そして〝DATE.REL〟ライブラリを使用する場合は，**リスト5.17**のように，プログラムの先頭で〝DATE.H〟をインクルードします．

これでいちいち関数の使用宣言を書く必要はなくなりました．また，各関数に渡すパラメータや戻り値のデータ型などをヘッダーファイルに注釈の形でメモしておくと，あとで忘れたときの参考になります．

```
#include <stdio.h>
#include <date.h>          ......... DATE.H をインクルードする

VOID main()
{
    int  y, m, d;

    y = year();            ......... year() が宣言なしに使用できる
    m = (int)month();      ......... month() 〃
    d = (int)day();        ......... day() 〃
    printf("%Ø4d/%Ø2d/%Ø2d¥n", y, m, d);
}
```

リスト5.17　DATE.Hの使用例

Appendix

特製ライブラリ・ソースプログラム集

2章で解説する「特製ライブラリ」を利用するに必要なプログラムを掲載します．掲載するプログラムは次のとおりです。

なお，バッチプログラムについては，MSX-DOSを使っている人は「MSX-DOS用バッチプログラム」を，MSX-DOS 2を使っている人は「MSX-DOS 2用バッチプログラム」を入力してください．

・特製ライブラリ本体

MSXALIB.MAC…… アセンブラ記述ライブラリ(MSXALIB.REL)のソース
MSXCLIB.C …… MSX-C記述ライブラリ（MSXCLIB.REL）のソース

・ヘッダーファイル

MSXALIB.H …… MSXALIB用ヘッダーファイル
MSXCLIB.H …… MSXCLIB用ヘッダーファイル

・MSX-DOS用バッチプログラム

AMAKE.BAT…… MSXALIB.RELを作成するバッチプログラム
CMAKE.BAT…… MSXCLIB.RELを作成するバッチプログラム
ACHG.BAT …… MSXALIB.RELの一部を更新するバッチプログラム
CCHG.BAT …… MSXCLIB.RELの一部を更新するバッチプログラム

・MSX-DOS 2用バッチプログラム

AMAKE.BAT…… MSXALIB.RELを作成するバッチプログラム
CMAKE.BAT…… MSXCLIB.RELを作成するバッチプログラム
ACHG.BAT …… MSXALIB.RELの一部を更新するバッチプログラム
CCHG.BAT …… MSXCLIB.RELの一部を更新するバッチプログラム

特製ライブラリ本体

リスト　MSXALIB.MAC

```
 Ø:          cseg
 1:
 2: ;--- dummy ---
 3:          MODULE  DMY1
 4:          ENDMODULE
 5:
 6: ;--- GINIT ---
 7:          MODULE  GINIT
 8: EXPTBL   equ     Øfcc1h          ;slot address of BIOS ROM
 9: GINIT@::
1Ø:          ld      hl,6
11:          ld      a,(EXPTBL)
12:          call    ØØØch
13:          ld      (RDPORT@),a
14:          ld      hl,7
15:          ld      a,(EXPTBL)
16:          call    ØØØch
17:          ei
18:          ld      (WRPORT@),a
19:          ret
2Ø:          dseg
21: RDPORT@::
22:          db      98h     ;This initial value is for ill-behaviored
23: WRPORT@::
24:          db      98h     ;programs
25:          cseg
26:          ENDMODULE
27:
28: ;--- CALBIO ---
29:          MODULE  CALBIO
3Ø: CALSLT   equ     ØØ1ch
31: EXPTBL   equ     Øfcc1h          ;slot address of BIOS ROM
32: CALBIO@::
33:          push    hl
34:          pop     ix
35:          push    de
36:          ex      de,hl
37:          ld      iy,Ø
38:          add     iy,sp
39:          di
4Ø:          ld      sp,hl
41:          pop     af
42:          pop     bc
43:          pop     de
44:          pop     hl
45:          ei
```



```
46:          ld      sp,iy
47:          push    iy
48:          ld      iy,(EXPTBL-1)
49:          call    CALSLT
5Ø:          pop     iy
51:          exx
52:          pop     hl
53:          ld      bc,8
54:          add     hl,bc
55:          di
56:          ld      sp,hl
57:          exx
58:          push    hl
59:          push    de
6Ø:          push    bc
61:          push    af
62:          ei
63:          ld      sp,iy
64:          pop     af
65:          ret
66:          ENDMODULE
67:
68: ;--- SETADR ---
69: ;low level routine for SETRD and SETWRT
7Ø: ;entry  HL..read/write address
71:          MODULE  SETADR
72: ACPAGE   equ     Øfaf6h
73: EXBRSA   equ     Øfaf8h
74: SCRMOD   equ     Øfcafh
75: SETADR::
76:          ld      c,a
77:          inc     c
78:          ld      a,(EXBRSA)
79:          or      a               ;MSX1?
8Ø:          jr      z,SETAD3        ;yes
81:          ld      a,(SCRMOD)
82:          cp      5
83:          jr      c,SETAD3
84:          ld      b,h             ;extract the high address
85:          cp      7
86:          ld      a,(ACPAGE)
87:          jr      c,SETAD1
88:          sla     b
89:          rla
9Ø:          jr      SETAD2
91: SETAD1:
92:          sla     b
93: SETAD2:
94:          sla     b
95:          rla
96:          di
97:          out     (c),a           ;set data to be written
98:          ld      a,14+8Øh        ;set it to register #14
```

```
 99:          out     (c),a
1ØØ: SETAD3:
1Ø1:          di
1Ø2:          out     (c),l          ;set the low address
1Ø3:          ld      a,h            ;prepare to set the middle address
1Ø4:          and     3fh
1Ø5:          ret
1Ø6:          ENDMODULE
1Ø7:
1Ø8: ;--- SETWRT ---
1Ø9: ;sets up the VDP for write
11Ø:          MODULE  SETWRT
111:          extrn   SETADR
112:          extrn   WRPORT@
113: SETWRT@::
114:          ld      a,(WRPORT@)
115:          call    SETADR
116:          or      Ø1ØØØØØØb      ;set the low address with write bit
117:          out     (c),a
118:          dec     c
119:          ei
12Ø:          ret
121:          ENDMODULE
122:
123: ;--- SETRD ---
124: ;sets up the VDP for read
125:          MODULE  SETRD
126:          extrn   SETADR
127:          extrn   RDPORT@
128: SETRD@::
129:          ld      a,(RDPORT@)
13Ø:          call    SETADR
131:          out     (c),a
132:          dec     c
133:          ei
134:          ret
135:          ENDMODULE
136:
137: ;--- VPOKE ---
138: ;writes to the VRAM
139:          MODULE  VPOKE
14Ø:          extrn   SETWRT@
141: VPOKE@::
142:          call    SETWRT@
143:          out     (c),e
144:          ret
145:          ENDMODULE
146:
147: ;--- VPEEK ---
148: ;reads the VRAM
149:          MODULE  VPEEK
15Ø:          extrn   SETRD@
151: VPEEK@::
```



```
152:          call    SETRD@
153:          in      a,(c)
154:          ret
155:          ENDMODULE
156:
157: ;--- LDIRVM ---
158: ;moves block from memory to the VRAM
159:          MODULE  LDIRVM
16Ø:          extrn   SETWRT@
161: LDIRVM@::
162:          push    bc             ;save length
163:          call    SETWRT@
164:          ex      de,hl          ;C..(WRPORT@)
165:          pop     de             ;HL..pointer to source  DE..length
166:          ld      b,e            ;set lower byte of length
167:          ld      a,e
168:          or      a
169:          ld      a,d            ;set higher byte of length
17Ø:          jr      z,LDIRV1
171:          inc     a
172: LDIRV1:
173:          outi
174:          jp      nz,LDIRV1
175:          dec     a
176:          jp      nz,LDIRV1
177:          ret
178:          ENDMODULE
179:
18Ø: ;--- LDIRMV ---
181: ;moves block from VRAM to memory
182:          MODULE  LDIRMV
183:          extrn   SETRD@
184: LDIRMV@::
185:          ex      de,hl          ;HL..pointer to source
186:          push    bc             ;save length
187:          call    SETRD@
188:          ex      de,hl          ;C..(RDPORT@)
189:          pop     de             ;HL..pointer to destination  DE..length
19Ø:          ld      b,e            ;set lower byte of length
191:          ld      a,e
192:          or      a
193:          ld      a,d            ;set higher byte of length
194:          jr      z,LDIRM1
195:          inc     a
196: LDIRM1:
197:          ini
198:          jp      nz,LDIRM1
199:          dec     a
2ØØ:          jp      nz,LDIRM1
2Ø1:          ret
2Ø2:          ENDMODULE
2Ø3:
2Ø4: ;--- FILVRM ---
```

```
205: ;fills the VRAM
206:          MODULE  FILVRM
207:          extrn   SETWRT@
208: FILVRM@::
209:          push    bc              ;save data
210:          call    SETWRT@
211:          pop     hl              ;L..data  DE..length  C..(WRPORT@)
212:          ld      b,e             ;set lower byte of length
213:          ld      a,e
214:          or      a
215:          ld      a,d             ;set higher byte of length
216:          jr      z,FILVR1
217:          inc     a
218: FILVR1:
219:          out     (c),l
220:          djnz    FILVR1
221:          dec     a
222:          jr      nz,FILVR1
223:          ret
224:          ENDMODULE
225:
226: ;--- GETSTAT ---
227: ; Wait VDP Ready
228: ;modify af,bc
229:          MODULE  GETSTAT
230:          extrn   WRPORT@
231:          extrn   RDPORT@
232: GETSTAT::
233:          push    bc
234:          ld      bc,(WRPORT@)  ;Read Status Register Specified by A
235:          inc     c
236:          out     (c),a
237:          ld      a,8fh
238:          out     (c),a
239:          ld      bc,(RDPORT@)
240:          inc     c
241:          in      a,(c)
242:          pop     bc
243:          ret
244:          ENDMODULE
245:
246: ;--- WAITVDP ---
247:          MODULE  WAITVDP
248:          extrn   GETSTAT
249: WAITVDP::
250:          ld      a,2
251:          call    GETSTAT
252:          and     1
253:          jr      nz,WAITVDP
254:          xor     a
255:          jp      GETSTAT
256:          ENDMODULE
257:
```



```
258: ;--- BOXFIL ---
259:          MODULE  BOXFIL
260:          extrn   WAITVDP
261:          extrn   WRPORT@
262: ACPAGE   equ     0faf6h
263: LINWRK   equ     0fc18h
264: SCRMOD   equ     0fcafh
265: BOXFIL@::
266:          pop     iy                  ;return address
267:          ld      (LINWRK+4),hl       ;save x start
268:          ld      a,(SCRMOD)
269:          cp      7
270:          ld      a,(ACPAGE)          ;active page
271:          jr      c,BOXFI1
272:          add     a,a
273: BOXFI1:
274:          add     a,d
275:          ld      (LINWRK+7),a        ;save y start
276:          ld      a,e
277:          ld      (LINWRK+6),a
278:          ld      a,4
279:          or      a
280:          sbc     hl,bc               ;hl..x start - x end
281:          jr      nc,BOXFI2
282:          ld      c,l
283:          ld      b,h
284:          ld      hl,0
285:          or      a
286:          sbc     hl,bc
287:          xor     a
288: BOXFI2:
289:          inc     hl
290:          ld      (LINWRK+8),hl       ;save number of dots for x
291:          ld      (LINWRK+13),a       ;arg
292:          pop     hl                  ;hl..y end
293:          or      a
294:          sbc     hl,de               ;hl..y end - y start
295:          jr      nc,BOXFI3
296:          ex      de,hl
297:          ld      hl,0
298:          or      a
299:          sbc     hl,de
300:          ld      a,(LINWRK+13)
301:          set     3,a
302:          ld      (LINWRK+13),a
303: BOXFI3:
304:          inc     hl
305:          ld      (LINWRK+10),hl      ;save number of dots for y
306:          pop     hl
307:          ld      a,l
308:          ld      (LINWRK+12),a       ;color code
309:          pop     hl                  ;logical operation
310:          ld      a,l
```

```
311:            or      10000000b       ;LMMV command
312:            ld      (LINWRK+14),a
313:            push    hl              ;adjust stack
314:            push    hl
315:            push    hl
316:            push    iy              ;return address
317:            ld      hl,LINWRK+4
318:            di
319:            call    WAITVDP
320:            ld      a,(WRPORT0)
321:            ld      c,a
322:            inc     c
323:            ld      a,36
324:            out     (c),a
325:            ld      a,17+80h
326:            out     (c),a
327:            inc     c
328:            inc     c
329:            ld      b,11
330:            otir
331:            ei
332:            ret
333:            ENDMODULE
334:
335: ;--- LINE ---
336:            MODULE  LINE
337:            extrn   WAITVDP
338:            extrn   WRPORT0
339: ACPAGE     equu    0faf6h
340: LINWRK     equ     0fc18h
341: SCRMOD     equ     0fcafh
342: LINE0::
343:            pop     iy              ;return address
344:            ld      (LINWRK+4),hl   ;save x start
345:            ld      a,(SCRMOD)
346:            cp      7
347:            ld      a,(ACPAGE)      ;active page
348:            jr      c,LINE1
349:            add     a,a
350: LINE1:
351:            add     a,d
352:            ld      (LINWRK+7),a    ;save y start
353:            ld      a,e
354:            ld      (LINWRK+6),a
355:            ld      a,4
356:            or      a
357:            sbc     hl,bc           ;hl..x start - x end
358:            jr      nc,LINE2
359:            ld      c,l
360:            ld      b,h
361:            ld      hl,0
362:            or      a
363:            sbc     hl,bc
```



```
364:            xor     a
365: LINE2:
366:            ld      (LINWRK+8),hl   ;save number of dots for x
367:            ld      (LINWRK+13),a   ;arg
368:            pop     hl              ;hl..y end
369:            or      a
370:            sbc     hl,de           ;hl..y end - y start
371:            jr      nc,LINE3
372:            ex      de,hl
373:            ld      hl,0
374:            or      a
375:            sbc     hl,de
376:            ld      a,(LINWRK+13)
377:            set     3,a
378:            ld      (LINWRK+13),a
379: LINE3:
380:            ld      de,(LINWRK+8)   ;number of dots for x
381:            or      a
382:            sbc     hl,de
383:            jr      nc,LINE4
384:            add     hl,de
385:            ld      (LINWRK+10),hl  ;save number of dots for y
386:            jr      LINE5
387: LINE4:
388:            ld      a,(LINWRK+13)
389:            set     0,a
390:            ld      (LINWRK+13),a
391:            ld      (LINWRK+10),de  ;save number of dots for x
392:            add     hl,de
393:            ld      (LINWRK+8),hl   ;number of dots for y
394: LINE5:
395:            pop     hl
396:            ld      a,l
397:            ld      (LINWRK+12),a   ;color code
398:            pop     hl              ;logical operation
399:            ld      a,l
400:            or      01110000b       ;LINE command
401:            ld      (LINWRK+14),a
402:            push    hl              ;adjust stack
403:            push    hl
404:            push    hl
405:            push    iy              ;return address
406:            ld      hl,LINWRK+4
407:            di
408:            call    WAITVDP
409:            ld      a,(WRPORT0)
410:            ld      c,a
411:            inc     c
412:            ld      a,36
413:            out     (c),a
414:            ld      a,17+80h
415:            out     (c),a
416:            inc     c
```

```
417:            inc     c
418:            ld      b,11
419:            otir
42Ø:            ei
421:            ret
422:            ENDMODULE
423:
424: ;--- CPYM2V ---
425:            MODULE  CPYM2V
426:            extrn   WAITVDP
427:            extrn   GETSTAT
428:            extrn   WRPORT@
429: LINWRK     equ     Øfc18h
43Ø: SCRMOD     equ     Øfcafh
431: CPYM2V@::
432:            pop     iy              ;return address
433:            ld      a,e
434:            and     3
435:            rlca
436:            rlca
437:            ld      (LINWRK+13),a   ;arg
438:            ld      (LINWRK+4),bc   ;save destination x start
439:            pop     bc              ;destination y start
44Ø:            pop     de              ;destination page
441:            ld      a,(SCRMOD)
442:            cp      7
443:            ld      a,e
444:            jr      c,CPYMV1
445:            add     a,a
446: CPYMV1:
447:            add     a,b
448:            ld      (LINWRK+7),a    ;save destination y start
449:            ld      a,c
45Ø:            ld      (LINWRK+6),a
451:            pop     bc              ;logop
452:            ld      a,c
453:            or      1Ø11ØØØØb       ;LMMC command
454:            ld      (LINWRK+14),a
455:            ld      de,LINWRK+8
456:            ld      bc,4
457:            ldir
458:            ld      a,(hl)
459:            ld      (LINWRK+12),a
46Ø:            push    hl              ;adjust stack
461:            push    hl
462:            push    hl
463:            push    iy              ;return address
464: ;
465:            push    hl
466:            ld      hl,LINWRK+4
467:            di
468:            call    WAITVDP
469:            ld      a,(WRPORT@)
```



```
47Ø:            ld      c,a
471:            inc     c
472:            ld      a,36
473:            out     (c),a
474:            ld      a,17+8Øh
475:            out     (c),a
476:            inc     c
477:            inc     c
478:            ld      b,11
479:            otir
48Ø:            pop     hl
481: ;
482:            dec     c
483:            dec     c               ;C:PORT#1's address
484: CPYMV2:
485:            ld      a,2
486:            call    GETSTAT
487:            bit     Ø,a
488:            jr      z,CPYMV3        ;check CE bit
489:            bit     7,a
49Ø:            jr      z,CPYMV2        ;check TR bit
491:            inc     hl
492:            ld      a,(hl)
493:            out     (c),a
494:            ld      a,8Øh+44
495:            out     (c),a
496:            jr      CPYMV2
497: CPYMV3:
498:            xor     a
499:            call    GETSTAT
5ØØ:            ei
5Ø1:            ret
5Ø2:            ENDMODULE
5Ø3:
5Ø4: ;--- CPYV2V ---
5Ø5:            MODULE  CPYV2V
5Ø6:            extrn   WAITVDP
5Ø7:            extrn   WRPORT@
5Ø8: LINWRK     equ     Øfc18h
5Ø9: SCRMOD     equ     Øfcafh
51Ø: CPYV2V@::
511:            pop     iy              ;return address
512:            ld      (LINWRK),hl     ;save source x start
513:            ld      a,4
514:            or      a
515:            sbc     hl,bc           ;hl..sx start - sx end
516:            jr      nc,CPYVV1
517:            ld      c,l
518:            ld      b,h
519:            ld      hl,Ø
52Ø:            or      a
521:            sbc     hl,bc
522:            xor     a
```

```
523: CPYVV1:
524:        inc     hl
525:        ld      (LINWRK+8),hl     ;save number of dots for x
526:        ld      (LINWRK+13),a     ;arg
527:        pop     hl                ;hl..source y end
528:        pop     bc                ;c..source page
529:        ld      a,(SCRMOD)
53Ø:        cp      7
531:        ld      a,c
532:        jr      c,CPYVV2
533:        add     a,a
534: CPYVV2:
535:        add     a,d
536:        ld      (LINWRK+3),a      ;save source y start
537:        ld      a,e
538:        ld      (LINWRK+2),a
539:        or      a
54Ø:        sbc     hl,de             ;hl..sy end - sy start
541:        jr      nc,CPYVV3
542:        ex      de,hl
543:        ld      hl,Ø
544:        or      a
545:        sbc     hl,de
546:        ld      a,(LINWRK+13)
547:        set     3,a
548:        ld      (LINWRK+13),a
549: CPYVV3:
55Ø:        inc     hl
551:        ld      (LINWRK+1Ø),hl    ;save number of dots for y
552:        pop     hl                ;destination x start
553:        pop     de                ;destination y start
554:        pop     bc                ;destination page
555:        ld      (LINWRK+4),hl
556:        ld      a,(SCRMOD)
557:        cp      7
558:        ld      a,c
559:        jr      c,CPYVV4
56Ø:        add     a,a
561: CPYVV4:
562:        add     a,d
563:        ld      (LINWRK+7),a      ;save destination y start
564:        ld      a,e
565:        ld      (LINWRK+6),a
566:        pop     hl                ;logical operation
567:        ld      a,l
568:        or      1ØØ1ØØØØb         ;LMMM command
569:        ld      (LINWRK+14),a
57Ø:        push    hl                ;adjust stack
571:        push    hl
572:        push    hl
573:        push    hl
574:        push    hl
575:        push    hl
```



```
576:        push    iy                ;return address
577:        ld      hl,LINWRK
578:        di
579:        call    WAITVDP
58Ø:        ld      a,(WRPORT@)
581:        ld      c,a
582:        inc     c
583:        ld      a,32
584:        out     (c),a
585:        ld      a,17+8Øh
586:        out     (c),a
587:        inc     c
588:        inc     c
589:        ld      b,15
59Ø:        otir
591:        ei
592:        ret
593:        ENDMODULE
594:
595: ;--- PALETT ---
596: ;set palette datas
597:        MODULE  PALETT
598:        extrn   WRPORT@
599: PALETT@::
6ØØ: ;set palette
6Ø1: ;entry  b...palette number  e...red*8+blue  d...green
6Ø2:        pop     ix
6Ø3:        ld      b,l
6Ø4:        pop     hl
6Ø5:        ld      a,c
6Ø6:        and     7
6Ø7:        ld      d,a
6Ø8:        ld      a,e
6Ø9:        and     7
61Ø:        rlca
611:        rlca
612:        rlca
613:        rlca
614:        ld      e,a
615:        ld      a,l
616:        and     7
617:        or      e
618:        ld      e,a
619:        push    hl
62Ø:        push    ix
621:        ld      a,(WRPORT@)
622:        ld      c,a
623:        inc     c
624:        di
625:        out     (c),b             ;palette number
626:        ld      a,16+8Øh
627:        out     (c),a
628:        inc     c
```

```
629:          out     (c),e
63Ø:          push    bc
631:          pop     bc
632:          push    bc
633:          pop     bc
634:          out     (c),d
635:          ei
636:          ret
637:          ENDMODULE
638:
639: ;--- RND ---
64Ø:          MODULE  RND
641: RND@::
642:          ld      a,h
643:          or      l
644:          ret     z
645:          ex      de,hl           ;save the range
646:          ld      hl,(RNDSPD@)
647:          ld      b,h
648:          ld      c,l
649:          add     hl,hl
65Ø:          add     hl,hl
651:          add     hl,bc
652:          ld      (RNDSPD@),hl
653:          ld      hl,(RNDSPD@+2)
654:          adc     hl,hl
655:          adc     hl,bc
656:          ld      (RNDSPD@+2),hl
657:          ld      a,h
658:          ld      c,l
659:          ld      hl,Ø
66Ø:          ld      b,16
661:          srl     a
662:          rr      c
663: RND1:    jr      nc,RND2
664:          add     hl,de
665: RND2:    rr      h
666:          rr      l
667:          rra
668:          rr      c
669:          djnz    RND1
67Ø:          ret
671:          dseg
672: RNDSPD@::        dw      1991h,Øc2Ø4h
673:          cseg
674:          ENDMODULE
675:
676: ;--- dummy ---
677:          MODULE  DMY2
678:          ENDMODULE
```


リスト　MSXCLIB.C

```
 Ø: #include <stdio.h>
 1: #include <msxalib.h>
 2:
 3: VOID _dmy1_() { }      ..... ダミーモジュール用関数
 4:
 5: VOID screen(md)
 6: int md;
 7: {
 8:     struct regs r;
 9:
1Ø:     r.a = (char)md;
11:     calbio(ØxØØ5f, &r);
12: }
13:
14: VOID color(fg, bg, bd)
15: int fg, bg, bd;
16: {
17:     struct regs r;
18:
19:     *((char *)Øxf3e9) = (char)fg;
2Ø:     *((char *)Øxf3ea) = (char)bg;
21:     *((char *)Øxf3eb) = (char)bd;
22:     calbio(ØxØØ62, &r);
23: }
24:
25: VOID inispr(sz)
26: int sz;
27: {
28:     struct regs r;
29:
3Ø:     r.bc = (int)((*((char *)Øxf3eØ) & Øxfc) | sz) * 256 + 1;
31:     calbio(ØxØØ47, &r);
32:     if (*((char *)Øxfaf8) != Ø) {
33:         r.bc = (int)(*((char *)Øxffe7) & Øxfd) * 256 + 8;
34:         calbio(ØxØØ47, &r);
35:     }
36:     calbio(ØxØØ69, &r);
37: }
38:
39: VOID sprite(no, dt)
4Ø: int no;
41: char *dt;
42: {
43:     if (*((char *)Øxf3eØ) & 2)
44:         ldirvm(*((int *)Øxf926) + no * 8, dt, 32);
45:     else
46:         ldirvm(*((int *)Øxf926) + no * 8, dt, 8);
47: }
48:
49: VOID putspr(pl, x, y, cl, pt)
5Ø: int pl, x, y, cl, pt;
```

```
51: {
52:     char dt[4];
53:
54:     dt[Ø] = (char)y;
55:     dt[1] = (char)x;
56:     dt[2] = (char)pt;
57:     dt[3] = (char)cl;
58:     ldirvm(*((int *)Øxf928) + pl * 4, dt, 4);
59: }
6Ø:
61: VOID colspr(pl, dt)
62: int pl;
63: char *dt;
64: {
65:     if (*((char *)Øxf3eØ) & 2)
66:         ldirvm(*((int *)Øxf928) - 512 + pl * 16, dt, 16);
67:     else
68:         ldirvm(*((int *)Øxf928) - 512 + pl * 16, dt, 8);
69: }
7Ø:
71: TINY gtstck(no)
72: int no;
73: {
74:     struct regs r;
75:
76:     r.a = (char)no;
77:     calbio(ØxØØd5, &r);
78:     return (r.a);
79: }
8Ø:
81: TINY gttrig(no)
82: int no;
83: {
84:     struct regs r;
85:
86:     r.a = (char)no;
87:     calbio(ØxØØd8, &r);
88:     return (r.a != Ø);
89: }
9Ø:
91: int gtpad(no)
92: int no;
93: {
94:     struct regs r;
95:
96:     r.a = (char)no;
97:     calbio(ØxØØdb, &r);
98:     if (r.a & Øx8Ø)
99:         return (int)(r.a) - 256;
1ØØ:    else
1Ø1:        return (int)(r.a);
1Ø2: }
1Ø3:
```



```
1Ø4: TINY snsmat(mt)
1Ø5: int mt;
1Ø6: {
1Ø7:     struct regs r;
1Ø8:
1Ø9:     r.a = (char)mt;
11Ø:     calbio(Øx Ø141, &r);
111:     return (r.a);
112: }
113:
114: VOID kilbuf()
115: {
116:     struct regs r;
117:
118:     calbio(Øx Ø156, &r);
119: }
12Ø:
121: VOID gicini()
122: {
123:     struct regs r;
124:
125:     calbio(ØxØØ9Ø, &r);
126: }
127:
128: VOID sound(rg, dt)
129: int rg, dt;
13Ø: {
131:     struct regs r;
132:
133:     r.a = (char)rg;
134:     r.de = dt;
135:     calbio(ØxØØ93, &r);
136: }
137:
138: int sgn(vl)
139: int vl;
14Ø: {
141:     if (vl < Ø)
142:         return -1;
143:     if (vl != Ø)
144:         return 1;
145:     return Ø;
146: }
147:
148: VOID srnd()
149: {
15Ø:     extern int RNDSPD[2];
151:
152:     RNDSPD[1] = *((int *)Øxfca2);
153: }
154:
155: VOID _dmy2_() { }      ..... ダミーモジュール用関数
```

ヘッダーファイル

リスト　MSXALIB.H

```
struct regs {
    char     f;
    char     a;
    unsigned bc;
    unsigned de;
    unsigned hl;
};

VOID ginit();
VOID calbio();
VOID vpoke();
TINY vpeek();
VOID ldirvm();
VOID ldirmv();
VOID filvrm();
VOID boxfil();
VOID line();
VOID cpym2v();
VOID cpyv2v();
VOID palett();
unsigned rnd();
```

リスト　MSXCLIB.H

```
VOID screen();
VOID color();
VOID inispr();
VOID sprite();
VOID putspr();
VOID colspr();
TINY gtstck();
TINY gttrig();
int  gtpad();
TINY snsmat();
VOID kilbuf();
VOID gicini();
VOID sound();
int  sgn();
VOID srnd();
```


MSX-DOS用バッチプログラム

リスト　AMAKE.BAT

```
del msxalib.rel
ren *.rel *.rrr
mx -l msxalib > temp.bat
echo "del *.rel" >> temp.bat
echo "ren *.rrr *.rel" >> temp.bat
echo "ren msxalib.lib msxalib.rel" >> temp.bat
temp
```

リスト　CMAKE.BAT

```
del msxclib.rel
ren *.rel *.rrr
del msxclib.tco
cf msxclib
mx -l msxclib >temp.bat
echo "del msxclib.tco" >> temp.bat
echo "del *.rel" >> temp.bat
echo "ren *.rrr *.rel" >> temp.bat
echo "ren msxclib.lib msxclib.rel" >> temp.bat
temp
```

リスト　ACHG.BAT

```
mx msxalib %1
m8Ø =%1/z
lib8Ø /e temp=msxalib<..%1-1>,%1,msxalib<%1+1..>
copy temp.rel msxalib.rel
del temp.rel
del %1.mac
del %1.rel
```

リスト　CCHG.BAT

```
cf msxclib
mx msxclib %1
cg %1
m8Ø =%1/z
lib8Ø /e temp=msxclib<..%1-1>,%1,msxclib<%1+1..>
copy temp.rel msxclib.rel
del temp.rel
del msxclib.tco
del %1.tco
del %1.mac
del %1.rel
```

MSX-DOS 2 用バッチプログラム

リスト AMAKE.BAT

```
del msxalib.rel
ren *.rel *.rrr
mx -l msxalib > temp.bat
echo del *.rel >> temp.bat
echo ren *.rrr *.rel >> temp.bat
echo ren msxalib.lib msxalib.rel >> temp.bat
temp
```

リスト CMAKE.BAT

```
del msxclib.rel
ren *.rel *.rrr
del msxclib.tco
cf msxclib
mx -l msxclib >temp.bat
echo del msxclib.tco >> temp.bat
echo del *.rel >> temp.bat
echo ren *.rrr *.rel >> temp.bat
echo ren msxclib.lib msxclib.rel >> temp.bat
temp
```

リスト ACHG.BAT

```
mx msxalib %1
m8Ø =%1/z
lib8Ø /e temp=msxalib<..%1-1>,%1,msxalib<%1+1..>
copy temp.rel msxalib.rel
del temp.rel
del %1.mac
del %1.rel
```

リスト CCHG.BAT

```
cf msxclib
mx msxclib %1
cg %1
m8Ø =%1/z
lib8Ø /e temp=msxclib<..%1-1>,%1,msxclib<%1+1..>
copy temp.rel msxclib.rel
del temp.rel
del msxclib.tco
del %1.tco
del %1.mac
del %1.rel
```


索　引

A

ACHG. BAT …… 21, 233, 251, 252
ACROSS. C …… 111
AMAKE. BAT … 21, 233, 251, 252
AREL. BAT …… 22

B

BIOS …… 95
BOXFIL. C …… 46
boxfil関数 …… 45

C

CALBIO. C …… 96
calbio関数 …… 95
CCHG. BAT …… 21, 233, 251, 252
CF. COM …… 22
CG. COM …… 22
CMAKE. BAT … 21, 233, 251, 252
COLOR. C …… 40
color関数 …… 39
COLSPR. C …… 63
colspr関数 …… 59
CPYM 2 V. C …… 50
cpym 2 v関数 …… 47
CPYV 2 V. C …… 53
cpyv 2 v関数 …… 51
CREL. BAT …… 22

E

ECHO. COM …… 22
extern …… 179

F

FILVRM. C …… 73
filvrm関数 …… 72
fprintf関数 …… 172
fscanf関数 …… 172

G

gicini関数 …… 89
ginit関数 …… 36
GTPAD. C …… 88
gtpad関数 …… 86
GTSTCK. C …… 82
gtstck関数 …… 80
GTTRIG. C …… 85
gettrig関数 …… 83

H

HIPENEMY. C …… 156, 190
HIPINIT. C …… 156, 188
HIPMAN. C …… 156, 195