782             locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
783
784             power(mfr[mfrno].dev[devno].UBmin);
785             done = read_verify_check(1);
786             power(0);
787
788             textattr( ( CYAN << 4 ) | WHITE );
789             locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
790
791             locate( 15, 41 );
792             if( done )
793             {
794                 ShowCounter( BufEnd );
795                 putchar( 7 );
796                 cprintf( " OK !" );
797                 setport( USERBITS, 0, 8 );
798             }
799             else
800             {
801                 errbeep();
802                 textattr( ( RED << 4 ) | WHITE );
803                 cprintf( " VERIFY ERROR ! at %04lX", addr );
804                 textattr( ( CYAN << 4 ) | WHITE );
805                 continue;
806             }
807
808             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
809             locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
810
811             power(mfr[mfrno].dev[devno].UBmax);
812             done = read_verify_check(1);
813             power(0);
814
815             textattr( ( CYAN << 4 ) | WHITE );
816             locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
817
818             locate( 17, 41 );
819             if( done )
820             {
821                 ShowCounter( BufEnd );
822                 putchar( 7 );
823                 cprintf( " OK !" );
824                 setport( USERBITS, 0, 8 );
825             }
826             else
827             {
828                 errbeep();
829                 textattr( ( RED << 4 ) | WHITE );
830                 cprintf( " VERIFY ERROR ! at %04lX", addr );
831                 textattr( ( CYAN << 4 ) | WHITE );
832             }
833         }
834     }
835
836     void flash_erase( void )
837     {
838         int done;
839
840         textattr( ( CYAN << 4 ) | WHITE );
841         _window( 12,40, 23,79 );
842
843         textattr( ( BLUE << 4 ) | WHITE );
844         locate( 12, 45 ); cprintf( "  EEPROM Erase:" );
845
846         for(;;)
847         {
848             textattr( ( CYAN << 4 ) | WHITE );
849             locate( 13, 41 ); cprintf( "Ready to erase (Y/<CR>)? " );
850
851             for( done = 0; !done; )
852             {
853                 switch( getch() )
```

```
854                 {
855                 case 0:
856                     getch();
857                     break;
858 
859                 case '\n':
860                 case '\r':
861                 case 0x1B:
862                     return;
863 
864                 case 'y':
865                 case 'Y':
866                     done = 1;
867                 }
868             }
869 
870             clscrn( 14,41, 22,78 );
871 
872             setport( USERBITS, 0, 0 );
873 
874             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
875             locate( 14, 41 ); cprintf( "Erase now... " );
876 
877             power(50);
878             setmode( MODE_ERASE );
879             pulse( _PP[0] );
880             done = wait_busy(500);
881             power(0);
882 
883             textattr( ( CYAN << 4 ) | WHITE );
884             locate( 14, 41 ); cprintf( "Erase now... " );
885 
886             locate( 15, 41 );
887             if( done )
888             {
889                 putchar( 7 );
890                 cprintf( " OK !" );
891                 setport( USERBITS, 0, 8 );
892             }
893             else
894             {
895                 errbeep();
896                 textattr( ( RED << 4 ) | WHITE );
897                 cprintf( " ERROR ! " );
898                 textattr( ( CYAN << 4 ) | WHITE );
899 
900                 locate( 21, 41 ); cprintf( "press any key to continue" );
901                 if( getch() == 0 ) getch();
902             }
903 
904             clscrn( 13,41, 22,78 );
905         }
906     }
907 
908     void flash_auto( void )
909     {
910         int done, l;
911 
912         textattr( ( CYAN << 4 ) | WHITE );
913         _window( 12,40, 23,79 );
914 
915         textattr( ( BLUE << 4 ) | WHITE );
916         locate( 12, 45 ); cprintf( "   AUTO :" );
917 
918         for(;;)
919         {
920             textattr( ( CYAN << 4 ) | WHITE );
921             locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
922 
923             for( done = 0; !done; )
924             {
925                 switch( getch() )
```

```
926             {
927             case 0:
928                 getch();
929                 break;
930 
931             case '\n':
932             case '\r':
933             case 0x1B:
934                 return;
935 
936             case 'y':
937             case 'Y':
938                 done = 1;
939 
940             case 'e':
941             case 'E':
942             case 'o':
943             case 'O':
944                 done = 1;
945             }
946         }
947 
948         clscrn( 14,41, 22,78 );
949 
950         setport( USERBITS, 0, 0 );
951 
952         l = 14;
953 
954         // BLANK CHECK
955 
956         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
957         locate( l, 41 ); cprintf( "Blank checking now... " );
958 
959         power(50); done = read_verify_check(2); power(0);
960 
961         textattr( ( CYAN << 4 ) | WHITE );
962         locate( l++, 41 ); cprintf( "Blank checking now... " );
963 
964         locate( l++, 41 );
965 
966         if( done )
967         {
968             ShowCounter( BufEnd );
969             cprintf( " OK !" );
970         }
971         else
972         {
973             errbeep();
974             textattr( ( RED << 4 ) | WHITE );
975             cprintf( "Blank check error at %04lX", addr );
976             textattr( ( CYAN << 4 ) | WHITE );
977 
978         // ERASE
979 
980             l -= 2;
981 
982             clscrn( l, 41, l+1, 78 );
983 
984             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
985             locate( l, 41 ); cprintf( "Erase now... " );
986 
987             power(50);
988             setmode( MODE_ERASE );
989             pulse( _PP[0] );
990             done = wait_busy(500);
991             power(0);
992 
993             textattr( ( CYAN << 4 ) | WHITE );
994             locate( l++, 41 ); cprintf( "Erase now... " );
995 
996             locate( l++, 41 );
997 
```

```
998             if( done )
999                 cprintf( " OK !" );
1000            else
1001            {
1002                errbeep();
1003                textattr( ( RED << 4 ) | WHITE );
1004                cprintf( " ERROR" );
1005                textattr( ( CYAN << 4 ) | WHITE );
1006
1007                continue;
1008            }
1009        }
1010
1011        // PROGRAM
1012
1013        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1014        locate( l, 41 ); cprintf( "Programming now... " );
1015
1016        power(50); done = program(); power(0);
1017
1018        textattr( ( CYAN << 4 ) | WHITE );
1019        locate( l++, 41 ); cprintf( "Programming now... " );
1020
1021        locate( l++, 41 );
1022        if( done )
1023        {
1024            ShowCounter( BufEnd );
1025            cprintf( " OK !" );
1026        }
1027        else
1028        {
1029            errbeep();
1030            textattr( ( RED << 4 ) | WHITE );
1031            cprintf( "Program error ! at %04lX", addr );
1032            textattr( ( CYAN << 4 ) | WHITE );
1033
1034            continue;
1035        }
1036
1037        // VERIFY
1038
1039        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1040        locate( l, 41 ); cprintf( "VDD max verifying now..." );
1041
1042        power(mfr[mfrno].dev[devno].UBmax);
1043        done = read_verify_check(1);
1044        power(0);
1045
1046        textattr( ( CYAN << 4 ) | WHITE );
1047        locate( l++, 41 ); cprintf( "VDD max verifying now..." );
1048
1049        locate( l++, 41 );
1050        if( done )
1051        {
1052            ShowCounter( BufEnd );
1053            putchar( 7 );
1054            cprintf( " OK !" );
1055            setport( USERBITS, 0, 8 );
1056        }
1057        else
1058        {
1059            errbeep();
1060            textattr( ( RED << 4 ) | WHITE );
1061            cprintf( " VERIFY ERROR ! at %04lX", addr );
1062            textattr( ( CYAN << 4 ) | WHITE );
1063            continue;
1064        }
1065
1066        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1067        locate( l, 41 ); cprintf( "VDD min verifying now..." );
1068
1069        power(mfr[mfrno].dev[devno].UBmin);
```

```
1070            done = read_verify_check(1);
1071            power(0);
1072
1073            textattr( ( CYAN << 4 ) | WHITE );
1074            locate( l++, 41 ); cprintf( "VDD min verifying now..." );
1075
1076            locate( l, 41 );
1077            if( done )
1078            {
1079                ShowCounter( BufEnd );
1080                cprintf( " OK !" );
1081                setport( USERBITS, 0, 8 );
1082                putchar( 7 );
1083            }
1084            else
1085            {
1086                textattr( ( RED << 4 ) | WHITE );
1087                cprintf( " VERIFY ERROR ! at %04lX", addr );
1088                textattr( ( CYAN << 4 ) | WHITE );
1089                errbeep();
1090            }
1091        }
1092    }
1093
1094    void edit_config( void )
1095    {
1096        int i, done;
1097
1098        textattr( ( CYAN << 4 ) | WHITE );
1099        _window(  1,0, 10,39 );
1100        _window( 11,1, 24,78 );
1101
1102        locate(  3, 1 ); cprintf( "Protection: " );
1103
1104        /*
1105        locate(  5, 1 ); cprintf( "Serial Pgm: " );
1106        locate(  6, 1 ); cprintf( "x2 clock  : " );
1107        locate(  7, 1 ); cprintf( "UsrRow Pgm: " );
1108        locate(  8, 1 ); cprintf( "CrystalClk: " );
1109        */
1110        locate( 14, 4 ); cprintf( "A : no protection" );
1111        locate( 14,43 ); cprintf( "B : MOVC protection" );
1112        locate( 15, 4 ); cprintf( "C : VERIFY protection" );
1113        locate( 15,43 ); cprintf( "D : EXT_EXEC protection" );
1114
1115        /*
1116        locate( 17, 4 ); cprintf( "E : Serial Pgm Enable toggle" );
1117        locate( 18, 4 ); cprintf( "F : x2 Clock Enable toggle" );
1118        locate( 19, 4 ); cprintf( "G : UsrRow Pgm Enable toggle" );
1119        locate( 20, 4 ); cprintf( "H : Crystal Clock Enable toggle" );
1120        */
1121
1122        locate( 23, 4 ); cprintf( "Select options or <CR><ESC> to go back to the main
            menu ?" );
1123
1124        textattr( ( BLUE << 4 ) | WHITE );
1125        locate(  1, 5 ); cprintf( " Configuration Bit Setting :" );
1126        locate( 11,28 ); cprintf( " Configuration Options :" );
1127
1128        for(;;)
1129        {
1130            textattr( ( BLUE << 4 ) | WHITE );
1131
1132            locate( 3, 13 );
1133            switch( lock_bits )
1134            {
1135                    default : cprintf( "none" ); lock_bits = 0; break;
1136                    case 1  : cprintf( "MOVC" ); break;
1137                    case 3  : cprintf( "MOVC & VERIFY" ); break;
1138                    case 7  : cprintf( "MOVC & VERIFY & EXT_EXEC" ); break;
1139            }
1140                    /*
```

```
1141            for( i = 0; i < 4; ++i )
1142            {
1143                locate( 5 + i, 13 );
1144                cprintf( ( fuse_bits & ( 1 << i ) ) ? "disable" : "enable" );
1145            }
1146                    */
1147            for( done = 0; !done; )
1148            {
1149                done = 1;
1150
1151                switch( toupper( getch() ) )
1152                {
1153                case 0: getch(); done = 0; break;
1154
1155                case '\n':
1156                case '\r':
1157                case 0x1B: return;
1158                case 'A' : lock_bits = 7; break;
1159                case 'B' : lock_bits = 6; break;
1160                case 'C' : lock_bits = 4; break;
1161                case 'D' : lock_bits = 0; break;
1162                /*
1163                case 'E' : fuse_bits ^= 1; break;
1164                case 'F' : fuse_bits ^= 2; break;
1165                case 'G' : fuse_bits ^= 4; break;
1166                case 'H' : fuse_bits ^= 8; break;
1167                */
1168                default  : done = 0;
1169                }
1170            }
1171        }
1172    }
1173
1174    void type_select( void )
1175    {
1176        int done, i, num, len=15, left = 40;
1177        char no[10];
1178
1179        if( ( num = mfr[mfrno].numdevs ) > 14 )
1180        {
1181            left = 0;
1182            for( i = len = 0; i < num; ++i )
1183                if( ( done = strlen( mfr[mfrno].dev[i].name ) ) > len )
1184                    len = done;
1185        }
1186
1187        textattr( ( CYAN << 4 ) | WHITE );
1188        _window( 12,left, 23,79 );
1189
1190        textattr( ( BLUE << 4 ) | WHITE );
1191        locate( 12, left+5 ); cprintf( "  TYPE SELECT:" );
1192
1193        textattr( ( CYAN << 4 ) | WHITE );
1194
1195        for( i = 0; i < num; ++i )
1196        {
1197            locate( 13+(i%7), left + 1 + (i/7)*(len+4) );
1198            cprintf( "%d.%s", i, mfr[mfrno].dev[i].name );
1199        }
1200
1201        locate( 21, left+1 ); cprintf( "<CR> back to main menu." );
1202        locate( 22, left+1 ); cprintf( "SELECT NUMBER ?" );
1203
1204        for(;;)
1205        {
1206            for( no[0] = done = 0; !done; )
1207            {
1208                locate( 22, left+16 ); cprintf( "%s ", no );
1209                locate( 22, left+16+strlen(no) );
1210
1211                switch( i = getch() )
1212                {
```

```
1213                case 0:
1214                    getch();
1215                    break;
1216
1217                case 8:
1218                    if( ( i = strlen(no) ) != 0 )
1219                        no[i-1] = 0;
1220                    break;
1221
1222                case '\n':
1223                case '\r':
1224                    if( strlen( no ) &&
1225                        ( i = atoi(no) ) >= 0 && i < mfr[mfrno].numdevs )
1226                    {
1227                        devno = i;
1228                        ShowType();
1229                        return;
1230                    }
1231                    break;
1232
1233                case 0x1B:
1234                    return;
1235
1236                default:
1237                    if( isdigit( i ) )
1238                        strcat( no, (char*)&i );
1239                    break;
1240                }
1241            }
1242        }
1243    }
1244
1245    void mfr_select( void )
1246    {
1247        int done, i;
1248        char no[10];
1249
1250        textattr( ( CYAN << 4 ) | WHITE );
1251        _window( 12,40, 23,79 );
1252
1253        textattr( ( BLUE << 4 ) | WHITE );
1254        locate( 12, 45 ); cprintf( "  MFR SELECT:" );
1255
1256        textattr( ( CYAN << 4 ) | WHITE );
1257
1258        for( i = 0; i < sizeof(mfr) / sizeof(mfr[0]); ++i )
1259        {
1260            locate( 13+i, 41 ); cprintf( "%d.%s", i, mfr[i].name );
1261        }
1262
1263        locate( 21, 41 ); cprintf( "<CR> back to main menu." );
1264        locate( 22, 41 ); cprintf( "SELECT NUMBER ?" );
1265
1266        for(;;)
1267        {
1268            for( no[0] = done = 0; !done; )
1269            {
1270                locate( 22, 56 ); cprintf( "%s ", no );
1271                locate( 22, 56+strlen(no) );
1272
1273                switch( i = getch() )
1274                {
1275                case 0:
1276                    getch();
1277                    break;
1278
1279                case 8:
1280                    if( ( i = strlen(no) ) != 0 )
1281                        no[i-1] = 0;
1282                    break;
1283
1284                case '\n':
```

```
1285                case '\r':
1286                    if( strlen( no ) &&
1287                        ( i = atoi(no) ) >= 0 && i < sizeof(mfr) / sizeof(mfr[0]) )
1288                    {
1289                        mfrno = i;
1290                        if( devno >= mfr[mfrno].numdevs )
1291                            devno = 0;
1292                        ShowType();
1293                        return;
1294                    }
1295                    break;
1296
1297                case 0x1B:
1298                    return;
1299
1300                default:
1301                    if( isdigit( i ) )
1302                        strcat( no, (char*)&i );
1303                    break;
1304                }
1305            }
1306        }
1307    }
1308
1309    /*=========================================================*/
1310    int main( void )
1311    {
1312        int redraw, first = 1;
1313        long tmpval;
1314
1315        /*---------------------------------------------------------*/
1316        /* main program starts here                                */
1317        /*---------------------------------------------------------*/
1318
1319        getcwd( oldpath, 260 );
1320        strcpy( path, oldpath );
1321
1322        if( ( buffer = farmalloc( BUFSIZE ) ) == NULL )
1323            return -1;
1324        memset( (void far *)buffer, 0, BUFSIZE );
1325
1326        ReadConfig();
1327        delay(0);
1328
1329        for( ;; )
1330        {
1331            if( first )
1332            {
1333                first = 0;
1334
1335                init_hw();
1336                initdacs();
1337                setport( USERBITS, 0, 0 );
1338            }
1339
1340            textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 0,0, 24,79 );
1341
1342            locate( 0,0 ); cprintf( "Universal   Programmer" );
1343            locate( 1,0 ); cprintf( "MODEL: PC Based" );
1344            locate( 2,0 ); cprintf( "MPU AT89S51/52 section " _VERSION_ );
1345
1346            textattr( ( BLUE << 4 ) + WHITE ); clscrn( 0,40, 6,79 );
1347
1348            ShowType();
1349
1350            textattr( ( BLUE << 4 ) + WHITE ); _window( 1,40, 6,79 );
1351            locate( 1,53 ); cprintf( " TARGET ZONE " );
1352            locate( 2,41 ); cprintf( "Buffer start addr.: %04lX", BufStart );
1353            locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
1354            locate( 4,41 ); cprintf( "       Check Sum  : %04X", Chks );
1355            locate( 5,41 ); cprintf( "Device start addr.: %04lX", DevStart );
1356
```

```
1357            _window( 3,69, 6,79 );
1358            locate( 4,71 ); cprintf( "COUNTER" );
1359            ShowCounter( 0 );
1360
1361            _window( 7,40, 10,79 );
1362            locate( 7,43 ); cprintf( " Device ID & Configuration bits " );
1363            locate( 8,42 ); cprintf( "Manuf./Device ID : " );
1364            locate( 9,42 ); cprintf( "Lock bits        : " );
1365            ShowConfig();
1366
1367            textattr( ( CYAN << 4 ) | WHITE ); clscrn( 3,0, 23,38 );
1368
1369            locate( 3,0 ); cprintf( "-------------  Main Menu  -------------" );
1370            locate( 4,0 ); cprintf( "1. DOS SHELL                           " );
1371            locate( 5,0 ); cprintf( "2. Load BIN or HEX file to buffer      " );
1372            locate( 6,0 ); cprintf( "3. Save buffer to disk                 " );
1373            locate( 7,0 ); cprintf( "4. Edit buffer     7. Display buffer   " );
1374            locate( 8,0 ); cprintf( "5. Change I/O base address            " );
1375            locate( 9,0 ); cprintf( "6. Display loaded file history         " );
1376            locate(10,0 ); cprintf( "W. Swap hi-low bytes in buffer         " );
1377            locate(11,0 ); cprintf( "T. Type select     Z. Target zone      " );
1378            locate(12,0 ); cprintf( "B. Blank check     D. Display          " );
1379            if( _DATASIZE != 0 ) {
1380            locate(13,0 ); cprintf( "P. Program (Program Mem & Data Mem)    " );
1381            locate(14,0 ); cprintf( "A. Auto(B&S&P&V&L)                     " );
1382            locate(15,0 ); cprintf( "S. Erase Program & Data memory         " );
1383            } else {
1384            locate(13,0 ); cprintf( "                                       " );
1385            locate(14,0 ); cprintf( "P. Program         A. Auto(B&S&P&V&L)  " );
1386            locate(15,0 ); cprintf( "S. Erase Program memory                " );
1387            }
1388            locate(16,0 ); cprintf( "R. Read            V. Verify           " );
1389            locate(17,0 ); cprintf( "                                       " );
1390                    locate(18,0 ); cprintf( "E. Configure protect bits              " );
1391                    locate(19,0 ); cprintf( "L. Program protect bits                " );
1392            locate(20,0 ); cprintf( "Q. Quit                                " );
1393            locate(21,0 ); cprintf( "---------------------------------------" );
1394            locate(22,0 ); cprintf( "Allocation Buffer size : %uK bytes", BUFSIZE/1024 );
1395            if( _DATASIZE != 0 ) {
1396            locate(23,0 ); cprintf( "Data memory buffer at %04lX ~ %04lX", _CODESIZE,
                _CODESIZE + _DATASIZE - 1 );
1397            }
1398
1399            for( redraw = 0; !redraw; )
1400            {
1401                int c;
1402
1403                textattr( ( BLUE << 4 ) + WHITE ); clscrn( 24,0, 24,38 );
1404
1405                locate( 24,0 ); cprintf( "Select function ? " );
1406
1407                for(;;)
1408                {
1409                    if( ( c = getch() ) != 0 )
1410                        break;
1411                    getch();                /* neglect extended code */
1412                }
1413
1414                switch( c = toupper(c) )
1415                {
1416                case '1': dos_shell( "" ); redraw = 1; break;
1417                case '2': tmpval = bufsize; bufsize = 0x8000;
1418                          memset( (void far *)buffer, 0, BUFSIZE );     // clear buffer
1419                          addr = 0;
1420                          load_file();
1421                          bufsize= tmpval;
1422                          redraw = 1; break;
1423                case '3': save_file(); break;
1424                case '4': tmpval = bufsize; bufsize = 0x8000;
1425                          addr = 0;
1426                          edit_buffer();
1427                          bufsize= tmpval;
```

```
1428                        redraw = 1; break;
1429                case '5': set_io_adr(); first = redraw = 1; break;
1430                case '7': disp_buffer(); redraw = 1; break;
1431
1432                case 'M': mfr_select(); redraw = first = 1; break;
1433                case 'T': type_select(); redraw = first = 1; break;
1434                case 'E': edit_config(); redraw = 1; break;
1435
1436                case 'R': flash_read(); break;
1437                case 'B': flash_check(); break;
1438                case 'S': flash_erase(); break;
1439                case 'P': flash_program(); break;
1440                case 'V': flash_verify(); break;
1441                case 'L': flash_protect(); break;
1442                case 'A': flash_auto(); break;
1443
1444    //          case '\n':
1445    //          case '\r': redraw = 1; break;       // refresh
1446                }
1447
1448                setport( USERBITS, 0, 0 );
1449
1450                if( c == 'Q' )
1451                {
1452                    WriteConfig();
1453                    textattr( LIGHTGRAY ); clrscr();
1454                    chdir( oldpath );
1455                    if( buffer ) farfree( (void far *)buffer );
1456                    return( 0 );
1457                }
1458
1459                textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 11,40, 23,79 );
1460            }
1461        }
1462    }
1463
1464
```