```c
#include "all03.h"

#include <bios.h>

/* PARALLEL MODE */

#define _VERSION_       "1.00"

#define MODE_ERASE      0x05
#define MODE_WRITE      0x1E
#define MODE_READ       0x1C
#define MODE_WRITE_EE   0x1A
#define MODE_READ_EE    0x18
#define MODE_LOCK       0x0D
#define MODE_LOCK_RD    0x07
#define MODE_USERROW    0x1D
#define MODE_USER_RD    0x14
#define MODE_SIGNATURE  0x04
#define MODE_FUSE       0x16
#define MODE_FUSE_RD    0x17

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*/  { "AT89S8253", 0x3000, 0x0800, 1, 64, 32,     2,    2,    2,    2,    2,  9, 10, 30, 40, 31, 20, 13,14,15,16,17, 39,38,37,36,35,34,33,32,  1, 2, 3, 4, 5, 6, 7, 8, 21,22,23,24,25,26, 0, 0, 40, 55, 120 }
};

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

int mfrno = 0, devno = 0;
int signature;
int lock_bits = 7, fuse_bits = 0;

#define BUFSIZE     0x8000U

long addr;
long DevStart= 0x0000;
long Counter = 0x0000;

void ShowCounter( long val )
```

```
66  {
67      struct text_info ti;
68
69      gettextinfo( &ti );
70
71      textattr( ( BLUE << 4 ) + WHITE );
72      locate( 5, 73 ); cprintf( "%04lX", val );
73
74      textattr( ti.attribute );
75      gotoxy( ti.curx, ti.cury );
76  }
77
78  void ShowConfig( void )
79  {
80      struct text_info ti;
81
82      gettextinfo( &ti );
83
84      textattr( ( BLUE << 4 ) + WHITE );
85      locate( 8,54 );
86      cprintf( "%02X", signature );
87
88      locate( 9,63 );
89      cprintf( "%02X %02X", lock_bits, fuse_bits );
90
91      locate( 2, 69 ); cprintf( "%04X", _DATASIZE );
92
93      textattr( ti.attribute );
94      gotoxy( ti.curx, ti.cury );
95  }
96
97  void ShowType( void )
98  {
99      int i;
100
101     if( mfrno < 0 || mfrno > sizeof(mfr) / sizeof(mfr[0]) )
102         mfrno = 0;
103     if( devno < 0 || devno >= mfr[mfrno].numdevs )
104         devno = 0;
105
106     _RST = mfr[mfrno].dev[devno].rst;
107     _RDY = mfr[mfrno].dev[devno].rdy;
108     _PGM = mfr[mfrno].dev[devno].pgm;
109
110     _VCC = mfr[mfrno].dev[devno].vcc;
111     _VPP = mfr[mfrno].dev[devno].vpp;
112     _GND = mfr[mfrno].dev[devno].gnd;
113
114     _CODESIZE = mfr[mfrno].dev[devno].Size;
115     _DATASIZE = mfr[mfrno].dev[devno].DataSize;
116     _SEPARATE = mfr[mfrno].dev[devno].separate;
117
118     for( i = 0; i < sizeof(_MODE); ++i )
119         _MODE[i]  = mfr[mfrno].dev[devno].mode[i];
120
121     for( i = 0; i < sizeof(_DATA); ++i )
122         _DATA[i]  = mfr[mfrno].dev[devno].data[i];
123
124     for( i = 0; i < sizeof(_ADDR); ++i )
125         _ADDR[i]  = mfr[mfrno].dev[devno].addr[i];
126
127     for( i = 0; i < sizeof(_PP)/2; ++i )
128         _PP[i]  = mfr[mfrno].dev[devno].progpulse[i];
129
130     bufsize = _CODESIZE + _DATASIZE;
131     BufEnd = bufsize - 1;
132
133     textattr( ( BLUE << 4 ) | WHITE );
134
135     locate( 0,40 ); cprintf( "*Mfr.: %s", mfr[mfrno].name );
136     locate( 0,60 ); cprintf( "*TYPE: %s", mfr[mfrno].dev[devno].name );
137
```

```
138         locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
139
140         textattr( ( CYAN << 4 ) | WHITE );
141     }
142
143     int getdata( void )
144     {
145         int i, val = 0;
146
147         for( i = sizeof(_DATA)-1; i >= 0; --i )
148             val = ( val << 1 ) | getpin( _DATA[i] );
149
150         return val;
151     }
152
153     void setdata( int val )
154     {
155         int i;
156
157         for( i = 0; i < sizeof(_DATA); ++i )
158         {
159             setpin( _DATA[i], TTLID, val & 1 );
160             val >>= 1;
161         }
162     }
163
164     void setaddr( void )
165     {
166         long a = addr;
167         int i;
168
169         if( _SEPARATE )
170             a %= _CODESIZE;
171
172         for( i = 0; i < sizeof(_ADDR); ++i )
173         {
174             setpin( _ADDR[i], TTLID, (int)(a & 1) );
175             a >>= 1;
176         }
177     }
178
179     void setmode( int md )
180     {
181         int i;
182
183         for( i = 0; i < sizeof(_MODE); ++i )
184         {
185             setpin( _MODE[i], TTLID, (int)(md & 1) );
186             md >>= 1;
187         }
188     }
189
190     void pulse( int pulselength )
191     {
192         us_delay(2);
193
194         setpin( _PGM, TTLID, 0 );
195         us_delay( pulselength );
196         setpin( _PGM, TTLID, 1 );
197
198         us_delay(2);
199     }
200
201     int wait_busy( int maxwait )
202     {
203         int t;
204
205         maxwait *= 10;
206
207         for( t = 0; t < maxwait; ++t )
208         {
209             if( getpin( _RDY ) )
```

```
210             break;
211         dly100u();
212     }
213     return getpin( _RDY ) ? 1 : 0;
214 }
215
216 void power( int voltage )
217 {
218     int i;
219
220     // we dont need VHH
221     setdac( VHHID, 0 );                                     // VHH = 0V
222     for( i = 0; i <= 4; ++i ) setport( VHHENID, i, 0 );     // no VHH
223     setport( VHHENCID,0, 0 );                               // no VHHC
224     setport( VHHENCID,1, 0 );                               // no VHHC
225
226     // Can't use Pins 2,3,4,6,8,32..35,37..40 for Vop
227     if( _VPP == 2 || _VPP == 3 || _VPP == 4 || _VPP == 6 || _VPP == 8 || ( _VPP > 31
        && _VPP != 36 ) )
228     {
229         errbeep();
230         textattr( ( RED << 4 ) | WHITE );
231         locate( 41, 23 ); cprintf( "Connect pin 1 to %d", _VPP );
232         textattr( ( CYAN << 4 ) | WHITE );
233         _VPP = 1;                                           // force using Pin 1
234     }
235
236     setport( OTHERENID, 0, 0 );                             // Pin 20 = GND
237
238     if( voltage )
239     {
240         // Set all pins LOW
241         for( i = 0; i < 5; ++i )
242             setport( TTLID, i, 0 );
243
244         setdac( VCCID, voltage );                   // Vcc = xV
245         setdac( VOPID, mfr[mfrno].dev[devno].Upp ); // Vpp = xV
246         delay( 500 );                               // wait to stabilize
247
248         setpin( _VCC, VCCENID, 1 );                 // set Vcc pin (automatically sets
            TTLID also)
249
250         dly20u();
251
252         setpin( _RST, TTLID, 1 );                   // RST = H
253
254         dly20u();
255
256         setpin( 18, TTLID, 1 );                     // start oscillator on pins 18/19
257         setpin( 19, TTLID, 1 );
258         setport( OTHERENID, 0, 0x30 );              // with 4MHz
259
260         dly50m();
261
262         setpin( _VPP, TTLID, 1 );                   // set EA = H
263         setpin( _PGM, TTLID, 1 );                   // set PGM = H
264
265         setdata( 0xFF );                            // set D0..7 = H
266         setpin( _RDY, TTLID, 1 );                   // set RDY = H (HiZ)
267
268         dly20u();
269
270         setpin( _VPP, VOPENID, 1 );                 // set EA = Vpp
271
272         dly20u();
273     }
274     else
275     {
276         setpin( _VPP, VOPENID, 0 );                 // remove Vpp
277         setpin( _VPP, TTLID, 0 );
278
279         dly20u();
```

```
280
281         setport( OTHERENID, 0, 0 );             // osc off
282
283         dly20u();
284
285         setpin( _RST, TTLID, 0 );               // pull reset low
286
287         dly20u();
288
289         // Set all pins LOW except VCC
290         for( i = 1; i < 40; ++i )
291             if( i != _VCC )
292                 setpin( i, TTLID, 0 );
293
294         dly20u();
295
296         setpin( _VCC, VCCENID, 0 );             // remove Vcc
297         setpin( _VCC, TTLID, 0 );
298
299         setdac( VOPID, 0 );                     // Vpp = 0V
300         setdac( VCCID, 0 );                     // Vcc = 0V
301     }
302 }
303
304 int program( void )
305 {
306     long end = _CODESIZE + _DATASIZE;
307     int i, blksize;
308
309     setmode( MODE_WRITE );
310     blksize = mfr[mfrno].dev[devno].BlkSize;
311     for( addr = BufStart; addr < end; )
312     {
313         if( _SEPARATE && addr == _CODESIZE )        // change area
314         {
315             setmode( MODE_WRITE_EE );
316             blksize = mfr[mfrno].dev[devno].DataBlkSize;
317         }
318
319         if( ( addr & 0xFF ) == 0 )
320             ShowCounter( addr );
321
322         disable();
323         for( i = 0; i < blksize; ++i )
324         {
325             setaddr();
326             setdata( buffer[addr] );
327             pulse( _PP[ (addr > _CODESIZE) ? 2 : 1 ] );
328             ++addr;
329         }
330         enable();
331
332         dly1m();                                // chip should have started
            programming after 1ms
333         if( !wait_busy( 10*blksize ) )          // wait for completion (max. 10ms
            per byte)
334             return 0;
335     }
336
337     ShowCounter( addr );
338     return 1;
339 }
340
341 int read_verify_check( int md )
342 {
343     long end = _CODESIZE + _DATASIZE;
344     int val;
345
346     setdata( 0xFF );        // release pin drivers on data pins
347
348     setmode( MODE_READ );
349     for( addr = BufStart; addr < end; ++addr )
```

```
350         {
351             if( _SEPARATE && addr == _CODESIZE )    // change area
352                 setmode( MODE_READ_EE );
353 
354             setaddr();
355             if( ( addr & 0xFF ) == 0 ) ShowCounter( addr );
356 
357             val = getdata();
358 
359             switch( md )
360             {
361             case 0:     // read
362                 Chks += ( buffer[addr] = val );
363                 break;
364 
365             case 1:     // verify
366                 if( buffer[addr] != val )
367                 {
368                     ShowCounter( addr );
369                     return 0;
370                 }
371                 break;
372 
373             case 2:     // blank check
374                 if( val != 0xFF )
375                 {
376                     ShowCounter( addr );
377                     return 0;
378                 }
379                 break;
380             }
381         }
382 
383         ShowCounter( addr );
384         return 1;
385     }
386 
387     int write_config( void )
388     {
389         setdata( lock_bits & 7 );
390         setmode( MODE_LOCK );
391         pulse( _PP[3] );
392         if( !wait_busy(50) )
393             return 0;
394 
395         setdata( fuse_bits & 0x0F );
396         setmode( MODE_FUSE );
397         pulse( _PP[4] );
398         return wait_busy(50);
399     }
400 
401     int read_config( void )
402     {
403         setdata( 0xFF );
404 
405         setmode( MODE_LOCK_RD );
406         lock_bits = getdata() & 7;
407 
408         setmode( MODE_FUSE_RD );
409         fuse_bits = getdata() & 0x0F;
410 
411         setmode( MODE_SIGNATURE );
412         signature = getdata();
413 
414         return 1;
415     }
416 
417 
418     int flash_check( void )
419     {
420         int done;
421 
```

```
422         textattr( ( CYAN << 4 ) | WHITE );
423         _window( 12,40, 23,79 );
424
425         textattr( ( BLUE << 4 ) | WHITE );
426         locate( 12, 45 ); cprintf( "   BLANK CHECK device:" );
427
428         for(;;)
429         {
430             textattr( ( CYAN << 4 ) | WHITE );
431             locate( 13, 41 ); cprintf( "Ready to check (Y/<CR>)? " );
432
433             for( done = 0; !done; )
434             {
435                 switch( getch() )
436                 {
437                 case 0:
438                     getch();
439                     break;
440
441                 case '\n':
442                 case '\r':
443                 case 0x1B:
444                     return;
445
446                 case 'y':
447                 case 'Y':
448                     done = 1;
449                 }
450             }
451
452             clscrn( 14,41, 22,78 );
453
454             setport( USERBITS, 0, 0 );
455
456             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
457             locate( 14, 41 ); cprintf( "Blank checking now... " );
458
459             power(50); done = read_verify_check(2); power(0);
460
461             textattr( ( CYAN << 4 ) | WHITE );
462             locate( 14, 41 ); cprintf( "Blank checking now... " );
463
464             locate( 15, 41 );
465             if( done )
466             {
467                 ShowCounter( BufEnd );
468                 putchar( 7 );
469                 cprintf( " OK !" );
470                 setport( USERBITS, 0, 8 );
471             }
472             else
473             {
474                 errbeep();
475                 textattr( ( RED << 4 ) | WHITE );
476                 cprintf( "Blank check error at %04lX", addr );
477                 textattr( ( CYAN << 4 ) | WHITE );
478             }
479         }
480     }
481
482     void flash_program( void )
483     {
484         int done;
485
486         textattr( ( CYAN << 4 ) | WHITE );
487         _window( 12,40, 23,79 );
488
489         textattr( ( BLUE << 4 ) | WHITE );
490         locate( 12, 45 ); cprintf( "   PROGRAM :" );
491
492         for(;;)
493         {
```

```
494             textattr( ( CYAN << 4 ) | WHITE );
495             locate( 13, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
496
497             for( done = 0; !done; )
498             {
499                 switch( getch() )
500                 {
501                 case 0:
502                     getch();
503                     break;
504
505                 case '\n':
506                 case '\r':
507                 case 0x1B:
508                     return;
509
510                 case 'y':
511                 case 'Y':
512                     done = 1;
513                 }
514             }
515
516             clscrn( 14,41, 22,78 );
517
518             setport( USERBITS, 0, 0 );
519
520             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
521             locate( 14, 41 ); cprintf( "Programming now... " );
522
523             power(50); done = program(); power(0);
524
525             textattr( ( CYAN << 4 ) | WHITE );
526             locate( 14, 41 ); cprintf( "Programming now... " );
527
528             locate( 15, 41 );
529             if( done )
530             {
531                 ShowCounter( BufEnd );
532                 putchar( 7 );
533                 setport( USERBITS, 0, 8 );
534                 cprintf( " OK !" );
535             }
536             else
537             {
538                 errbeep();
539                 textattr( ( RED << 4 ) | WHITE );
540                 cprintf( "Program error ! at %04lX", addr );
541                 textattr( ( CYAN << 4 ) | WHITE );
542             }
543         }
544     }
545
546     void flash_protect( void )
547     {
548         int done;
549
550         textattr( ( CYAN << 4 ) | WHITE );
551         _window( 12,40, 23,79 );
552
553         textattr( ( BLUE << 4 ) | WHITE );
554         locate( 12, 42 ); cprintf( " Program ID & CFG & protect bits: " );
555
556         for(;;)
557         {
558             textattr( ( CYAN << 4 ) | WHITE );
559             locate( 13, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
560
561             for( done = 0; !done; )
562             {
563                 switch( getch() )
564                 {
565                 case 0:
```

```
                getch();
                break;

            case '\n':
            case '\r':
            case 0x1B:
                return;

            case 'y':
            case 'Y':
                done = 1;
            }
        }

        clscrn( 14,41, 22,78 );

        setport( USERBITS, 0, 0 );

        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
        locate( 14, 41 ); cprintf( "Programming now... " );

        power(50); done = write_config(); power(0);

        textattr( ( CYAN << 4 ) | WHITE );
        locate( 14, 41 ); cprintf( "Programming now... " );

        locate( 15, 41 );
        if( done )
        {
            ShowCounter( BufEnd );
            putchar( 7 );
            setport( USERBITS, 0, 8 );
            cprintf( " OK !" );
        }
        else
        {
            errbeep();
            textattr( ( RED << 4 ) | WHITE );
            cprintf( "Program error ! at %04lX", addr );
            textattr( ( CYAN << 4 ) | WHITE );
        }
    }
}

void flash_read( void )
{
    int done;

    textattr( ( CYAN << 4 ) | WHITE );
    _window( 12,40, 23,79 );

    textattr( ( BLUE << 4 ) | WHITE );
    locate( 12, 45 ); cprintf( "   READ to buffer :" );

    for(;;)
    {
        textattr( ( CYAN << 4 ) | WHITE );
        locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );

        for( done = 0; !done; )
        {
            switch( getch() )
            {
            case 0:
                getch();
                break;

            case '\n':
            case '\r':
            case 0x1B:
                return;
```

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

```
710             case 'E':
711             case 'o':
712             case 'O':
713                 done = 1;
714             }
715         }
716 
717         clscrn( 14,41, 22,78 );
718 
719         setport( USERBITS, 0, 0 );
720 
721         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
722         locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
723 
724         power(mfr[mfrno].dev[devno].UBmin);
725         done = read_verify_check(1);
726         power(0);
727 
728         textattr( ( CYAN << 4 ) | WHITE );
729         locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
730 
731         locate( 15, 41 );
732         if( done )
733         {
734             ShowCounter( BufEnd );
735             putchar( 7 );
736             cprintf( " OK !" );
737             setport( USERBITS, 0, 8 );
738         }
739         else
740         {
741             errbeep();
742             textattr( ( RED << 4 ) | WHITE );
743             cprintf( " VERIFY ERROR ! at %04lX", addr );
744             textattr( ( CYAN << 4 ) | WHITE );
745             continue;
746         }
747 
748         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
749         locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
750 
751         power(mfr[mfrno].dev[devno].UBmax);
752         done = read_verify_check(1);
753         power(0);
754 
755         textattr( ( CYAN << 4 ) | WHITE );
756         locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
757 
758         locate( 17, 41 );
759         if( done )
760         {
761             ShowCounter( BufEnd );
762             putchar( 7 );
763             cprintf( " OK !" );
764             setport( USERBITS, 0, 8 );
765         }
766         else
767         {
768             errbeep();
769             textattr( ( RED << 4 ) | WHITE );
770             cprintf( " VERIFY ERROR ! at %04lX", addr );
771             textattr( ( CYAN << 4 ) | WHITE );
772         }
773     }
774 }
775 
776 void flash_erase( void )
777 {
778     int done;
779 
780     textattr( ( CYAN << 4 ) | WHITE );
781     _window( 12,40, 23,79 );
```

```
782
783         textattr( ( BLUE << 4 ) | WHITE );
784         locate( 12, 45 ); cprintf( "  EEPROM Erase:" );
785
786         for(;;)
787         {
788             textattr( ( CYAN << 4 ) | WHITE );
789             locate( 13, 41 ); cprintf( "Ready to erase (Y/<CR>)? " );
790
791             for( done = 0; !done; )
792             {
793                 switch( getch() )
794                 {
795                 case 0:
796                     getch();
797                     break;
798
799                 case '\n':
800                 case '\r':
801                 case 0x1B:
802                     return;
803
804                 case 'y':
805                 case 'Y':
806                     done = 1;
807                 }
808             }
809
810             clscrn( 14,41, 22,78 );
811
812             setport( USERBITS, 0, 0 );
813
814             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
815             locate( 14, 41 ); cprintf( "Erase now... " );
816
817             power(50);
818             setmode( MODE_ERASE );
819             pulse( _PP[0] );
820             done = wait_busy(50);
821             power(0);
822
823             textattr( ( CYAN << 4 ) | WHITE );
824             locate( 14, 41 ); cprintf( "Erase now... " );
825
826             locate( 15, 41 );
827             if( done )
828             {
829                 putchar( 7 );
830                 cprintf( " OK !" );
831                 setport( USERBITS, 0, 8 );
832             }
833             else
834             {
835                 errbeep();
836                 textattr( ( RED << 4 ) | WHITE );
837                 cprintf( " ERROR ! " );
838                 textattr( ( CYAN << 4 ) | WHITE );
839
840                 locate( 21, 41 ); cprintf( "press any key to continue" );
841                 if( getch() == 0 ) getch();
842             }
843
844             clscrn( 13,41, 22,78 );
845         }
846     }
847
848     void flash_auto( void )
849     {
850         int done, l;
851
852         textattr( ( CYAN << 4 ) | WHITE );
853         _window( 12,40, 23,79 );
```

```
854
855         textattr( ( BLUE << 4 ) | WHITE );
856         locate( 12, 45 ); cprintf( "   AUTO :" );
857
858         for(;;)
859         {
860             textattr( ( CYAN << 4 ) | WHITE );
861             locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
862
863             for( done = 0; !done; )
864             {
865                 switch( getch() )
866                 {
867                 case 0:
868                     getch();
869                     break;
870
871                 case '\n':
872                 case '\r':
873                 case 0x1B:
874                     return;
875
876                 case 'y':
877                 case 'Y':
878                     done = 1;
879
880                 case 'e':
881                 case 'E':
882                 case 'o':
883                 case 'O':
884                     done = 1;
885                 }
886             }
887
888             clscrn( 14,41, 22,78 );
889
890             setport( USERBITS, 0, 0 );
891
892             l = 14;
893
894             // BLANK CHECK
895
896             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
897             locate( l, 41 ); cprintf( "Blank checking now... " );
898
899             power(50); done = read_verify_check(2); power(0);
900
901             textattr( ( CYAN << 4 ) | WHITE );
902             locate( l++, 41 ); cprintf( "Blank checking now... " );
903
904             locate( l++, 41 );
905
906             if( done )
907             {
908                 ShowCounter( BufEnd );
909                 cprintf( " OK !" );
910             }
911             else
912             {
913                 errbeep();
914                 textattr( ( RED << 4 ) | WHITE );
915                 cprintf( "Blank check error at %04lX", addr );
916                 textattr( ( CYAN << 4 ) | WHITE );
917
918             // ERASE
919
920                 l -= 2;
921
922                 clscrn( l, 41, l+1, 78 );
923
924                 textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
925                 locate( l, 41 ); cprintf( "Erase now... " );
```

```
926
927                 power(50);
928                 setmode( MODE_ERASE );
929                 pulse( _PP[0] );
930                 done = wait_busy(50);
931                 power(0);
932
933                 textattr( ( CYAN << 4 ) | WHITE );
934                 locate( l++, 41 ); cprintf( "Erase now... " );
935
936                 locate( l++, 41 );
937
938                 if( done )
939                     cprintf( " OK !" );
940                 else
941                 {
942                     errbeep();
943                     textattr( ( RED << 4 ) | WHITE );
944                     cprintf( " ERROR" );
945                     textattr( ( CYAN << 4 ) | WHITE );
946
947                     continue;
948                 }
949             }
950
951             // PROGRAM
952
953             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
954             locate( l, 41 ); cprintf( "Programming now... " );
955
956             power(50); done = program(); power(0);
957
958             textattr( ( CYAN << 4 ) | WHITE );
959             locate( l++, 41 ); cprintf( "Programming now... " );
960
961             locate( l++, 41 );
962             if( done )
963             {
964                 ShowCounter( BufEnd );
965                 cprintf( " OK !" );
966             }
967             else
968             {
969                 errbeep();
970                 textattr( ( RED << 4 ) | WHITE );
971                 cprintf( "Program error ! at %04lX", addr );
972                 textattr( ( CYAN << 4 ) | WHITE );
973
974                 continue;
975             }
976
977             // VERIFY
978
979             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
980             locate( l, 41 ); cprintf( "VDD max verifying now..." );
981
982             power(mfr[mfrno].dev[devno].UBmax);
983             done = read_verify_check(1);
984             power(0);
985
986             textattr( ( CYAN << 4 ) | WHITE );
987             locate( l++, 41 ); cprintf( "VDD max verifying now..." );
988
989             locate( l++, 41 );
990             if( done )
991             {
992                 ShowCounter( BufEnd );
993                 putchar( 7 );
994                 cprintf( " OK !" );
995                 setport( USERBITS, 0, 8 );
996             }
997             else
```

```
 998          {
 999              errbeep();
1000              textattr( ( RED << 4 ) | WHITE );
1001              cprintf( " VERIFY ERROR ! at %04lX", addr );
1002              textattr( ( CYAN << 4 ) | WHITE );
1003              continue;
1004          }
1005
1006          textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1007          locate( l, 41 ); cprintf( "VDD min verifying now..." );
1008
1009          power(mfr[mfrno].dev[devno].UBmin);
1010          done = read_verify_check(1);
1011          power(0);
1012
1013          textattr( ( CYAN << 4 ) | WHITE );
1014          locate( l++, 41 ); cprintf( "VDD min verifying now..." );
1015
1016          locate( l, 41 );
1017          if( done )
1018          {
1019              ShowCounter( BufEnd );
1020              cprintf( " OK !" );
1021              setport( USERBITS, 0, 8 );
1022              putchar( 7 );
1023          }
1024          else
1025          {
1026              textattr( ( RED << 4 ) | WHITE );
1027              cprintf( " VERIFY ERROR ! at %04lX", addr );
1028              textattr( ( CYAN << 4 ) | WHITE );
1029              errbeep();
1030          }
1031      }
1032  }
1033
1034  void edit_config( void )
1035  {
1036      int i, done;
1037
1038      textattr( ( CYAN << 4 ) | WHITE );
1039      _window(  1,0, 10,39 );
1040      _window( 11,1, 24,78 );
1041
1042      locate(  3, 1 ); cprintf( "Protection: " );
1043
1044      locate(  5, 1 ); cprintf( "Serial Pgm: " );
1045      locate(  6, 1 ); cprintf( "x2 clock  : " );
1046      locate(  7, 1 ); cprintf( "UsrRow Pgm: " );
1047      locate(  8, 1 ); cprintf( "CrystalClk: " );
1048
1049      locate( 14, 4 ); cprintf( "A : no protection" );
1050      locate( 14,43 ); cprintf( "B : MOVC protection" );
1051      locate( 15, 4 ); cprintf( "C : VERIFY protection" );
1052      locate( 15,43 ); cprintf( "D : EXT_EXEC protection" );
1053
1054      locate( 17, 4 ); cprintf( "E : Serial Pgm Enable toggle" );
1055      locate( 18, 4 ); cprintf( "F : x2 Clock Enable toggle" );
1056      locate( 19, 4 ); cprintf( "G : UsrRow Pgm Enable toggle" );
1057      locate( 20, 4 ); cprintf( "H : Crystal Clock Enable toggle" );
1058
1059      locate( 23, 4 ); cprintf( "Select options or <CR><ESC> to go back to the main
         menu ?" );
1060
1061      textattr( ( BLUE << 4 ) | WHITE );
1062      locate(  1, 5 ); cprintf( " Configuration Bit Setting :" );
1063      locate( 11,28 ); cprintf( " Configuration Options :" );
1064
1065      for(;;)
1066      {
1067          textattr( ( BLUE << 4 ) | WHITE );
1068
```

```
1069            locate( 3, 13 );
1070            switch( lock_bits )
1071            {
1072            default : cprintf( "none" ); lock_bits = 7; break;
1073            case 6  : cprintf( "MOVC" ); break;
1074            case 4  : cprintf( "MOVC & VERIFY" ); break;
1075            case 0  : cprintf( "MOVC & VERIFY & EXT_EXEC" ); break;
1076            }
1077
1078            for( i = 0; i < 4; ++i )
1079            {
1080                locate( 5 + i, 13 );
1081                cprintf( ( fuse_bits & ( 1 << i ) ) ? "disable" : "enable" );
1082            }
1083
1084            for( done = 0; !done; )
1085            {
1086                done = 1;
1087
1088                switch( toupper( getch() ) )
1089                {
1090                case 0: getch(); done = 0; break;
1091
1092                case '\n':
1093                case '\r':
1094                case 0x1B: return;
1095                case 'A' : lock_bits = 7; break;
1096                case 'B' : lock_bits = 6; break;
1097                case 'C' : lock_bits = 4; break;
1098                case 'D' : lock_bits = 0; break;
1099                case 'E' : fuse_bits ^= 1; break;
1100                case 'F' : fuse_bits ^= 2; break;
1101                case 'G' : fuse_bits ^= 4; break;
1102                case 'H' : fuse_bits ^= 8; break;
1103                default  : done = 0;
1104                }
1105            }
1106        }
1107    }
1108
1109    void type_select( void )
1110    {
1111        int done, i, num, len=15, left = 40;
1112        char no[10];
1113
1114        if( ( num = mfr[mfrno].numdevs ) > 14 )
1115        {
1116            left = 0;
1117            for( i = len = 0; i < num; ++i )
1118                if( ( done = strlen( mfr[mfrno].dev[i].name ) ) > len )
1119                    len = done;
1120        }
1121
1122        textattr( ( CYAN << 4 ) | WHITE );
1123        _window( 12,left, 23,79 );
1124
1125        textattr( ( BLUE << 4 ) | WHITE );
1126        locate( 12, left+5 ); cprintf( "  TYPE SELECT:" );
1127
1128        textattr( ( CYAN << 4 ) | WHITE );
1129
1130        for( i = 0; i < num; ++i )
1131        {
1132            locate( 13+(i%7), left + 1 + (i/7)*(len+4) );
1133            cprintf( "%d.%s", i, mfr[mfrno].dev[i].name );
1134        }
1135
1136        locate( 21, left+1 ); cprintf( "<CR> back to main menu." );
1137        locate( 22, left+1 ); cprintf( "SELECT NUMBER ?" );
1138
1139        for(;;)
1140        {
```

```
1141            for( no[0] = done = 0; !done; )
1142            {
1143                locate( 22, left+16 ); cprintf( "%s ", no );
1144                locate( 22, left+16+strlen(no) );
1145
1146                switch( i = getch() )
1147                {
1148                case 0:
1149                    getch();
1150                    break;
1151
1152                case 8:
1153                    if( ( i = strlen(no) ) != 0 )
1154                        no[i-1] = 0;
1155                    break;
1156
1157                case '\n':
1158                case '\r':
1159                    if( strlen( no ) &&
1160                        ( i = atoi(no) ) >= 0 && i < mfr[mfrno].numdevs )
1161                    {
1162                        devno = i;
1163                        ShowType();
1164                        return;
1165                    }
1166                    break;
1167
1168                case 0x1B:
1169                    return;
1170
1171                default:
1172                    if( isdigit( i ) )
1173                        strcat( no, (char*)&i );
1174                    break;
1175                }
1176            }
1177        }
1178    }
1179
1180    void mfr_select( void )
1181    {
1182        int done, i;
1183        char no[10];
1184
1185        textattr( ( CYAN << 4 ) | WHITE );
1186        _window( 12,40, 23,79 );
1187
1188        textattr( ( BLUE << 4 ) | WHITE );
1189        locate( 12, 45 ); cprintf( "  MFR SELECT:" );
1190
1191        textattr( ( CYAN << 4 ) | WHITE );
1192
1193        for( i = 0; i < sizeof(mfr) / sizeof(mfr[0]); ++i )
1194        {
1195            locate( 13+i, 41 ); cprintf( "%d.%s", i, mfr[i].name );
1196        }
1197
1198        locate( 21, 41 ); cprintf( "<CR> back to main menu." );
1199        locate( 22, 41 ); cprintf( "SELECT NUMBER ?" );
1200
1201        for(;;)
1202        {
1203            for( no[0] = done = 0; !done; )
1204            {
1205                locate( 22, 56 ); cprintf( "%s ", no );
1206                locate( 22, 56+strlen(no) );
1207
1208                switch( i = getch() )
1209                {
1210                case 0:
1211                    getch();
1212                    break;
```

```
1213
1214                case 8:
1215                    if( ( i = strlen(no) ) != 0 )
1216                        no[i-1] = 0;
1217                    break;
1218
1219                case '\n':
1220                case '\r':
1221                    if( strlen( no ) &&
1222                        ( i = atoi(no) ) >= 0 && i < sizeof(mfr) / sizeof(mfr[0]) )
1223                    {
1224                        mfrno = i;
1225                        if( devno >= mfr[mfrno].numdevs )
1226                            devno = 0;
1227                        ShowType();
1228                        return;
1229                    }
1230                    break;
1231
1232                case 0x1B:
1233                    return;
1234
1235                default:
1236                    if( isdigit( i ) )
1237                        strcat( no, (char*)&i );
1238                    break;
1239                }
1240            }
1241        }
1242    }
1243
1244    /*============================================================*/
1245    int main( void )
1246    {
1247        int redraw, first = 1;
1248        long tmpval;
1249
1250        /*------------------------------------------------------------*/
1251        /* main program starts here                                   */
1252        /*------------------------------------------------------------*/
1253
1254        getcwd( oldpath, 260 );
1255        strcpy( path, oldpath );
1256
1257        if( ( buffer = farmalloc( BUFSIZE ) ) == NULL )
1258            return -1;
1259        memset( (void far *)buffer, 0, BUFSIZE );
1260
1261        ReadConfig();
1262        delay(0);
1263
1264        for( ;; )
1265        {
1266            if( first )
1267            {
1268                first = 0;
1269
1270                init_hw();
1271                initdacs();
1272                setport( USERBITS, 0, 0 );
1273            }
1274
1275            textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 0,0, 24,79 );
1276
1277            locate( 0,0 ); cprintf( "Universal   Programmer" );
1278            locate( 1,0 ); cprintf( "MODEL: PC Based" );
1279            locate( 2,0 ); cprintf( "MPU 89Sxxxx section  " _VERSION_ );
1280
1281            textattr( ( BLUE << 4 ) + WHITE ); clscrn( 0,40, 6,79 );
1282
1283            ShowType();
1284
```

```
1285            textattr( ( BLUE << 4 ) + WHITE ); _window( 1,40, 6,79 );
1286            locate( 1,53 ); cprintf( " TARGET ZONE " );
1287            locate( 2,41 ); cprintf( "Buffer start addr.: %04lX", BufStart );
1288            locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
1289            locate( 4,41 ); cprintf( "       Check Sum  : %04X", Chks );
1290            locate( 5,41 ); cprintf( "Device start addr.: %04lX", DevStart );
1291
1292            _window( 3,69, 6,79 );
1293            locate( 4,71 ); cprintf( "COUNTER" );
1294            ShowCounter( 0 );
1295
1296            _window( 7,40, 10,79 );
1297            locate( 7,43 ); cprintf( " Device ID & Configuration bits " );
1298            locate( 8,42 ); cprintf( "Device ID :" );
1299            locate( 9,42 ); cprintf( "Configuration bits : " );
1300            ShowConfig();
1301
1302            textattr( ( CYAN << 4 ) | WHITE ); clscrn( 3,0, 23,38 );
1303
1304            locate( 3,0 ); cprintf( "-------------  Main Menu  -------------" );
1305            locate( 4,0 ); cprintf( "1. DOS SHELL                           " );
1306            locate( 5,0 ); cprintf( "2. Load BIN or HEX file to buffer      " );
1307            locate( 6,0 ); cprintf( "3. Save buffer to disk                 " );
1308            locate( 7,0 ); cprintf( "4. Edit buffer     7. Display buffer   " );
1309            locate( 8,0 ); cprintf( "5. Change I/O base address            " );
1310            locate( 9,0 ); cprintf( "6. Display loaded file history         " );
1311            locate(10,0 ); cprintf( "W. Swap hi-low bytes in buffer         " );
1312            locate(11,0 ); cprintf( "T. Type select     Z. Target zone      " );
1313            locate(12,0 ); cprintf( "B. Blank check     D. Display          " );
1314            if( _DATASIZE != 0 ) {
1315            locate(13,0 ); cprintf( "P. Program (Program Mem & Data Mem)    " );
1316            locate(14,0 ); cprintf( "A. Auto(B&S&P&V&L)                     " );
1317            locate(15,0 ); cprintf( "S. Erase Program & Data memory         " );
1318            } else {
1319            locate(13,0 ); cprintf( "                                       " );
1320            locate(14,0 ); cprintf( "P. Program         A. Auto(B&S&P&V&L)  " );
1321            locate(15,0 ); cprintf( "S. Erase Program memory                " );
1322            }
1323            locate(16,0 ); cprintf( "R. Read            V. Verify           " );
1324            locate(17,0 ); cprintf( "C. Compare and display error           " );
1325            locate(18,0 ); cprintf( "E. Configuration & ID code function    " );
1326            locate(19,0 ); cprintf( "L. Program ID & config. & protect bits " );
1327            locate(20,0 ); cprintf( "Q. Quit                                " );
1328            locate(21,0 ); cprintf( "---------------------------------------" );
1329            locate(22,0 ); cprintf( "Allocation Buffer size : %uK bytes", BUFSIZE/1024 );
1330            if( _DATASIZE != 0 ) {
1331            locate(23,0 ); cprintf( "Data memory buffer at %04lX ~ %04lX", _CODESIZE,
                _CODESIZE + _DATASIZE - 1 );
1332            }
1333
1334            for( redraw = 0; !redraw; )
1335            {
1336                int c;
1337
1338                textattr( ( BLUE << 4 ) + WHITE ); clscrn( 24,0, 24,38 );
1339
1340                locate( 24,0 ); cprintf( "Select function ? " );
1341
1342                for(;;)
1343                {
1344                    if( ( c = getch() ) != 0 )
1345                        break;
1346                    getch();                /* neglect extended code */
1347                }
1348
1349                switch( c = toupper(c) )
1350                {
1351                case '1': dos_shell( "" ); redraw = 1; break;
1352                case '2': tmpval = bufsize; bufsize = 0x8000;
1353                          memset( (void far *)buffer, 0, BUFSIZE );     // clear buffer
1354                          addr = 0;
1355                          load_file();
```

```
1356                        bufsize= tmpval;
1357                        redraw = 1; break;
1358                case '3': save_file(); break;
1359                case '4': tmpval = bufsize; bufsize = 0x8000;
1360                        addr = 0;
1361                        edit_buffer();
1362                        bufsize= tmpval;
1363                        redraw = 1; break;
1364                case '5': set_io_adr(); first = redraw = 1; break;
1365                case '7': disp_buffer(); redraw = 1; break;
1366
1367                case 'M': mfr_select(); redraw = first = 1; break;
1368                case 'T': type_select(); redraw = first = 1; break;
1369                case 'E': edit_config(); redraw = 1; break;
1370
1371                case 'R': flash_read(); break;
1372                case 'B': flash_check(); break;
1373                case 'S': flash_erase(); break;
1374                case 'P': flash_program(); break;
1375                case 'V': flash_verify(); break;
1376                case 'L': flash_protect(); break;
1377                case 'A': flash_auto(); break;
1378
1379    //          case '\n':
1380    //          case '\r': redraw = 1; break;       // refresh
1381                }
1382
1383                setport( USERBITS, 0, 0 );
1384
1385                if( c == 'Q' )
1386                {
1387                    WriteConfig();
1388                    textattr( LIGHTGRAY ); clrscr();
1389                    chdir( oldpath );
1390                    if( buffer ) farfree( (void far *)buffer );
1391                    return( 0 );
1392                }
1393
1394                textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 11,40, 23,79 );
1395            }
1396        }
1397    }
1398
1399
```