```c
#include "all03.h"

long start = 0, end = 0, dest = 0;
static char cmd[80], *pos;
static FILE *f = NULL;

void dos_shell( char *cmd )
{
    textattr( LIGHTGRAY ); clrscr();
    printf( "Type EXIT to return to Programmer menu\n" );
    system( cmd );
}

static void displine( long addr )
{
    int i, c;
    unsigned char buf[16];

    if( buffer )
        memcpy( buf, (void far *)(buffer+addr), 16 );
    else
    {
        fseek( f, addr, SEEK_SET );
        fread( buf, 1, 16, f );
    }

    cprintf( "%05lX  ", addr );

    for( i = 0; i < 8; ++i )
        cprintf( "%02X ", buf[ i ] );

    cprintf( "--" );

    for( i = 8; i < 16; ++i )
        cprintf( "%02X ", buf[ i ] );

    for( i = 0; i < 16; ++i )
        putch( isprint( c = buf[ i ] ) ? c : '.' );

    cprintf( "\r\n" );
}

static void dispbuf( void )
{
    int c;

    cprintf( "Press <ESC> to terminate display\n\r\n" );
    delay( 500 );

    while( start <= end )
    {
        if( kbhit() )
        {
            if( ( c = getch() ) == 0x1B )
                break;
            else if( c == 0 )
                getch();
        }
        displine( start ); start += 16;
    }
}

void disp_buffer( void )
{
    textattr( ( CYAN << 4 ) + WHITE ); clrscr();

    start = 0; end = bufsize; dispbuf();

    cprintf( "Press any key to continue" );
    if( getch() == 0 ) getch();
}

```

```
73  static unsigned char xval( char c )
74  {
75      if( ( c = toupper( c ) ) > '9' )
76          return c - 'A' + 10;
77      else
78          return c - '0';
79  }
80
81  static void hexedit( void )
82  {
83      int i, c, loop, changed = 0;
84      long addr, top = start, lasttop = -1L;
85      int ascii = 0, right = 0, curx = 0, cury = 0;
86
87      textattr( ( CYAN << 4 ) + WHITE ); clrscr();
88
89      for( ;; )
90      {
91          if( top != lasttop )
92          {
93              lasttop = top;
94              changed = 0;
95
96              locate( 0,0 );
97              for( addr = top; addr < top + 24*16; addr += 16 )
98                  if( addr < start + bufsize )
99                      displine( addr );
100
101             textattr( ( BLUE << 4 ) + WHITE ); clreol();
102             cprintf( " [ESC]: Back to command prompt, [TAB]: Toggle entry mode,
                [ALT-G] Goto address" );
103             textattr( ( CYAN << 4 ) + WHITE );
104         }
105         else if( changed )
106         {
107             changed = 0;
108             c = !curx ? cury-1 : cury;      // check for wrap
109             locate( c, 0 );
110             displine( top + 16*c );
111         }
112
113         locate( cury, !ascii ? ( (curx<8) ? 7:9 ) + 3*curx + right : 57+curx );
114
115         for( loop = 1; loop; )
116         {
117             switch( c = getch() )
118             {
119             case 0:
120                 switch( getch() )
121                 {
122                 case 34:    // ALT-G
123                     locate( 24,0 );
124                     textattr( ( BLUE << 4 ) + WHITE ); clreol();
125                     cprintf( "Enter Address:" );
126                     cmd[0] = 79; xgets( cmd );
127                     textattr( ( CYAN << 4 ) + WHITE );
128
129                     sscanf( cmd+2, "%X", &addr );
130                     if( addr >= start && addr < start + bufsize )
131                     {
132                         top = addr & 0xFFF00L;
133                         if( top > start + bufsize - 24*16 )
134                             top = start +bufsize - 24*16;
135                         if( top < 0 )
136                             top = 0;
137                         curx = (int)(addr & 0x0F); right = 0;
138                         cury = (int)((addr - top) >> 4);
139                         loop = 0;
140                         lasttop = -1L;
141                     }
142                     break;
143
```

```
144                     case 71:    // HOME
145                         curx = cury = 0;
146                         top = start;
147                         loop = 0;
148                         break;
149
150                     case 79:    // END
151                         curx = 15; cury = 23;
152                         top = start + bufsize - 24*16;
153                         if( top < 0 ) top = 0;
154                         loop = 0;
155                         break;
156
157                     case 75:    // LEFT
158                         if( !ascii && right )
159                         {
160                             right = 0;
161                             loop = 0;
162                             break;
163                         }
164                         right = 1;
165                         if( curx )
166                         {
167                             --curx;
168                             loop = 0;
169                             break;
170                         }
171                         curx = 15;
172                         loop = 0;
173
174                         // fall through
175
176                     case 72:    // UP
177                         if( cury == 0 )
178                         {
179                             if( top > start )
180                             {
181                                 loop = 0;
182                                 top -= 16;
183                             }
184                         }
185                         else
186                         {
187                             loop = 0;
188                             --cury;
189                         }
190                         break;
191
192                     case 77:    // RIGHT
193     Advance:            if( !ascii && !right )
194                         {
195                             right = 1;
196                             loop = 0;
197                             break;
198                         }
199                         right = 0;
200                         if( curx < 15 )
201                         {
202                             ++curx;
203                             loop = 0;
204                             break;
205                         }
206                         curx = 0;
207                         loop = 0;
208
209                         // fall through
210
211                     case 80:    // DOWN
212                         if( cury < 23 )
213                         {
214                             loop = 0;
215                             ++cury;
```

```
216                         }
217                         else if( top < start + bufsize - 24*16 )
218                         {
219                             loop = 0;
220                             top += 16;
221                         }
222                         break;
223
224                     case 73:    // PGUP
225                         if( top >= start + 24*16 )
226                         {
227                             top -= 24*16;
228                             if( top < 0 ) top = 0;
229                             loop = 0;
230                         }
231                         else if( top != start )
232                         {
233                             loop = 0;
234                             top = start;
235                         }
236                         break;
237
238                     case 81:    // PGDN
239                         if( top + 24*16 <= start + bufsize - 24*16 )
240                         {
241                             top += 24*16;
242                             loop = 0;
243                         }
244                         else if( top != start + bufsize - 24*16 )
245                         {
246                             top = start + bufsize - 24*16;
247                             if( top < 0 ) top = 0;
248                             loop = 0;
249                         }
250                         break;
251                     }
252                     break;
253
254                 case 0x1B: locate( 24, 0 ); clreol(); return;
255
256                 case 0x09: ascii ^= 1; loop = 0; break;
257
258                 default:
259                     addr = top + 16*cury + curx;
260
261                     if( !ascii )
262                     {
263                         if( isxdigit( c ) )
264                         {
265                             loop = 0;
266                             changed = 1;
267                             if( !right )
268                             {
269                                 right = 1;
270                                 if( buffer )
271                                     buffer[addr] = ( buffer[addr] & 0x0F ) | ( xval(c)
                                     << 4 );
272                                 else
273                                 {
274                                     fseek( f, addr, SEEK_SET ); i = getc( f );
275                                     i = ( i & 0x0F ) | ( xval(c) << 4 );
276                                     fseek( f, addr, SEEK_SET ); putc( i, f );
277                                 }
278                             }
279                             else
280                             {
281                                 if( buffer )
282                                     buffer[addr] = ( buffer[addr] & 0xF0 ) | xval(c);
283                                 else
284                                 {
285                                     fseek( f, addr, SEEK_SET ); i = getc( f );
286                                     i = ( i & 0xF0 ) | xval(c);
```

```
287                                 fseek( f, addr, SEEK_SET ); putc( i, f );
288                             }
289                             goto Advance;
290                         }
291                     }
292                 }
293                 else
294                 {
295                     changed = 1;
296                     loop = 0;
297                     if( buffer )
298                         buffer[addr] = c;
299                     else
300                     {
301                         fseek( f, addr, SEEK_SET ); putc( c, f );
302                     }
303                     goto Advance;
304                 }
305                 break;
306             }
307         }
308     }
309 }
310
311 static void split( void )
312 {
313     end = dest = -1L;
314
315     if( ( pos = strtok( pos, ", \t\n\r" ) ) != NULL )
316         sscanf( pos, "%X", &start );
317     if( ( pos = strtok( NULL, ", \t\n\r" ) ) != NULL )
318         sscanf( pos, "%X", &end );
319     if( ( pos = strtok( NULL, ", \t\n\r" ) ) != NULL )
320         sscanf( pos, "%X", &dest );
321
322     start &= 0xFFFF0L;
323     if( end == -1L ) end = start + 0x7F;
324 }
325
326 void edit_buffer( void )
327 {
328     int redraw, i, l;
329     unsigned char str[16];
330     unsigned Ck;
331     long x;
332
333     if( !buffer && ( f = fopen( buffile, "rb+" ) ) == NULL )
334         return;
335
336     textattr( ( CYAN << 4 ) + WHITE ); clrscr();
337
338     for(;;)
339     {
340         cprintf( "\n\r            EDITING COMMAND SUMMARY\n\r" );
341         cprintf( "D [start],[end]         <RETURN> : DUMP\n\r" );
342         cprintf( "E start                 <RETURN> : EDIT\n\r" );
343         cprintf( "M start,end,destination <RETURN> : MOVE BLOCK\n\r" );
344         cprintf( "F start,end,data        <RETURN> : FILL BLOCK\n\r" );
345         cprintf( "P start,end             <RETURN> : PRINT BLOCK\n\r" );
346         cprintf( "C start,end             <RETURN> : CHECK SUM\n\r" );
347         cprintf( "S start,end,ASCII data  <RETURN> : ASCII SEARCH MAX. 15
            characters\n\r" );
348         cprintf( "B start,end,BINARY data <RETURN> : BINARY SEARCH MAX. 7 BYTEs\n\r"
            );
349         cprintf( ". filename [args]*      <RETURN> : SHELL\n\r" );
350         cprintf( "?                       <RETURN> : HELP\n\r" );
351         cprintf( "Q                       <RETURN> : QUIT\n\r" );
352         cprintf( "=========================================\n\r" );
353
354         for( redraw = 0; !redraw; )
355         {
356             cprintf( "\r\n\==" ); cmd[0] = 79; xgets( cmd ); cprintf( "\n\r" );
```

```
357
358             if( cmd[1] == 0 )
359                 continue;
360
361             i = toupper( cmd[2] );
362
363             for( pos = cmd+3; isspace(*pos); ++pos );
364
365             switch( i )
366             {
367             case 'Q':   if( !buffer )
368                         {
369                             fseek( f, bufsize, SEEK_SET );
370                             fclose( f );
371                         }
372                         return;
373             case '?':   redraw = 1; break;
374             case '.':   system( pos ); break;
375             case 'D':   split(); if( end == -1L ) end = start + 0x7F;
376                         dispbuf();
377                         break;
378             case 'E':   start = 0L; split(); hexedit(); break;
379             case 'M':   split();
380                         if( start != -1L && end != -1L && dest != -1L )
381                         {
382                             if( buffer )
383                             {
384                                 memcpy( (char far *)(buffer+dest),
385                                         (char far *)(buffer+start),
386                                         (unsigned)(end-start+1) );
387                             }
388                             else
389                             {
390                                 for( x = 0; x <= end-start; ++x )
391                                 {
392                                     fseek( f, start+x, SEEK_SET ); i = getc(f);
393                                     fseek( f, dest+x, SEEK_SET ); putc(i,f);
394                                 }
395                             }
396                         }
397                         break;
398             case 'F':   split();
399                         if( start != -1L && end != -1L && dest != -1L )
400                         {
401                             if( buffer )
402                             {
403                                 memset( (char far *)(buffer+start),
404                                         (unsigned char)(dest & 0xFF),
405                                         (unsigned)(end-start+1) );
406                             }
407                             else
408                             {
409                                 dest &= 0xFF;
410                                 fseek( f, start+x, SEEK_SET );
411                                 for( x = 0; x <= end-start; ++x )
412                                     putc( dest, f );
413                             }
414                         }
415                         break;
416             case 'C':   split();
417                         if( start != -1L && end != -1L )
418                         {
419                             Ck = 0;
420                             if( !buffer ) fseek( f, start+x, SEEK_SET );
421                             for( x = start; x <= end; ++x )
422                                 Ck += ( buffer ? buffer[x] : getc(f) );
423                             cprintf( "%04X", Ck );
424                         }
425                         break;
426             case 'S':   split();
427                         if( start != -1L && end != -1L && pos && buffer )
428                         {
```

```
429                             l = strlen( pos );
430                             cprintf( "%05lX-%05lX: '%s'\n\r", start, end, pos );
431                             for( x = start; x <= end-l+1; ++x )
432                             {
433                                 if( !strncmpi( (char far *)(buffer+x), pos, l ) )
434                                     cprintf( "%05lX\n\r", x );
435                             }
436                         }
437                         break;
438             case 'B':   split();
439                         if( start != -1L && end != -1L && dest != -1 && buffer )
440                         {
441                             str[0] = dest; l = 1;
442                             while( ( pos = strtok( NULL, ", \t\n\r" ) ) != NULL )
443                                 sscanf( pos, "%x", &str[l++] );
444                             cprintf( "%05lX-%05lX: ", start, end );
445                             for( i = 0; i < l; ++i )
446                                 cprintf( "%02X ", str[i] );
447                             cprintf( "\n\r" );
448                             for( x = start; x <= end-l+1; ++x )
449                             {
450                                 if( !memcmp( (char far *)(buffer+x), str, l ) )
451                                     cprintf( "%05lX\n\r", x );
452                             }
453                         }
454                         break;
455             }
456         }
457     }
458 }
459
```