```c
#include "all03.h"

// AMD AM29{F,LV,BV}0{1,2,4}0/A/B
//
// MFG ID  AA->555  55->2AA  90->555  X00->id (01)
// DEV ID  AA->555  55->2AA  90->555  X01->id (6E)
// SECERA  AA->555  55->2AA  80->555  AA->555  55->2AA  30->aaaaa
//
// PROG    555,AA  2AA,55  555,A0  aaaaa,dd
// CHPERA  555,AA  2AA,55  555,80  555,AA  2AA,55  555,10
//

//0.4  5: A18       VCC  :36 4.3
//0.5  6: A16       /WE  :35 4.2
//0.6  7: A15       A17  :34 4.1
//0.7  8: A12       A14  :33 4.0
//1.0  9: A7        A13  :32 3.7
//1.1 10: A6        A8   :31 3.6
//1.2 11: A5        A9   :30 3.5 VID=+12V (from VOP) \  SECTOR (UN)PROTECT
//1.3 12: A4        A11  :29 3.4                      +
//1.4 13: A3        /OE  :28 3.3 VID=+12V (from VOP) /  SECTOR PROTECT VERIFY
//1.5 14: A2        A10  :27 3.2
//1.6 15: A1        /CE  :26 3.1
//1.7 16: A0        D7   :25 3.0
//2.0 17: D0        D6   :24 2.7
//2.1 18: D1        D5   :23 2.6
//2.2 19: D2        D4   :22 2.5
//    20: GND       D3   :21 2.4

#define _CE     26
#define _OE     28
#define _WE     35

static char apin[] = { 16,15,14,13,12,11,10,9,31,30,27,29,8,32,33,7,6,34,5 };
static char dpin[] = { 17,18,19,21,22,23,24,25 };

struct DEV {
    char name[20];
    long Size;
    char VCC;
};

struct DEV AMD_devs[] = {
    { "Am29LV010/A/B",  0x20000L, 33 },
    { "Am29LV020/A/B",  0x40000L, 33 },
    { "Am29LV040/A/B",  0x80000L, 33 },
    { "Am29F010",       0x20000L, 50 },
    { "Am29F020",       0x40000L, 50 },
    { "Am29F040",       0x80000L, 50 }
};

struct DEV MXIC_devs[] = {
    { "MX29F010",   0x20000L, 50 },
    { "MX29F020",   0x40000L, 50 },
    { "MX29F040",   0x80000L, 50 }
};

struct {
    char name[20];
    int numdevs;
    struct DEV *dev;
} mfr[] = {
    { "AMD/MMI", sizeof(AMD_devs)/sizeof(struct DEV), AMD_devs },
    { "MXIC", sizeof(MXIC_devs)/sizeof(struct DEV), MXIC_devs }
};

int mfrno = 0, devno = 0;

long DevStart= 0L;
long Counter = 0L;
long lastaddr;

```

```
73  void ShowCounter( long val )
74  {
75      struct text_info ti;
76
77      gettextinfo( &ti );
78
79      textattr( ( BLUE << 4 ) + WHITE );
80      locate( 8,71 ); cprintf( " %05lX ", val );
81
82      textattr( ti.attribute );
83      gotoxy( ti.curx, ti.cury );
84  }
85
86  void power( int state )
87  {
88      int i;
89
90      // we dont need these...
91      setdac( VHHID, 0 );                 // VHH = 0V
92      setdac( VOPID, 0 );                 // VOP = 0V
93
94      setport( OTHERENID, 0, 0 );         // Pin20 = GND
95
96      setport( VHHENCID,0, 0 );           // no VHHC
97      setport( VHHENCID,1, 0 );           // no VHHC
98
99      setport( VCCENID, 0, 0 );           // no VCC
100     setport( VCCENID, 1, 0 );           // no VCC
101
102     for( i = 0; i <= 4; ++i )
103     {
104         setport( VOPENID, i, 0 );       // no VOP
105         setport( VHHENID, i, 0 );       // no VHH
106         setport( TTLID,   i, 0xFF );    // TTL hi on all pins
107     }
108
109     lastaddr = -1L;
110
111     // VCC is pin 32 (programmer socket pin 36)
112
113     if( state )
114     {
115         setdac( VCCID, mfr[mfrno].dev[devno].VCC ); // VCC = 3.3/5.0V
116         waitms( 100 );
117         setpin( 36, VCCENID, 1 );
118     }
119     else
120     {
121         setdac( VCCID, 0 );             // VCC = 0V
122         waitms( 100 );
123         setpin( 36, VCCENID, 0 );
124     }
125 }
126
127 void setaddr( unsigned long addr )
128 {
129     register unsigned char i;
130     register unsigned a, b;
131
132     a = (unsigned)( addr & 0xFFFF );
133     b = (unsigned)( lastaddr & 0xFFFF );
134
135     for( i = 0; i < 16; ++i )       // A0..A15
136     {
137         if( ( a & 1 ) != ( b & 1 ) )
138             setpin( apin[i], TTLID, a & 1 );
139         a >>= 1; b >>= 1;
140     }
141
142     a = (unsigned)( addr >> 16 );
143     b = (unsigned)( lastaddr >> 16 );
144
```

```
145         for( i = 16; i < 19; ++i )      // A16..A18
146         {
147             if( ( a & 1 ) != ( b & 1 ) )
148                 setpin( apin[i], TTLID, a & 1 );
149             a >>= 1; b >>= 1;
150         }
151
152         lastaddr = addr;
153     }
154
155     void f_write( long addr, register unsigned char d )
156     {
157         register int i;
158
159         setaddr( addr );
160
161         for( i = 0; i < 8; ++i )
162         {
163             setpin( dpin[i], TTLID, d & 1 );
164             d >>= 1;
165         }
166
167         setpin( _OE, TTLID, 1 );
168         setpin( _CE, TTLID, 0 );
169         setpin( _WE, TTLID, 0 );
170
171         setpin( _WE, TTLID, 1 );
172         setpin( _CE, TTLID, 1 );
173     }
174
175     void f_prepare( void )
176     {
177         int i;
178
179         for( i = 0; i < 8; ++i )
180             setpin( dpin[i], TTLID, 1 );    // set D0..D7 to 1 for reading
181     }
182
183     unsigned char f_read( void )
184     {
185         register unsigned char d = 0;
186         register int i;
187
188         setpin( _WE, TTLID, 1 );
189         setpin( _CE, TTLID, 0 );
190         setpin( _OE, TTLID, 0 );
191
192         for( i = 7; i >= 0; --i )
193             d = ( d << 1 ) | getpin( dpin[i] );
194
195         setpin( _OE, TTLID, 1 );
196         setpin( _CE, TTLID, 1 );
197
198         return d;
199     }
200
201     int do_erase( void )
202     {
203         unsigned char d, n;
204         int i;
205
206         for( i = 0; i < 2; ++i )
207         {
208             f_write( 0x555L, 0xAA );
209             f_write( 0x2AAL, 0x55 );
210             f_write( 0x555L, 0x80 );
211
212             f_write( 0x555L, 0xAA );
213             f_write( 0x2AAL, 0x55 );
214             f_write( 0x555L, 0x10 );
215
216             f_prepare();
```

```
217
218         d = f_read();
219
220         while( ( ( n = f_read() ) ^ d ) & 0x40 )        // while D6 toggles
221         {
222             if( ( d = n ) & 0x20 )                      // D5=1 : timeout
223             {
224                 if( ( f_read() ^ f_read() ) & 0x40 )    // still toggling
225                 {
226                     f_write( 0L, 0xF0 );                // reset device
227                     f_prepare();
228                     waitms( 10 );
229                     return 0;                           // failure
230                 }
231                 break;                                  // just in time
232             }
233         }
234     }
235     return 1;
236 }
237
238 int do_sector_erase( int block )
239 {
240     unsigned char d, n;
241     int i;
242
243     for( i = 0; i < 2; ++i )
244     {
245         f_write( 0x555L, 0xAA );
246         f_write( 0x2AAL, 0x55 );
247         f_write( 0x555L, 0x80 );
248
249         f_write( 0x555L, 0xAA );
250         f_write( 0x2AAL, 0x55 );
251
252         f_write( block * (bufsize/8), 0x30 );
253
254         f_prepare();
255
256         d = f_read();
257
258         while( ( ( n = f_read() ) ^ d ) & 0x40 )        // while D6 toggles
259         {
260             if( ( d = n ) & 0x20 )                      // D5=1 : timeout
261             {
262                 if( ( f_read() ^ f_read() ) & 0x40 )    // still toggling
263                 {
264                     f_write( 0L, 0xF0 );                // reset device
265                     f_prepare();
266                     waitms( 10 );
267                     return 0;                           // failure
268                 }
269                 break;                                  // just in time
270             }
271         }
272     }
273     return 1;
274 }
275
276 int do_program( long addr, unsigned char val )
277 {
278     unsigned char d, n;
279
280     f_write( 0x555L, 0xAA );
281     f_write( 0x2AAL, 0x55 );
282     f_write( 0x555L, 0xA0 );
283
284     f_write( addr, val );
285
286     f_prepare();
287
288     d = f_read();
```

```
289
290         while( ( ( n = f_read() ) ^ d ) & 0x40 )        // while D6 toggles
291         {
292             if( ( d = n ) & 0x20 )                      // D5=1 : timeout
293             {
294                 if( ( f_read() ^ f_read() ) & 0x40 )    // still toggling
295                 {
296                     f_write( 0L, 0xF0 );                // reset device
297                     f_prepare();
298                     waitms( 10 );
299                     return 0;                           // failure
300                 }
301                 n = f_read();
302                 break;                                  // just in time
303             }
304         }
305         return (n == val);
306     }
307
308     void ShowProtect( void )
309     {
310         int i,v;
311
312         // A16..A14 = sector#
313         // A13..A10 = x -> 0
314         // A9       = Vid (12V)
315         // A8 .. A7 = x -> 0
316         // A6       = 0
317         // A5 .. A2 = x -> 0
318         // A1 .. A0 = 2
319
320         textattr( ( LIGHTGRAY << 4 ) | WHITE );
321         locate( 24, 41 ); cprintf( "*Protect bit[7..0] :" );
322
323         for( i = 7; i >= 0; --i )
324         {
325             setaddr( 0x00202L | 0x04000L * i ); // A9=1, A1=1, all others=0
326
327             setpin( 30, VOPENID, 1 );           // A9 = Vid (12V)
328             v = f_read();
329             setpin( 30, VOPENID, 0 );
330
331             cprintf( "%c", !v ? '0':'1' );
332         }
333     }
334
335     void ShowDevice( void )
336     {
337         int mfr,dev;
338
339         // A16..A14 = x
340         // A13..A10 = x -> 0
341         // A9       = Vid (12V)
342         // A8 .. A7 = x -> 0
343         // A6       = 0
344         // A5 .. A2 = x -> 0
345         // A1 .. A0 = 0:MFGID, 1:DEVID
346
347         power(1);
348     #if(0)
349
350         setaddr( 0x00200L );            // A9=1, all others=0
351
352         setpin( apin[9], VOPENID, 1 );  // A9 = Vid (12V)
353         mfr = f_read();
354         setpin( apin[9], VOPENID, 0 );
355
356         setpin( apin[0], TTLID, 1 );    // A0=1
357         setpin( apin[9], VOPENID, 1 );  // A9 = Vid (12V)
358         dev = f_read();
359         setpin( apin[9], VOPENID, 0 );
360
```

```
#else

    f_write( 0x555L, 0xAA );
    f_write( 0x2AAL, 0x55 );
    f_write( 0x555L, 0x90 );
    f_prepare();
    setaddr( 0L ); mfr = f_read();

    f_write( 0x555L, 0xAA );
    f_write( 0x2AAL, 0x55 );
    f_write( 0x555L, 0x90 );
    f_prepare();
    setaddr( 1L ); dev = f_read();

#endif
    power(0);

    textattr( ( BLUE << 4 ) | WHITE );
    locate( 2, 40 ); cprintf( "*MFR/DEV : %02X/%02X", mfr, dev );
}

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

    bufsize = mfr[mfrno].dev[devno].Size;
    BufEnd = bufsize - 1;

    textattr( ( BLUE << 4 ) | WHITE );

    locate( 0,40 ); cprintf( "*Mfr.: %s", mfr[mfrno].name );
    locate( 1,40 ); cprintf( "*TYPE: %s", mfr[mfrno].dev[devno].name );

    locate( 0,62 ); cprintf( "*BLANK BIT: 1" );
    locate( 1,62 ); cprintf( "*PGMSPEED: INTL" );
    locate( 2,62 ); cprintf( "*VCP :%3.1fV", 0.1 * mfr[mfrno].dev[devno].VCC );

    locate( 6,41 ); cprintf( "       end   addr.: %05lX", BufEnd );

    textattr( ( CYAN << 4 ) | WHITE );
}

int flash_check( void )
{
    int done;
    long addr;

    textattr( ( CYAN << 4 ) | WHITE );
    _window( 11,40, 23,79 );

    textattr( ( BLUE << 4 ) | WHITE );
    locate( 11, 45 ); cprintf( "   BLANK CHECK device:" );

    for(;;)
    {
        textattr( ( CYAN << 4 ) | WHITE );
        locate( 12, 41 ); cprintf( "Ready to check (Y/<CR>)? " );

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

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

```
433                 return;
434
435             case 'y':
436             case 'Y':
437                 done = 1;
438             }
439         }
440
441         clscrn( 13,41, 22,78 );
442
443         power(1);
444         setport( USERBITS, 0, 0 );
445
446         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
447         locate( 13, 41 ); cprintf( "Blank checking now... " );
448
449         ShowCounter( BufStart );
450         for( addr = BufStart; addr <= BufEnd; ++addr )
451         {
452             if( ( addr & 0xFF ) == 0 )
453                 ShowCounter( addr );
454
455             setaddr( addr );
456             if( ( done = f_read() ) != 0xFF )
457                 break;
458         }
459
460         textattr( ( CYAN << 4 ) | WHITE );
461         locate( 13, 41 ); cprintf( "Blank checking now... " );
462
463         locate( 14, 41 );
464         if( addr > BufEnd )
465         {
466             ShowCounter( BufEnd );
467             putchar( 7 );
468             cprintf( " OK !" );
469             setport( USERBITS, 0, 8 );
470         }
471         else
472         {
473             errbeep();
474             textattr( ( RED << 4 ) | WHITE );
475             cprintf( "Blank check error at %05lX (%02X)", addr, done );
476             textattr( ( CYAN << 4 ) | WHITE );
477         }
478
479         power(0);
480     }
481 }
482
483 void flash_program( void )
484 {
485     int done;
486     long addr;
487     FILE *f;
488
489     textattr( ( CYAN << 4 ) | WHITE );
490     _window( 11,40, 23,79 );
491
492     textattr( ( BLUE << 4 ) | WHITE );
493     locate( 11, 45 ); cprintf( "   PROGRAM :" );
494
495     for(;;)
496     {
497         textattr( ( CYAN << 4 ) | WHITE );
498         locate( 12, 41 ); cprintf( "Ready to program (Y/<CR>)? " );
499
500         for( done = 0; !done; )
501         {
502             switch( getch() )
503             {
504             case 0:
```

```
505                 getch();
506                 break;
507
508             case '\n':
509             case '\r':
510             case 0x1B:
511                 return;
512
513             case 'y':
514             case 'Y':
515                 done = 1;
516             }
517         }
518
519         clscrn( 13,41, 22,78 );
520
521         setport( USERBITS, 0, 0 );
522         power(1);
523
524         if( !buffer && ( f = fopen( buffile, "rb" ) ) == NULL )
525         {
526             errbeep();
527             textattr( ( RED << 4 ) | WHITE );
528             locate( 13, 41 ); cprintf( "File open error !" );
529             textattr( ( CYAN << 4 ) | WHITE );
530             delay(1000);
531             break;
532         }
533
534         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
535         locate( 13, 41 ); cprintf( "Programming now... " );
536
537         ShowCounter( BufStart );
538         for( addr = BufStart; addr <= BufEnd; ++addr )
539         {
540             if( ( addr & 0xFF ) == 0 )
541                 ShowCounter( addr );
542
543             if( !do_program( addr, ( buffer ? buffer[addr] : getc(f) ) ) )
544                 break;
545         }
546
547         if( !buffer ) fclose( f );
548
549         textattr( ( CYAN << 4 ) | WHITE );
550         locate( 13, 41 ); cprintf( "Programming now... " );
551
552         locate( 14, 41 );
553         if( addr > BufEnd )
554         {
555             ShowCounter( BufEnd );
556             putchar( 7 );
557             setport( USERBITS, 0, 8 );
558             cprintf( " OK !" );
559         }
560         else
561         {
562             errbeep();
563             textattr( ( RED << 4 ) | WHITE );
564             cprintf( "Program error ! at %05lX", addr );
565             textattr( ( CYAN << 4 ) | WHITE );
566         }
567
568         power(0);
569     }
570 }
571
572 void flash_read( void )
573 {
574     int done;
575     long addr;
576     unsigned char val;
```

```
577         FILE *f;
578
579         textattr( ( CYAN << 4 ) | WHITE );
580         _window( 11,40, 23,79 );
581
582         textattr( ( BLUE << 4 ) | WHITE );
583         locate( 11, 45 ); cprintf( "   READ to buffer :" );
584
585         for(;;)
586         {
587             textattr( ( CYAN << 4 ) | WHITE );
588             locate( 12, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
589
590             for( done = 0; !done; )
591             {
592                 switch( getch() )
593                 {
594                 case 0:
595                     getch();
596                     break;
597
598                 case '\n':
599                 case '\r':
600                 case 0x1B:
601                     return;
602
603                 case 'y':
604                 case 'Y':
605                     done = 1;
606
607                 case 'e':
608                 case 'E':
609                 case 'o':
610                 case 'O':
611                     done = 1;
612                 }
613             }
614
615             clscrn( 13,41, 22,78 );
616
617             setport( USERBITS, 0, 0 );
618             power(1);
619
620             if( !buffer && ( f = fopen( buffile, "wb" ) ) == NULL )
621             {
622                 errbeep();
623                 textattr( ( RED << 4 ) | WHITE );
624                 locate( 13, 41 ); cprintf( "File open error !" );
625                 textattr( ( CYAN << 4 ) | WHITE );
626                 delay(1000);
627                 break;
628             }
629
630             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
631             locate( 13, 41 ); cprintf( "Reading now... " );
632
633             Chks = 0;
634
635             ShowCounter( BufStart );
636             for( addr = BufStart; addr <= BufEnd; ++addr )
637             {
638                 if( ( addr & 0xFF ) == 0 )
639                     ShowCounter( addr );
640
641                 setaddr( addr );
642
643                 Chks += ( val = f_read() );
644
645                 if( buffer ) buffer[ addr ] = val;
646                 else putc( val, f );
647             }
648             ShowCounter( BufEnd );
```

```
649
650             textattr( ( BLUE << 4 ) | WHITE );
651             locate( 7,41 ); cprintf( "       Check Sum  :  %04X", Chks );
652
653             if( !buffer ) fclose( f );
654
655             textattr( ( CYAN << 4 ) | WHITE );
656             locate( 13, 41 ); cprintf( "Reading now... " );
657
658             locate( 14, 41 );
659             putchar( 7 );
660             cprintf( " OK !" );
661
662             power(0);
663         }
664     }
665
666     void flash_verify( void )
667     {
668         int done;
669         long addr;
670         FILE *f;
671
672         textattr( ( CYAN << 4 ) | WHITE );
673         _window( 11,40, 23,79 );
674
675         textattr( ( BLUE << 4 ) | WHITE );
676         locate( 11, 45 ); cprintf( "   VERIFY with buffer :" );
677
678         for(;;)
679         {
680             textattr( ( CYAN << 4 ) | WHITE );
681             locate( 12, 41 ); cprintf( "Ready to verify (Y/Even/Odd/<CR>)? " );
682
683             for( done = 0; !done; )
684             {
685                 switch( getch() )
686                 {
687                 case 0:
688                     getch();
689                     break;
690
691                 case '\n':
692                 case '\r':
693                 case 0x1B:
694                     return;
695
696                 case 'y':
697                 case 'Y':
698                     done = 1;
699
700                 case 'e':
701                 case 'E':
702                 case 'o':
703                 case 'O':
704                     done = 1;
705                 }
706             }
707
708             clscrn( 13,41, 22,78 );
709
710             setport( USERBITS, 0, 0 );
711             power(1);
712
713             if( !buffer && ( f = fopen( buffile, "rb" ) ) == NULL )
714             {
715                 errbeep();
716                 textattr( ( RED << 4 ) | WHITE );
717                 locate( 13, 41 ); cprintf( "File open error !" );
718                 textattr( ( CYAN << 4 ) | WHITE );
719                 delay(1000);
720                 break;
```

```
721             }
722
723             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
724             locate( 13, 41 ); cprintf( "Verifying now... " );
725
726             ShowCounter( BufStart );
727             for( addr = BufStart; addr <= BufEnd; ++addr )
728             {
729                 if( ( addr & 0xFF ) == 0 )
730                     ShowCounter( addr );
731
732                 setaddr( addr );
733
734                 if( ( ( buffer ? buffer[addr] : getc(f) ) & 0xFF ) != f_read() )
735                     break;
736             }
737
738             if( !buffer ) fclose( f );
739
740             textattr( ( CYAN << 4 ) | WHITE );
741             locate( 13, 41 ); cprintf( "Verifying now... " );
742
743             locate( 14, 41 );
744             if( addr > BufEnd )
745             {
746                 ShowCounter( BufEnd );
747                 putchar( 7 );
748                 cprintf( " OK !" );
749                 setport( USERBITS, 0, 8 );
750             }
751             else
752             {
753                 errbeep();
754                 textattr( ( RED << 4 ) | WHITE );
755                 cprintf( " VERIFY ERROR ! at %05lX", addr );
756                 textattr( ( CYAN << 4 ) | WHITE );
757             }
758
759             power(0);
760         }
761     }
762
763     void flash_erase( void )
764     {
765         int i, done, sel=8, er=0xFF;
766
767         textattr( ( CYAN << 4 ) | WHITE );
768         _window( 10,40, 23,79 );
769
770         textattr( ( BLUE << 4 ) | WHITE );
771         locate( 10, 45 ); cprintf( "  EEPROM Erase:" );
772
773         for(;;)
774         {
775             textattr( ( CYAN << 4 ) | WHITE );
776             locate( 11, 41 ); cprintf( "Ready to erase (Y/<CR>)? " );
777
778             for( done = 0; !done; )
779             {
780                 for( i = 0; i < 9; ++i )
781                 {
782                     locate( 12+i, 45 );
783                     if( i == sel )
784                     {
785                         textattr( ( RED << 4 ) | WHITE );
786                         cprintf( "> " );
787                     }
788                     else
789                     {
790                         textattr( ( CYAN << 4 ) | WHITE );
791                         cprintf( "  " );
792                     }
```

```
793
794                 if( i == 8 )
795                     cprintf( "[ 9 ] ALL            %s",
796                         ( er == 0xFF ) ? "erase":"     " );
797                 else
798                     cprintf( "[ %d ] %05lX - %05lX  %s",
799                         i+1, i * (bufsize/8), (i+1) * (bufsize/8) - 1,
800                         ( er & (1<<i) ) ? "erase":"     " );
801             }
802
803             textattr( ( CYAN << 4 ) | WHITE );
804             locate( 21, 41 ); cprintf( "press space to select erase status" );
805             locate( 22, 41 ); cprintf( "press <CR> to go to main menu" );
806
807             switch( i = getch() )
808             {
809             case 0:
810                 switch( getch() )
811                 {
812                 case 72:    // UP
813                     if( sel > 0 )
814                         --sel;
815                     break;
816
817                 case 80:    // DOWN
818                     if( sel < 8 )
819                         ++sel;
820                     break;
821                 }
822                 break;
823
824             case '\n':
825             case '\r':
826             case 0x1B:
827                 return;
828
829             case 0x20:
830                 if( sel == 8 )
831                 {
832                     if( er == 0xFF )
833                         er = 0;
834                     else
835                         er = 0xFF;
836                 }
837                 else
838                     er ^= ( 1 << sel );
839                 break;
840
841             case 'y':
842             case 'Y':
843                 done = 1;
844
845             default:
846                 if( i >= '1' && i <= '8' )
847                     er ^= ( 1 << (i-'1') );
848                 else if( i == '9' )
849                 {
850                     if( er == 0xFF )
851                         er = 0;
852                     else
853                         er = 0xFF;
854                 }
855                 break;
856             }
857         }
858
859         if( !er ) continue;          // nothing to do
860
861         clscrn( 12,41, 22,78 );
862
863         setport( USERBITS, 0, 0 );
864         power(1);
```

```
865
866             textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
867             locate( 12, 41 ); cprintf( "Erase now... " );
868
869             if( er == 0xFF )
870                 done = do_erase();
871             else
872             {
873                 textattr( ( CYAN << 4 ) | WHITE );
874
875                 for( i = 0; i < 8; ++i )
876                 {
877                     if( er & ( 1<<i ) )
878                     {
879                         locate( 20, 41 ); cprintf( "Block : %d", i+1 );
880
881                         if( ( done = do_sector_erase(i) ) == 0 )
882                             break;
883                     }
884                 }
885             }
886
887             textattr( ( CYAN << 4 ) | WHITE );
888             locate( 12, 41 ); cprintf( "Erase now... " );
889
890             locate( 13, 41 );
891             if( done )
892             {
893                 putchar( 7 );
894                 cprintf( " OK !" );
895                 setport( USERBITS, 0, 8 );
896             }
897             else
898             {
899                 errbeep();
900                 textattr( ( RED << 4 ) | WHITE );
901                 cprintf( " ERROR ! " );
902                 textattr( ( CYAN << 4 ) | WHITE );
903
904                 locate( 21, 41 ); cprintf( "press any key to continue" );
905                 if( getch() == 0 ) getch();
906             }
907
908             clscrn( 11,41, 22,78 );
909
910             power(0);
911         }
912     }
913
914     void flash_auto( void )
915     {
916         int done, l;
917         long addr;
918         FILE *f;
919
920         textattr( ( CYAN << 4 ) | WHITE );
921         _window( 11,40, 23,79 );
922
923         textattr( ( BLUE << 4 ) | WHITE );
924         locate( 11, 45 ); cprintf( "   AUTO :" );
925
926         for(;;)
927         {
928             textattr( ( CYAN << 4 ) | WHITE );
929             locate( 12, 41 ); cprintf( "Ready to start (Y/Even/Odd/<CR>)? " );
930
931             for( done = 0; !done; )
932             {
933                 switch( getch() )
934                 {
935                 case 0:
936                     getch();
```

```
937                 break;
938
939             case '\n':
940             case '\r':
941             case 0x1B:
942                 return;
943
944             case 'y':
945             case 'Y':
946                 done = 1;
947
948             case 'e':
949             case 'E':
950             case 'o':
951             case 'O':
952                 done = 1;
953             }
954         }
955
956         clscrn( 13,41, 22,78 );
957
958         setport( USERBITS, 0, 0 );
959         power(1);
960
961         l = 13;
962
963         // BLANK CHECK
964
965         textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
966         locate( l, 41 ); cprintf( "Blank checking now... " );
967
968         ShowCounter( BufStart );
969         for( addr = BufStart; addr <= BufEnd; ++addr )
970         {
971             if( ( addr & 0xFF ) == 0 )
972                 ShowCounter( addr );
973
974             setaddr( addr );
975             if( f_read() != 0xFF )
976                 break;
977         }
978
979         textattr( ( CYAN << 4 ) | WHITE );
980         locate( l++, 41 ); cprintf( "Blank checking now... " );
981
982         locate( l++, 41 );
983         if( addr > BufEnd )
984         {
985             ShowCounter( BufEnd );
986             cprintf( " OK !" );
987         }
988         else
989         {
990             errbeep();
991             textattr( ( RED << 4 ) | WHITE );
992             cprintf( "Blank check error at %05lX", addr );
993             textattr( ( CYAN << 4 ) | WHITE );
994
995         // ERASE
996
997             l -= 2;
998
999             clscrn( l, 41, l+1, 78 );
1000
1001            textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1002            locate( l, 41 ); cprintf( "Erase now... " );
1003
1004            done = do_erase();
1005
1006            textattr( ( CYAN << 4 ) | WHITE );
1007            locate( l++, 41 ); cprintf( "Erase now... " );
1008
```

```
1009            locate( l++, 41 );
1010            if( done )
1011                cprintf( " OK !" );
1012            else
1013            {
1014                errbeep();
1015                textattr( ( RED << 4 ) | WHITE );
1016                cprintf( " ERROR" );
1017                textattr( ( CYAN << 4 ) | WHITE );
1018
1019                power(0);
1020                continue;
1021            }
1022        }
1023
1024        // PROGRAM
1025
1026        if( !buffer && ( f = fopen( buffile, "rb" ) ) == NULL )
1027        {
1028            errbeep();
1029            textattr( ( RED << 4 ) | WHITE );
1030            locate( 13, 41 ); cprintf( "File open error !" );
1031            textattr( ( CYAN << 4 ) | WHITE );
1032            delay(1000);
1033            break;
1034        }
1035
1036        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1037        locate( l, 41 ); cprintf( "Programming now... " );
1038
1039        ShowCounter( BufStart );
1040        for( addr = BufStart; addr <= BufEnd; ++addr )
1041        {
1042            if( ( addr & 0xFF ) == 0 )
1043                ShowCounter( addr );
1044
1045            if( !do_program( addr, ( buffer ? buffer[addr] : getc(f) ) ) )
1046                break;
1047        }
1048
1049        textattr( ( CYAN << 4 ) | WHITE );
1050        locate( l++, 41 ); cprintf( "Programming now... " );
1051
1052        locate( l++, 41 );
1053        if( addr > BufEnd )
1054        {
1055            ShowCounter( BufEnd );
1056            cprintf( " OK !" );
1057        }
1058        else
1059        {
1060            if( !buffer ) fclose( f );
1061
1062            errbeep();
1063            textattr( ( RED << 4 ) | WHITE );
1064            cprintf( "Program error ! at %05lX", addr );
1065            textattr( ( CYAN << 4 ) | WHITE );
1066
1067            power(0);
1068            continue;
1069        }
1070
1071        // VERIFY
1072
1073        textattr( BLINK | ( LIGHTGREEN << 4 ) | WHITE );
1074        locate( l, 41 ); cprintf( "Verifying now... " );
1075
1076        if( !buffer ) rewind( f );
1077
1078        ShowCounter( BufStart );
1079        for( addr = BufStart; addr <= BufEnd; ++addr )
1080        {
```

```
1081                if( ( addr & 0xFF ) == 0 )
1082                    ShowCounter( addr );
1083
1084                setaddr( addr );
1085
1086                if( ( ( buffer ? buffer[addr] : getc(f) ) & 0xFF ) != f_read() )
1087                    break;
1088            }
1089
1090            if( !buffer ) fclose( f );
1091
1092            textattr( ( CYAN << 4 ) | WHITE );
1093            locate( l++, 41 ); cprintf( "Verifying now... " );
1094
1095            locate( l, 41 );
1096            if( addr > BufEnd )
1097            {
1098                ShowCounter( BufEnd );
1099                cprintf( " OK !" );
1100                setport( USERBITS, 0, 8 );
1101                putchar( 7 );
1102            }
1103            else
1104            {
1105                textattr( ( RED << 4 ) | WHITE );
1106                cprintf( " VERIFY ERROR ! at %05lX", addr );
1107                textattr( ( CYAN << 4 ) | WHITE );
1108                errbeep();
1109            }
1110
1111            power(0);
1112        }
1113        power(0);
1114    }
1115
1116    void type_select( void )
1117    {
1118        int done, i;
1119        char no[10];
1120
1121        textattr( ( CYAN << 4 ) | WHITE );
1122        _window( 11,40, 23,79 );
1123
1124        textattr( ( BLUE << 4 ) | WHITE );
1125        locate( 11, 45 ); cprintf( "  TYPE SELECT:" );
1126
1127        textattr( ( CYAN << 4 ) | WHITE );
1128
1129        for( i = 0; i < mfr[mfrno].numdevs; ++i )
1130        {
1131            locate( 12+i, 41 ); cprintf( "%d.%s", i, mfr[mfrno].dev[i].name );
1132        }
1133
1134        locate( 21, 41 ); cprintf( "<CR> back to main menu." );
1135        locate( 22, 41 ); cprintf( "SELECT NUMBER ?" );
1136
1137        for(;;)
1138        {
1139            for( no[0] = done = 0; !done; )
1140            {
1141                locate( 22, 56 ); cprintf( "%s ", no );
1142                locate( 22, 56+strlen(no) );
1143
1144                switch( i = getch() )
1145                {
1146                case 0:
1147                    getch();
1148                    break;
1149
1150                case 8:
1151                    if( ( i = strlen(no) ) != 0 )
1152                        no[i-1] = 0;
```

```
                    break;

                case '\n':
                case '\r':
                    if( strlen( no ) &&
                        ( i = atoi(no) ) >= 0 && i < mfr[mfrno].numdevs )
                    {
                        devno = i;
                        ShowType();
                        return;
                    }
                    break;

                case 0x1B:
                    return;

                default:
                    if( isdigit( i ) )
                        strcat( no, (char*)&i );
                    break;
                }
            }
        }
    }

    void mfr_select( void )
    {
        int done, i;
        char no[10];

        textattr( ( CYAN << 4 ) | WHITE );
        _window( 11,40, 23,79 );

        textattr( ( BLUE << 4 ) | WHITE );
        locate( 11, 45 ); cprintf( "  MFR SELECT:" );

        textattr( ( CYAN << 4 ) | WHITE );

        for( i = 0; i < sizeof(mfr) / sizeof(mfr[0]); ++i )
        {
            locate( 12+i, 41 ); cprintf( "%d.%s", i, mfr[i].name );
        }

        locate( 21, 41 ); cprintf( "<CR> back to main menu." );
        locate( 22, 41 ); cprintf( "SELECT NUMBER ?" );

        for(;;)
        {
            for( no[0] = done = 0; !done; )
            {
                locate( 22, 56 ); cprintf( "%s ", no );
                locate( 22, 56+strlen(no) );

                switch( i = getch() )
                {
                case 0:
                    getch();
                    break;

                case 8:
                    if( ( i = strlen(no) ) != 0 )
                        no[i-1] = 0;
                    break;

                case '\n':
                case '\r':
                    if( strlen( no ) &&
                        ( i = atoi(no) ) >= 0 && i < sizeof(mfr) / sizeof(mfr[0]) )
                    {
                        mfrno = i;
                        if( devno >= mfr[mfrno].numdevs )
                            devno = 0;
```

```
1225                        ShowType();
1226                        return;
1227                    }
1228                    break;
1229
1230                case 0x1B:
1231                    return;
1232
1233                default:
1234                    if( isdigit( i ) )
1235                        strcat( no, (char*)&i );
1236                    break;
1237                }
1238            }
1239        }
1240    }
1241
1242    /*=========================================================*/
1243    int main( void )
1244    {
1245        int ch = 0, redraw, first = 1;
1246
1247        /*---------------------------------------------------------*/
1248        /* main program starts here                                */
1249        /*---------------------------------------------------------*/
1250
1251        getcwd( oldpath, 260 );
1252        strcpy( path, oldpath );
1253
1254        ReadConfig();
1255
1256        for( ;; )
1257        {
1258            if( first )
1259            {
1260                first = 0;
1261
1262                init_hw();
1263                initdacs();
1264                setport( USERBITS, 0, 0 );
1265
1266    #ifdef WIN32
1267                if( !buffer ) buffer = farmalloc( bufsize );
1268                else buffer = farrealloc( buffer, bufsize );
1269    #else
1270                if( !buffer ) buffer = (void huge *)farmalloc( bufsize );
1271                else buffer = (void huge*)farrealloc( (void far*)buffer, bufsize );
1272    #endif
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( "FLASH section V1.00" );
1280
1281            textattr( ( BLUE << 4 ) + WHITE ); clscrn( 0,40, 8,79 );
1282
1283            ShowType();
1284            power(0);
1285
1286            textattr( ( BLUE << 4 ) + WHITE ); _window( 3,40, 9,79 );
1287            locate( 4,41 ); cprintf( "             TARGET ZONE" );
1288            locate( 5,41 ); cprintf( "Buffer start addr.: %05lX (BYTE WIDE)", BufStart );
1289            locate( 6,41 ); cprintf( "       end   addr.: %05lX", BufEnd );
1290            locate( 7,41 ); cprintf( "       Check Sum  :  %04X", Chks );
1291            locate( 8,41 ); cprintf( "Device start addr.: %05lX", DevStart );
1292
1293            _window( 6,69, 9,79 );
1294            locate( 7,71 ); cprintf( "COUNTER" );
1295
1296            ShowCounter( 0L );
```

```
1297
1298            textattr( ( CYAN << 4 ) | WHITE ); clscrn( 3,0, 23,38 );
1299
1300            locate( 3,0 ); cprintf( "-------------  Main Menu  -------------" );
1301            locate( 4,0 ); cprintf( "1. DOS SHELL                           " );
1302            locate( 5,0 ); cprintf( "2. Load BIN or HEX file to buffer      " );
1303            locate( 6,0 ); cprintf( "3. Save buffer to disk                 " );
1304            locate( 7,0 ); cprintf( "4. Edit buffer     7. Display buffer   " );
1305            locate( 8,0 ); cprintf( "5. Change I/O base address            " );
1306            locate( 9,0 ); cprintf( "6. Display loaded file history         " );
1307            locate(10,0 ); cprintf( "9. Modify buffer structure             " );
1308            locate(11,0 ); cprintf( "W. swapping LOW-HI byte in buffer      " );
1309            locate(12,0 ); cprintf( "T. Type select     M. Mfr. select      " );
1310            locate(13,0 ); cprintf( "Z. Target zone                         " );
1311            locate(14,0 ); cprintf( "                                       " );
1312            locate(15,0 ); cprintf( "B. Blank check     D. Display          " );
1313            locate(16,0 ); cprintf( "P. Program         A. Auto(B&P&S&V)    " );
1314            locate(17,0 ); cprintf( "R. Read            V. Verify           " );
1315            locate(18,0 ); cprintf( "C. Compare and display error           " );
1316            locate(19,0 ); cprintf( "E. Erase           S. Data protection  " );
1317            locate(20,0 ); cprintf( "Q. Quit                                " );
1318            locate(21,0 ); cprintf( "---------------------------------------" );
1319            locate(22,0 ); cprintf( "Buffer size      : %ldK bytes", bufsize / 1024 );
1320            locate(23,0 ); cprintf( "Buffer structure : %s", buffile );
1321
1322            ShowProtect();
1323    //      ShowDevice();
1324
1325            for( redraw = 0; !redraw; )
1326            {
1327                textattr( ( BLUE << 4 ) + WHITE ); clscrn( 24,0, 24,38 );
1328
1329                locate( 24,0 ); cprintf( "Select function ? " );
1330
1331                for(;;)
1332                {
1333                    if( ( ch = getch() ) != 0 )
1334                        break;
1335                    getch();                        /* neglect extended code */
1336                }
1337
1338                switch( ch = toupper(ch) )
1339                {
1340                case '1': dos_shell( "" ); redraw = 1; break;
1341                case '2': load_file(); redraw = 1; break;
1342                case '3': save_file(); break;
1343                case '4': edit_buffer(); redraw = 1; break;
1344                case '5': set_io_adr(); redraw = first = 1; break;
1345
1346                case 'M': mfr_select(); redraw = first = 1; break;
1347                case 'T': type_select(); redraw = first = 1; break;
1348
1349                case 'B': flash_check(); break;
1350                case 'E': flash_erase(); break;
1351                case 'P': flash_program(); break;
1352                case 'V': flash_verify(); break;
1353                case 'A': flash_auto(); break;
1354                case 'R': flash_read(); break;
1355
1356                case '\n':
1357                case '\r': redraw = 1; break;   // refresh
1358                }
1359
1360                setport( USERBITS, 0, 0 );
1361
1362                if( ch == 'Q' )
1363                {
1364                    WriteConfig();
1365                    textattr( LIGHTGRAY ); clrscr();
1366                    chdir( oldpath );
1367                    return( 0 );
1368                }
```

```
1369
1370                textattr( ( LIGHTGRAY << 4 ) | YELLOW ); clscrn( 10,40, 23,79 );
1371            }
1372        }
1373    }
1374
```