```
#include "all03.h"

#include <bios.h>

#define _VERSION_           "1.00"

#define MODE_WRITE      0x0E        // Bit 0 - 3 : P3.3 - P3.7
#define MODE_READ       0x0C
#define MODE_LOCK1      0x0F
#define MODE_LOCK2      0x03
#define MODE_ERASE      0x01
#define MODE_SIGNAT     0x00

unsigned char _RST, _RDY, _PROG, _XTAL;
unsigned char _VCC, _GND, _MODE[4], _DATA[8];

struct DEV {
    char name[20];
    unsigned Size, DataSize;    // ROM & EEPROM sizes in bytes
    char rst,rdy,prog,xtal;     // Pin numbers for RESET,RDY/BUSY,PROG and XTAL
    char vcc, gnd, mode[4];     // Pin numbers for VCC, GND and MODE0..3
    char data[8];               // Pin numbers for data0..7
    unsigned char UBmin, UBmax; // min. and max. Vcc value
};

struct DEV ATMEL_devs[] = {
//          TYPE        ROMSIZE EESIZE  rst rdy prg xtl vcc gnd md0 md1 md2 md3 d0 d1 d2 d3 d4 d5 d6 d7 Umin Umax
/* 0*/  { "AT89C1051",  0x0400, 0x0000, 11, 13, 16, 15, 30, 20, 17, 18, 19, 21, 22,23,24,25,26,27,28,29, 30, 60 },
/* 1*/  { "AT89C2051",  0x0800, 0x0000, 11, 13, 16, 15, 30, 20, 17, 18, 19, 21, 22,23,24,25,26,27,28,29, 30, 60 },
/* 2*/  { "AT89C4051",  0x1000, 0x0000, 11, 13, 16, 15, 30, 20, 17, 18, 19, 21, 22,23,24,25,26,27,28,29, 30, 60 }
};

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

int mfrno = 0, devno = 0;

#define BUFSIZE     0x8000U

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

void ShowCounter( long val )
{
    struct text_info ti;

    gettextinfo( &ti );

    textattr( ( BLUE << 4 ) + WHITE );
    locate( 5, 73 ); cprintf( "%04lX", val );

    textattr( ti.attribute );
    gotoxy( ti.curx, ti.cury );
}

void ShowType( void )
{
    if( mfrno < 0 || mfrno > sizeof(mfr) / sizeof(mfr[0]) )
        mfrno = 0;
    if( devno < 0 || devno >= mfr[mfrno].numdevs )
        devno = 0;

```

```
69      bufsize = mfr[mfrno].dev[devno].Size;
70      BufEnd = bufsize - 1;
71
72      textattr( ( BLUE << 4 ) | WHITE );
73
74      locate( 0,40 ); cprintf( "*Mfr.: %s", mfr[mfrno].name );
75      locate( 0,60 ); cprintf( "*TYPE: %s", mfr[mfrno].dev[devno].name );
76
77      locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
78
79      textattr( ( CYAN << 4 ) | WHITE );
80  }
81
82  void setmode( int md )
83  {
84      setpin( _MODE[0], TTLID, (md & 1) ? 1:0 );
85      setpin( _MODE[1], TTLID, (md & 2) ? 1:0 );
86      setpin( _MODE[2], TTLID, (md & 4) ? 1:0 );
87      setpin( _MODE[3], TTLID, (md & 8) ? 1:0 );
88  }
89
90  void power( int voltage )
91  {
92      int i;
93
94      _RST = mfr[mfrno].dev[devno].rst;
95      _RDY = mfr[mfrno].dev[devno].rdy;
96      _PROG= mfr[mfrno].dev[devno].prog;
97      _XTAL= mfr[mfrno].dev[devno].xtal;
98
99      _VCC = mfr[mfrno].dev[devno].vcc;
100     _GND = mfr[mfrno].dev[devno].gnd;
101
102     for( i = 0; i < 4; ++i )
103         _MODE[i]  = mfr[mfrno].dev[devno].mode[i];
104
105     for( i = 0; i < 8; ++i )
106         _DATA[i]  = mfr[mfrno].dev[devno].data[i];
107
108     // we dont need VHH
109     setdac( VHHID, 0 );                                          // VHH = 0V
110     for( i = 0; i <= 4; ++i ) setport( VHHENID, i, 0 );          // no VHH
111     setport( VHHENCID,0, 0 );                                    // no VHHC
112     setport( VHHENCID,1, 0 );                                    // no VHHC
113
114     // Can't use Pins 2,3,4,6 and 8 for neither Vop nor Vhh
115     if( _RST == 2 || _RST == 3 || _RST == 4 || _RST == 6 || _RST == 8 ||
116         ( _RST > 32 && _RST != 36 ) )
117     {
118         errbeep();
119         textattr( ( RED << 4 ) | WHITE );
120         locate( 41, 23 ); cprintf( "Connect pin 1 to %d", _RST );
121         textattr( ( CYAN << 4 ) | WHITE );
122         _RST = 1;                                        // force using Pin 1
123     }
124     setport( OTHERENID, 0, 0 );          // Pin 20 = GND
125
126     if( voltage )
127     {
128         // Set all pins HIGH
129         for( i = 1; i <= 40; ++i )
130             setpin( i, TTLID, 1 );
131
132         // Set RST and XTAL pin LOW
133         setpin( _RST, TTLID, 0 );
134         setpin( _XTAL, TTLID, 0 );
135
136         // Set MODE
137         setmode( MODE_ERASE );                   // CHIP ERASE preset
138
139         setdac( VCCID, voltage );                // Vcc = xV
140         setpin( _VCC, VCCENID, 1 );              // set Vcc pin
```

```
141             setdac( VOPID, 120 );                   // Vpp = 12V
142             delay( 500 );                           // wait to stabilize
143
144             setpin( _RST, TTLID, 1 );               // RST = H
145         }
146         else
147         {
148             setpin( _RST, VOPENID, 0 );             // remove Vpp
149             setpin( _RST, TTLID, 0 );               // pull reset low
150
151             setdac( VOPID, 0 );                     // Vpp = 0V
152             setdac( VCCID, 0 );                     // Vcc = 0V
153
154             setpin( _VCC, VCCENID, 0 );             // disable Vcc(s)
155
156             // Set all pins LOW
157             for( i = 1; i <= 40; ++i )
158                 setpin( i, TTLID, 0 );
159         }
160     }
161
162     void reset( void )
163     {
164         setpin( _RST, VOPENID, 0 ); dly1u();    // remove Vpp (if not already done)
165
166         setpin( _RST, TTLID, 0 ); dly1u();      // pulse RST H->L->H
167         setpin( _RST, TTLID, 1 ); dly10m();
168     }
169
170     int do_erase( void )
171     {
172         setmode( MODE_ERASE );
173
174         setpin( _RST, VOPENID, 1 ); dly10u();   // apply Vpp
175
176         setpin( _PROG, TTLID, 0 );              // pulse PROG low for 10ms
177         dly10m();
178         setpin( _PROG, TTLID, 1 );
179
180         dly10u(); setpin( _RST, VOPENID, 0 );   // remove Vpp
181         return 1;
182     }
183
184     int do_read( void )
185     {
186         int i, val = 0;
187
188         for( i = 0; i < 8; ++i )
189             setpin( _DATA[i], TTLID, 1 );   // release pin drivers on data pins
190
191         setmode( MODE_READ );
192
193         for( i = 7; i >= 0; --i )
194             val = ( val << 1 ) | getpin( _DATA[i] );
195
196         setmode( MODE_ERASE );
197         return val;
198     }
199
200     void do_load( int val )
201     {
202         int i;
203
204         for( i = 0; i < 8; ++i )
205         {
206             setpin( _DATA[i], TTLID, val & 1 );
207             val >>= 1;
208         }
209     }
210
211     void do_increment( int num )
212     {
```

```
213         do
214         {
215             setpin( _XTAL, TTLID, 1 ); dly1u();
216             setpin( _XTAL, TTLID, 0 );
217             ++addr;
218         } while( --num );
219     }
220
221     int do_prog( void )
222     {
223         long t;
224
225         setpin( _RST, VOPENID, 1 ); dly20u();
226
227         setpin( _PROG, TTLID, 0 ); dly1u();
228         setpin( _PROG, TTLID, 1 ); dly1u();
229
230         t = biostime( 0, 0 );
231
232         while( biostime( 0, 0 ) - t < 2 )   // max. 55..110ms
233         {
234             if( getpin( _RDY ) )
235                 break;
236             dly100u();
237         }
238
239         dly20u(); setpin( _RST, VOPENID, 0 );
240
241         return getpin( _RDY ) ? 1 : 0;
242     }
243
244     int check( void )       // BLANK check, return 1 on success
245     {
246         unsigned end = mfr[mfrno].dev[devno].Size;
247
248         reset();
249
250         for( addr = BufStart; addr < end; )
251         {
252             if( ( addr & 0xFF ) == 0 ) ShowCounter( addr );
253
254             if( do_read() != 0xFF )
255             {
256                 ShowCounter( addr );
257                 return 0;
258             }
259
260             do_increment( 1 );
261         }
262         ShowCounter( addr );
263         return 1;
264     }
265
266     int program( void )
267     {
268         unsigned end = mfr[mfrno].dev[devno].Size;
269
270         reset();
271
272         setmode( MODE_WRITE );
273
274         for( addr = BufStart; addr < end; )
275         {
276             if( ( addr & 0xFF ) == 0 ) ShowCounter( addr );
277
278             do_load( buffer[addr] );
279
280             if( !do_prog() )
281                 return 0;
282
283             do_increment( 1 );
284         }
```

```
285         ShowCounter( addr );
286         return 1;
287     }
288
289     int protect( void )
290     {
291         setmode( MODE_LOCK1 );
292         if( !do_prog() )
293             return 0;
294
295         setmode( MODE_LOCK2 );
296         return do_prog();
297     }
298
299
300     int read_verify( int verify )
301     {
302         unsigned end = mfr[mfrno].dev[devno].Size;
303
304         reset();
305
306         for( addr = BufStart; addr < end; )
307         {
308             if( ( addr & 0xFF ) == 0 ) ShowCounter( addr );
309
310             if( verify )
311             {
312                 if( buffer[addr] != do_read() )
313                 {
314                     ShowCounter( addr );
315                     return 0;
316                 }
317             }
318             else
319                 Chks += ( buffer[addr] = do_read() & 0xFF );
320
321             do_increment( 1 );
322         }
323         ShowCounter( addr );
324         return 1;
325     }
326
327     int flash_check( void )
328     {
329         int done;
330
331         textattr( ( CYAN << 4 ) | WHITE );
332         _window( 12,40, 23,79 );
333
334         textattr( ( BLUE << 4 ) | WHITE );
335         locate( 12, 45 ); cprintf( "   BLANK CHECK device:" );
336
337         for(;;)
338         {
339             textattr( ( CYAN << 4 ) | WHITE );
340             locate( 13, 41 ); cprintf( "Ready to check (Y/<CR>)? " );
341
342             for( done = 0; !done; )
343             {
344                 switch( getch() )
345                 {
346                 case 0:
347                     getch();
348                     break;
349
350                 case '\n':
351                 case '\r':
352                 case 0x1B:
353                     return;
354
355                 case 'y':
356                 case 'Y':
```

```
                done = 1;
            }
        }

        clscrn( 14,41, 22,78 );

        setport( USERBITS, 0, 0 );

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

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

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

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

void flash_program( void )
{
    int done;

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

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

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

        for( done = 0; !done; )
        {
            switch( getch() )
            {
            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 );

```

```
429             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
430             locate( 14, 41 ); cprintf( "Programming now... " );
431
432             power(50); done = program(); power(0);
433
434             textattr( ( CYAN << 4 ) | WHITE );
435             locate( 14, 41 ); cprintf( "Programming now... " );
436
437             locate( 15, 41 );
438             if( done )
439             {
440                 ShowCounter( BufEnd );
441                 putchar( 7 );
442                 setport( USERBITS, 0, 8 );
443                 cprintf( " OK !" );
444             }
445             else
446             {
447                 errbeep();
448                 textattr( ( RED << 4 ) | WHITE );
449                 cprintf( "Program error ! at %04lX", addr );
450                 textattr( ( CYAN << 4 ) | WHITE );
451             }
452         }
453     }
454
455     void flash_protect( void )
456     {
457         int done;
458
459         textattr( ( CYAN << 4 ) | WHITE );
460         _window( 12,40, 23,79 );
461
462         textattr( ( BLUE << 4 ) | WHITE );
463         locate( 12, 42 ); cprintf( " Program ID & CFG & protect bits: " );
464
465         for(;;)
466         {
467             textattr( ( CYAN << 4 ) | WHITE );
468             locate( 13, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
469
470             for( done = 0; !done; )
471             {
472                 switch( getch() )
473                 {
474                 case 0:
475                     getch();
476                     break;
477
478                 case '\n':
479                 case '\r':
480                 case 0x1B:
481                     return;
482
483                 case 'y':
484                 case 'Y':
485                     done = 1;
486                 }
487             }
488
489             clscrn( 14,41, 22,78 );
490
491             setport( USERBITS, 0, 0 );
492
493             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
494             locate( 14, 41 ); cprintf( "Programming now... " );
495
496             power(50); done = protect(); power(0);
497
498             textattr( ( CYAN << 4 ) | WHITE );
499             locate( 14, 41 ); cprintf( "Programming now... " );
500
```

```
501             locate( 15, 41 );
502             if( done )
503             {
504                 ShowCounter( BufEnd );
505                 putchar( 7 );
506                 setport( USERBITS, 0, 8 );
507                 cprintf( " OK !" );
508             }
509             else
510             {
511                 errbeep();
512                 textattr( ( RED << 4 ) | WHITE );
513                 cprintf( "Program error ! at %04lX", addr );
514                 textattr( ( CYAN << 4 ) | WHITE );
515             }
516         }
517     }
518
519     void flash_read( void )
520     {
521         int done;
522
523         textattr( ( CYAN << 4 ) | WHITE );
524         _window( 12,40, 23,79 );
525
526         textattr( ( BLUE << 4 ) | WHITE );
527         locate( 12, 45 ); cprintf( "   READ to buffer :" );
528
529         for(;;)
530         {
531             textattr( ( CYAN << 4 ) | WHITE );
532             locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
533
534             for( done = 0; !done; )
535             {
536                 switch( getch() )
537                 {
538                 case 0:
539                     getch();
540                     break;
541
542                 case '\n':
543                 case '\r':
544                 case 0x1B:
545                     return;
546
547                 case 'y':
548                 case 'Y':
549                     done = 1;
550
551                 case 'e':
552                 case 'E':
553                 case 'o':
554                 case 'O':
555                     done = 1;
556                 }
557             }
558
559             clscrn( 14,41, 22,78 );
560
561             setport( USERBITS, 0, 0 );
562
563             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
564             locate( 14, 41 ); cprintf( "Reading now... " );
565
566             Chks = 0;
567             power(50); read_verify(0); power(0);
568
569             textattr( ( CYAN << 4 ) | WHITE );
570             locate( 14, 41 ); cprintf( "Reading now... " );
571
572             locate( 15, 41 );
```

```
573             putchar( 7 );
574             cprintf( " OK !" );
575
576             textattr( ( BLUE << 4 ) | WHITE );
577             locate( 4, 41 ); cprintf( "       Check Sum  : %04X", Chks );
578         }
579     }
580
581     void flash_verify( void )
582     {
583         int done;
584
585         textattr( ( CYAN << 4 ) | WHITE );
586         _window( 12,40, 23,79 );
587
588         textattr( ( BLUE << 4 ) | WHITE );
589         locate( 12, 45 ); cprintf( "   VERIFY with buffer :" );
590
591         for(;;)
592         {
593             textattr( ( CYAN << 4 ) | WHITE );
594             locate( 13, 41 ); cprintf( "Ready to verify (Y/Even/Odd/<CR>)? " );
595
596             for( done = 0; !done; )
597             {
598                 switch( getch() )
599                 {
600                 case 0:
601                     getch();
602                     break;
603
604                 case '\n':
605                 case '\r':
606                 case 0x1B:
607                     return;
608
609                 case 'y':
610                 case 'Y':
611                     done = 1;
612
613                 case 'e':
614                 case 'E':
615                 case 'o':
616                 case 'O':
617                     done = 1;
618                 }
619             }
620
621             clscrn( 14,41, 22,78 );
622
623             setport( USERBITS, 0, 0 );
624
625             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
626             locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
627
628             power(mfr[mfrno].dev[devno].UBmin);
629             done = read_verify(1);
630             power(0);
631
632             textattr( ( CYAN << 4 ) | WHITE );
633             locate( 14, 41 ); cprintf( "Verifying now @ VDDmin... " );
634
635             locate( 15, 41 );
636             if( done )
637             {
638                 ShowCounter( BufEnd );
639                 putchar( 7 );
640                 cprintf( " OK !" );
641                 setport( USERBITS, 0, 8 );
642             }
643             else
644             {
```

```
645                 errbeep();
646                 textattr( ( RED << 4 ) | WHITE );
647                 cprintf( " VERIFY ERROR ! at %04lX", addr );
648                 textattr( ( CYAN << 4 ) | WHITE );
649                 continue;
650             }
651
652             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
653             locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
654
655             power(mfr[mfrno].dev[devno].UBmax);
656             done = read_verify(1);
657             power(0);
658
659             textattr( ( CYAN << 4 ) | WHITE );
660             locate( 16, 41 ); cprintf( "Verifying now @ VDDmax... " );
661
662             locate( 17, 41 );
663             if( done )
664             {
665                 ShowCounter( BufEnd );
666                 putchar( 7 );
667                 cprintf( " OK !" );
668                 setport( USERBITS, 0, 8 );
669             }
670             else
671             {
672                 errbeep();
673                 textattr( ( RED << 4 ) | WHITE );
674                 cprintf( " VERIFY ERROR ! at %04lX", addr );
675                 textattr( ( CYAN << 4 ) | WHITE );
676             }
677         }
678     }
679
680     void flash_erase( void )
681     {
682         int done;
683
684         textattr( ( CYAN << 4 ) | WHITE );
685         _window( 12,40, 23,79 );
686
687         textattr( ( BLUE << 4 ) | WHITE );
688         locate( 12, 45 ); cprintf( "  EEPROM Erase:" );
689
690         for(;;)
691         {
692             textattr( ( CYAN << 4 ) | WHITE );
693             locate( 13, 41 ); cprintf( "Ready to erase (Y/<CR>)? " );
694
695             for( done = 0; !done; )
696             {
697                 switch( getch() )
698                 {
699                 case 0:
700                     getch();
701                     break;
702
703                 case '\n':
704                 case '\r':
705                 case 0x1B:
706                     return;
707
708                 case 'y':
709                 case 'Y':
710                     done = 1;
711                 }
712             }
713
714             clscrn( 14,41, 22,78 );
715
716             setport( USERBITS, 0, 0 );
```

```
717
718             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
719             locate( 14, 41 ); cprintf( "Erase now... " );
720
721             power(50); done = do_erase(); power(0);
722
723             textattr( ( CYAN << 4 ) | WHITE );
724             locate( 14, 41 ); cprintf( "Erase now... " );
725
726             locate( 15, 41 );
727             if( done )
728             {
729                 putchar( 7 );
730                 cprintf( " OK !" );
731                 setport( USERBITS, 0, 8 );
732             }
733             else
734             {
735                 errbeep();
736                 textattr( ( RED << 4 ) | WHITE );
737                 cprintf( " ERROR ! " );
738                 textattr( ( CYAN << 4 ) | WHITE );
739
740                 locate( 21, 41 ); cprintf( "press any key to continue" );
741                 if( getch() == 0 ) getch();
742             }
743
744             clscrn( 13,41, 22,78 );
745         }
746     }
747
748     void flash_auto( void )
749     {
750         int done, l;
751
752         textattr( ( CYAN << 4 ) | WHITE );
753         _window( 12,40, 23,79 );
754
755         textattr( ( BLUE << 4 ) | WHITE );
756         locate( 12, 45 ); cprintf( "   AUTO :" );
757
758         for(;;)
759         {
760             textattr( ( CYAN << 4 ) | WHITE );
761             locate( 13, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
762
763             for( done = 0; !done; )
764             {
765                 switch( getch() )
766                 {
767                 case 0:
768                     getch();
769                     break;
770
771                 case '\n':
772                 case '\r':
773                 case 0x1B:
774                     return;
775
776                 case 'y':
777                 case 'Y':
778                     done = 1;
779
780                 case 'e':
781                 case 'E':
782                 case 'o':
783                 case 'O':
784                     done = 1;
785                 }
786             }
787
788             clscrn( 14,41, 22,78 );
```

```
789
790             setport( USERBITS, 0, 0 );
791
792             l = 14;
793
794             // BLANK CHECK
795
796             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
797             locate( l, 41 ); cprintf( "Blank checking now... " );
798
799             power(50); done = check(); power(0);
800
801             textattr( ( CYAN << 4 ) | WHITE );
802             locate( l++, 41 ); cprintf( "Blank checking now... " );
803
804             locate( l++, 41 );
805
806             if( done )
807             {
808                 ShowCounter( BufEnd );
809                 cprintf( " OK !" );
810             }
811             else
812             {
813                 errbeep();
814                 textattr( ( RED << 4 ) | WHITE );
815                 cprintf( "Blank check error at %04lX", addr );
816                 textattr( ( CYAN << 4 ) | WHITE );
817
818             // ERASE
819
820                 l -= 2;
821
822                 clscrn( l, 41, l+1, 78 );
823
824                 textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
825                 locate( l, 41 ); cprintf( "Erase now... " );
826
827                 power(50); done = do_erase(); power(0);
828
829                 textattr( ( CYAN << 4 ) | WHITE );
830                 locate( l++, 41 ); cprintf( "Erase now... " );
831
832                 locate( l++, 41 );
833
834                 if( done )
835                     cprintf( " OK !" );
836                 else
837                 {
838                     errbeep();
839                     textattr( ( RED << 4 ) | WHITE );
840                     cprintf( " ERROR" );
841                     textattr( ( CYAN << 4 ) | WHITE );
842
843                     continue;
844                 }
845             }
846
847             // PROGRAM
848
849             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
850             locate( l, 41 ); cprintf( "Programming now... " );
851
852             power(50); done = program(); power(0);
853
854             textattr( ( CYAN << 4 ) | WHITE );
855             locate( l++, 41 ); cprintf( "Programming now... " );
856
857             locate( l++, 41 );
858             if( done )
859             {
860                 ShowCounter( BufEnd );
```

```
861             cprintf( " OK !" );
862         }
863         else
864         {
865             errbeep();
866             textattr( ( RED << 4 ) | WHITE );
867             cprintf( "Program error ! at %04lX", addr );
868             textattr( ( CYAN << 4 ) | WHITE );
869 
870             continue;
871         }
872 
873         // VERIFY
874 
875         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
876         locate( l, 41 ); cprintf( "VDD max verifying now..." );
877 
878         power(mfr[mfrno].dev[devno].UBmax);
879         done = read_verify(1);
880         power(0);
881 
882         textattr( ( CYAN << 4 ) | WHITE );
883         locate( l++, 41 ); cprintf( "VDD max verifying now..." );
884 
885         locate( l++, 41 );
886         if( done )
887         {
888             ShowCounter( BufEnd );
889             putchar( 7 );
890             cprintf( " OK !" );
891             setport( USERBITS, 0, 8 );
892         }
893         else
894         {
895             errbeep();
896             textattr( ( RED << 4 ) | WHITE );
897             cprintf( " VERIFY ERROR ! at %04lX", addr );
898             textattr( ( CYAN << 4 ) | WHITE );
899             continue;
900         }
901 
902         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
903         locate( l, 41 ); cprintf( "VDD min verifying now..." );
904 
905         power(mfr[mfrno].dev[devno].UBmin);
906         done = read_verify(1);
907         power(0);
908 
909         textattr( ( CYAN << 4 ) | WHITE );
910         locate( l++, 41 ); cprintf( "VDD min verifying now..." );
911 
912         locate( l, 41 );
913         if( done )
914         {
915             ShowCounter( BufEnd );
916             cprintf( " OK !" );
917             setport( USERBITS, 0, 8 );
918             putchar( 7 );
919         }
920         else
921         {
922             textattr( ( RED << 4 ) | WHITE );
923             cprintf( " VERIFY ERROR ! at %04lX", addr );
924             textattr( ( CYAN << 4 ) | WHITE );
925             errbeep();
926         }
927     }
928 }
929 
930 #if(0)
931 void edit_config( void )
932 {
```

```
933         static char *osctype[4] = { "LP","XT","HS","RC" };
934         int i, done;
935 
936         textattr( ( CYAN << 4 ) | WHITE );
937         _window(  1,0, 10,39 );
938         _window( 11,1, 24,78 );
939 
940         locate(  2, 1 ); cprintf( "CODE Protection:" );
941         locate(  3, 1 ); cprintf( "DATA Protection:" );
942 
943         locate( 16, 4 ); cprintf( "K : CODE protection toggle" );
944         locate( 16,43 ); cprintf( "L : DATA protection toggle" );
945 
946         locate( 23, 4 ); cprintf( "Select options or <CR><ESC> to go back to the main
            menu ?" );
947 
948         textattr( ( BLUE << 4 ) | WHITE );
949         locate(  1, 5 ); cprintf( " Configuration Bit Setting :" );
950         locate( 11,28 ); cprintf( " Configuration Options :" );
951 
952         for(;;)
953         {
954             textattr( ( BLUE << 4 ) | WHITE );
955 
956             for( done = 0; !done; )
957             {
958                 done = 1;
959 
960                 switch( toupper( getch() ) )
961                 {
962                 case 0: getch(); done = 0; break;
963 
964                 case '\n':
965                 case '\r':
966                 case 0x1B: return;
967 
968                 case 'K' : // CODE protection toggle
969                     cfg = ( cfg & ~0x3030 ) | ( ( (cfg&0x3030) + 0x1010) & 0x3030 );
                        break;
970 
971                 case 'L' : // DATA protection toggle
972                     cfg ^= 0x100; break;
973 
974                 default: done = 0;
975                 }
976             }
977         }
978     }
979     #endif
980 
981     void type_select( void )
982     {
983         int done, i, num, len=15, left = 40;
984         char no[10];
985 
986         if( ( num = mfr[mfrno].numdevs ) > 14 )
987         {
988             left = 0;
989             for( i = len = 0; i < num; ++i )
990                 if( ( done = strlen( mfr[mfrno].dev[i].name ) ) > len )
991                     len = done;
992         }
993 
994         textattr( ( CYAN << 4 ) | WHITE );
995         _window( 12,left, 23,79 );
996 
997         textattr( ( BLUE << 4 ) | WHITE );
998         locate( 12, left+5 ); cprintf( "  TYPE SELECT:" );
999 
1000        textattr( ( CYAN << 4 ) | WHITE );
1001 
1002        for( i = 0; i < num; ++i )
```

```
1003        {
1004            locate( 13+(i%7), left + 1 + (i/7)*(len+4) );
1005            cprintf( "%d.%s", i, mfr[mfrno].dev[i].name );
1006        }
1007
1008        locate( 21, left+1 ); cprintf( "<CR> back to main menu." );
1009        locate( 22, left+1 ); cprintf( "SELECT NUMBER ?" );
1010
1011        for(;;)
1012        {
1013            for( no[0] = done = 0; !done; )
1014            {
1015                locate( 22, left+16 ); cprintf( "%s ", no );
1016                locate( 22, left+16+strlen(no) );
1017
1018                switch( i = getch() )
1019                {
1020                case 0:
1021                    getch();
1022                    break;
1023
1024                case 8:
1025                    if( ( i = strlen(no) ) != 0 )
1026                        no[i-1] = 0;
1027                    break;
1028
1029                case '\n':
1030                case '\r':
1031                    if( strlen( no ) &&
1032                        ( i = atoi(no) ) >= 0 && i < mfr[mfrno].numdevs )
1033                    {
1034                        devno = i;
1035                        ShowType();
1036                        return;
1037                    }
1038                    break;
1039
1040                case 0x1B:
1041                    return;
1042
1043                default:
1044                    if( isdigit( i ) )
1045                        strcat( no, (char*)&i );
1046                    break;
1047                }
1048            }
1049        }
1050    }
1051
1052    void mfr_select( void )
1053    {
1054        int done, i;
1055        char no[10];
1056
1057        textattr( ( CYAN << 4 ) | WHITE );
1058        _window( 12,40, 23,79 );
1059
1060        textattr( ( BLUE << 4 ) | WHITE );
1061        locate( 12, 45 ); cprintf( "  MFR SELECT:" );
1062
1063        textattr( ( CYAN << 4 ) | WHITE );
1064
1065        for( i = 0; i < sizeof(mfr) / sizeof(mfr[0]); ++i )
1066        {
1067            locate( 13+i, 41 ); cprintf( "%d.%s", i, mfr[i].name );
1068        }
1069
1070        locate( 21, 41 ); cprintf( "<CR> back to main menu." );
1071        locate( 22, 41 ); cprintf( "SELECT NUMBER ?" );
1072
1073        for(;;)
1074        {
```

```
1075            for( no[0] = done = 0; !done; )
1076            {
1077                locate( 22, 56 ); cprintf( "%s ", no );
1078                locate( 22, 56+strlen(no) );
1079
1080                switch( i = getch() )
1081                {
1082                case 0:
1083                    getch();
1084                    break;
1085
1086                case 8:
1087                    if( ( i = strlen(no) ) != 0 )
1088                        no[i-1] = 0;
1089                    break;
1090
1091                case '\n':
1092                case '\r':
1093                    if( strlen( no ) &&
1094                        ( i = atoi(no) ) >= 0 && i < sizeof(mfr) / sizeof(mfr[0]) )
1095                    {
1096                        mfrno = i;
1097                        if( devno >= mfr[mfrno].numdevs )
1098                            devno = 0;
1099                        ShowType();
1100                        return;
1101                    }
1102                    break;
1103
1104                case 0x1B:
1105                    return;
1106
1107                default:
1108                    if( isdigit( i ) )
1109                        strcat( no, (char*)&i );
1110                    break;
1111                }
1112            }
1113        }
1114    }
1115
1116    /*=========================================================*/
1117    int main( void )
1118    {
1119        int ch = 0, redraw, first = 1;
1120        long tmpval;
1121
1122        /*---------------------------------------------------------*/
1123        /* main program starts here                                */
1124        /*---------------------------------------------------------*/
1125
1126        getcwd( oldpath, 260 );
1127        strcpy( path, oldpath );
1128
1129        if( ( buffer = farmalloc( BUFSIZE ) ) == NULL )
1130            return -1;
1131
1132        memset( (void far *)buffer, 0, BUFSIZE );
1133
1134        ReadConfig();
1135        delay(0);
1136
1137        for( ;; )
1138        {
1139            if( first )
1140            {
1141                first = 0;
1142
1143                init_hw();
1144                initdacs();
1145                setport( USERBITS, 0, 0 );
1146            }
```

```
1147
1148            textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 0,0, 24,79 );
1149
1150            locate( 0,0 ); cprintf( "Universal   Programmer" );
1151            locate( 1,0 ); cprintf( "MODEL: PC Based" );
1152            locate( 2,0 ); cprintf( "MPU 89Cx051 section  " _VERSION_ );
1153
1154            textattr( ( BLUE << 4 ) + WHITE ); clscrn( 0,40, 6,79 );
1155
1156            ShowType();
1157
1158            textattr( ( BLUE << 4 ) + WHITE ); _window( 1,40, 6,79 );
1159            locate( 1,53 ); cprintf( " TARGET ZONE " );
1160            locate( 2,41 ); cprintf( "Buffer start addr.: %04lX", BufStart );
1161            locate( 3,41 ); cprintf( "       end   addr.: %04lX", BufEnd );
1162            locate( 4,41 ); cprintf( "       Check Sum  : %04X", Chks );
1163            locate( 5,41 ); cprintf( "Device start addr.: %04lX", DevStart );
1164
1165            _window( 3,69, 6,79 );
1166            locate( 4,71 ); cprintf( "COUNTER" );
1167            ShowCounter( 0 );
1168
1169            textattr( ( CYAN << 4 ) | WHITE ); clscrn( 3,0, 23,38 );
1170
1171            locate( 3,0 ); cprintf( "-------------  Main Menu  -------------" );
1172            locate( 4,0 ); cprintf( "1. DOS SHELL                           " );
1173            locate( 5,0 ); cprintf( "2. Load BIN or HEX file to buffer      " );
1174            locate( 6,0 ); cprintf( "3. Save buffer to disk                 " );
1175            locate( 7,0 ); cprintf( "4. Edit buffer     7. Display buffer   " );
1176            locate( 8,0 ); cprintf( "5. Change I/O base address            " );
1177            locate( 9,0 ); cprintf( "6. Display loaded file history         " );
1178            locate(10,0 ); cprintf( "W. Swap hi-low bytes in buffer         " );
1179            locate(11,0 ); cprintf( "T. Type select     Z. Target zone      " );
1180            locate(12,0 ); cprintf( "B. Blank check     D. Display          " );
1181            if( ( ch = mfr[mfrno].dev[devno].DataSize ) != 0 ) {
1182            locate(13,0 ); cprintf( "P. Program (Program Mem & Data Mem)    " );
1183            locate(14,0 ); cprintf( "A. Auto(B&S&P&V&L)                     " );
1184            locate(15,0 ); cprintf( "S. Erase Program & Data memory         " );
1185            } else {
1186            locate(13,0 ); cprintf( "                                       " );
1187            locate(14,0 ); cprintf( "P. Program         A. Auto(B&S&P&V&L)  " );
1188            locate(15,0 ); cprintf( "S. Erase Program memory                " );
1189            }
1190            locate(16,0 ); cprintf( "R. Read            V. Verify           " );
1191            locate(17,0 ); cprintf( "C. Compare and display error           " );
1192            locate(18,0 ); cprintf( "E. Configuration & ID code function    " );
1193            locate(19,0 ); cprintf( "L. Program ID & config. & protect bits " );
1194            locate(20,0 ); cprintf( "Q. Quit                                " );
1195            locate(21,0 ); cprintf( "---------------------------------------" );
1196            locate(22,0 ); cprintf( "Allocation Buffer size : %uK bytes", BUFSIZE/1024 );
1197            if( ( ch = mfr[mfrno].dev[devno].DataSize ) != 0 ) {
1198            locate(23,0 ); cprintf( "Data memory buffer at 4200 ~ %04X", 0x4200 + ch - 1
                );
1199            }
1200
1201            for( redraw = 0; !redraw; )
1202            {
1203                textattr( ( BLUE << 4 ) + WHITE ); clscrn( 24,0, 24,38 );
1204
1205                locate( 24,0 ); cprintf( "Select function ? " );
1206
1207                for(;;)
1208                {
1209                    if( ( ch = getch() ) != 0 )
1210                        break;
1211                    getch();                /* neglect extended code */
1212                }
1213
1214                switch( ch = toupper(ch) )
1215                {
1216                case '1': dos_shell( "" ); redraw = 1; break;
1217                case '2': tmpval = bufsize; bufsize = 0x8000;
```

```
1218                        memset( (void far *)buffer, 0, BUFSIZE );
1219                        load_file();
1220                        bufsize= tmpval;
1221                        redraw = 1; break;
1222                case '3': save_file(); break;
1223                case '4': tmpval = bufsize; bufsize = 0x8000;
1224                        edit_buffer();
1225                        bufsize= tmpval;
1226                        redraw = 1; break;
1227                case '5': set_io_adr(); first = redraw = 1; break;
1228                case '7': disp_buffer(); redraw = 1; break;
1229
1230                case 'M': mfr_select(); redraw = first = 1; break;
1231                case 'T': type_select(); redraw = first = 1; break;
1232    //          case 'E': edit_config(); redraw = 1; break;
1233
1234                case 'R': flash_read(); break;
1235                case 'B': flash_check(); break;
1236                case 'S': flash_erase(); break;
1237                case 'P': flash_program(); break;
1238                case 'V': flash_verify(); break;
1239                case 'L': flash_protect(); break;
1240                case 'A': flash_auto(); break;
1241
1242    //          case '\n':
1243    //          case '\r': redraw = 1; break;       // refresh
1244                }
1245
1246                setport( USERBITS, 0, 0 );
1247
1248                if( ch == 'Q' )
1249                {
1250                    WriteConfig();
1251                    textattr( LIGHTGRAY ); clrscr();
1252                    chdir( oldpath );
1253                    if( buffer ) farfree( (void far *)buffer );
1254                    return( 0 );
1255                }
1256
1257                textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 11,40, 23,79 );
1258            }
1259        }
1260    }
1261
```