Event Commands

Character Action Queues (00-34)

The action queue will consist of movement actions.

Character Numbers (xx):

- $00-$0F: Player Characters (Terra, Locke, Cyan ...)
- $10-$2F: NPC's
- $30: Camera
- $31-$34: Party Characters

```
xx nn    xx = Character Number, nn = Number of actions to perform (including
the final end command)
         If nn & 0x80, the next action will not take place until this and
any before it are complete.
```

General Actions (35-FF)

Jump table will be at C0/98C4

Character Actions (35-47)

```
35 xx                   $C09C44     Pauses execution until camera queue is
complete
36 xx                   $C09C6F     Disable ability to pass through objects
for character XX
37 xx yy                $C09C8F     Assign graphic set yy to object xx
38                      $C09D0E     Hold screen--does not scroll when
character moves
39                      $C09D16     Free screen--does scroll when character
moves
3A                      $C09D1E     Enable player to move while event
commands execute
3B                      $C09D2E     Position player in a "ready-to-go"
stance
3C c1 c2 c3 c4          $C09D6D     Put characters c1-4 in the party
3D xx                   $C09E3C     Create object xx
3E xx                   $C09E67     Delete object xx
3F xx yy                $C09D3B     Assign character xx to party yy (if yy
== 0, remove character from party)
40 xx yy                $C0A07C     Assign properties yy to character xx
41 xx                   $C0A2FA     Show object xx
42 xx                   $C0A336     Hide object xx
```

```
43 xx yy                $C09CA9     Assign palette yy to object xx
44 xx yy                $C09CCA     Place object xx on vehicle yy
                                    00: No vehicle         40: Magitek Armor
80: Object visible
                                    20: Chocobo            60: Raft
45                      $C09CE2     Refresh objects
46 xx                   $C09CEA     Make xx the current party
47                      $C09D03     Make character in slot 0 the lead
character
```

Dialogue (48-4B)

```
48 xxxx                 $C0A475     Display dialogue message xxxx, continue
executing commands
49                      $C0A4A6     If dialogue window is up, wait for
keypress then dismiss
4A                      $C0A4B0     Wait for keypress
4B xxxx                 $C0A4BC     Display dialogue message xxxx, halt
execution until gone
                                    If xxxx & 0x4000, only the text will be
shown (no dialogue window)
                                    If xxxx & 0x8000, the text will be shown
at the bottom of the screen
                                    (note these flags apply to both command
48 and 4B)
```

Invoke Battle (4C-4F)

```
4C xx bb                $C0A591     Center screen on party and invoke
battle, enemy set xx, background bb
4D xx bb                $C0A578     Battle Enemy Set xx, Background Scenery
bb
4E                      $C0A4F9     Invoke random battle, used for
dungeons/towns, etc
4F                      $C0A5F3     Exit current location
```

Screen Actions (50-65)

```
50 xx                   $C0A5FD     Tint screen (cumulative) (takes 32
executions at intensity 15 to saturate)
51 ooorgbii pb pe       $C0A640     Modify BG color range from [pb, pe]
                                    001: Add color component
                                    101: Subtract color component
52 xx                   $C0A686     Tint characters (cumulative)
53 ooorgbii pb pe       $C0A6C5     Modify OBJ color range [pb, pe]
```

```
54                          $C0A784
55 ci                       $C0A795     Flash screen with color component(s) c,
intensity i
56 ci                       $C0A7BA     Increase color component(s) c, intensity
i
57 ci                       $C0A7D0     Decrease color component(s) c, intensity
i
                                        Color components:
                                        2: Red                      A: Red +
Blue (Magenta)
                                        4: Green                    C: Green +
Blue (Cyan)
                                        6: Red + Green (Yellow)     E: Red +
Green + Blue (White)
                                        8: Blue
58 xx                       $C0A7E6     Shake the screen (xx?)
59 xx                       $C0A80A     Unfade the screen at speed x
5A _x                       $C0A817     Fade the screen at speed x
5B                          $C0A826
5C                          $C0A82D     Pause execution until fade in or fade
out is complete
5D xx yy                    $C0A838     Scroll BG0, speed xx x yy (00 --> 7F
left/up, 80 <-- FF right/down)
5E xx yy                    $C0A8CE     Scroll BG1, speed xx x yy (00 --> 7F
left/up, 80 <-- FF right/down)
5F xx yy                    $C0A964     Scroll BG2, speed xx x yy (00 --> 7F
left/up, 80 <-- FF right/down)
60 xx yy                    $C0A9FA     Change background layer xx palette to yy
61 _c pb pe                 $C0AA3D     Colorize color range [pb, pe] to color c
62 xx                       $C0AACB     Mosaic screen with speed xx (lower ==
slower)
63 xx                       $C0AADB     Create spotlight effect with radius xx
64 xx yy                    $C0AAE9
65 xx yy                    $C0AB09
```

Map Actions (6A-76)

```
6A nnnn xx yy ff            $C0AB47     Loads map nnnn, positions party at (xx,
yy)
                                        ff:
                                        01:     Party is in airship
                                        02:     Party is on Chocobo
6B nnnn xx yy ff            $C0AB55     Loads map nnnn, positions party at (xx,
yy)
6C xxxx xx yy ff            $C0AC0B     Set parent map to xxxx, parent
coordinates to (xx, yy)
70 xx yy                    $C0A881     Scroll BG0
71 xx yy                    $C0A917     Scroll BG1
72 xx yy                    $C0A9AD     Scroll BG2
73 xx yy rr cc data...      $C0AC45     Copy data of size rr*cc to current map's
```

```
BG0 at (xx, yy) and refresh background
74 xx yy rr cc data...  $C0AC62     Copy data of size rr*cc to current map's
BG0 at (xx, yy)
75                      $C0AC1F     Refresh background after map has been
changed
```

Character and Party Actions (77-90)

```
77 xx                   $C09F32     Restore character xx HP and MP to full
78 xx                   $C09C7F     Enable ability to pass through other
objects for object
79 xx yy zz             $C0A36A     Place party on map xxxx
7A xx aaaaaa            $C0A42A     Modify entity event--call
$aaaaaa+$CA0000 when triggered
7B                      $C0A441     Restore backup party to active status
7C xx                   $C0A455     Enable activation of event for object xx
on collision
7D xx                   $C0A465
7E xx yy                $C0A39A     Move the characters to (xx, yy) on the
current map
7F xx yy                $C0A03A     Change character xx's name to yy
80 xx                   $C0ACF0     Add item xx from inventory
81 xx                   $C0AD2D     Remove item xx from inventory
82                      $C0A570
84 xxxx                 $C0AD50     Give xxxx amount of GP to party
85 xxxx                 $C0AD7F     Take xxxx amount of GP from party
86 xx                   $C0ADB8     Give Esper xx to party
87 xx                   $C0ADD7     Take Esper xx from party
                                    Espers:
                                    36: Ramuh           40: Tritoch
4A: Phantom
                                    37: Ifrit           41: Odin
4B: Sraphim
                                    38: Shiva           42: Raiden
4C: Golem
                                    39: Siren           43: Bahamut
4D: Unicorn
                                    3A: Terrato         44: Alexandr
4E: Fenrir
                                    3B: Shoat           45: Crusader
4F: Starlet
                                    3C: Maduin          46: Ragnarok
50: Phoenix
                                    3D: Bismark         47: Kirin
                                    3E: Stray           48: ZoneSeek
                                    3F: Palidor         49: Carbunkl
88 xx c1 c2             $C0AE2D     Remove all but status conditions c1 and
c2 from character xx (c1 = Condition Effects 1, c2 = Condition Effects 4)
```

```
89 xx c1 c2             $C0AE47     Inflict status conditions c1 and c2 on
character xx (c1 = Condition Effects 1, c2 = Condition Effects 4)
8A xx c1 c2             $C0AE61     Toggle status conditions c1 and c2 for
character xx (c1 = Condition Effects 1, c2 = Condition Effects 4)
8B xx yy                $C0AE7B
8C xx yy                $C0AF3E
8D xx                   $C09FCE     Remove all equipment from character xx
and place it in the inventory
8E                      $C0A54E
8F                      $C0AFF8     Unlock all of Cyan's SwordTechs
90                      $C0B002     Grant Sabin the Bum Rush
```

Wait/Pause (91-95)

```
91                      $C0B23F     Pause for 1/4 second
92                      $C0B249     Pause for 1/2 second
93                      $C0B253     Pause for 3/4 second
94                      $C0B25D     Pause for 1 second
95                      $C0B267     Pause for 2 seconds
```

Menus, Timers and Cinematics (96-aF)

```
96                      $C0A7F0     Refresh screen after a menu or battle
97                      $C0A7FD     Fade screen to black
98 xx                   $C0B00F     Invoke name change screen for character
xx
99 xx yy zz             $C0B035     Invoke party selection screen:  xx =
number of groups
9A                      $C0B0B2     Invoke Coliseum item selection screen
9B xx                   $C0B06D     Invoke shop xx
9C xx                   $C0B08C
9D                      $C0B09C     Invoke Party Order screen (from final
battle)
A0 xx xx yy yy zz       $C0B0E0     Set timer to XXXX jiffies, jump to YYYY
on expiration, flags ZZ (includes index, etc.)
A1 xx                   $C0B10E     Reset timer XX
A2                      $C0B130
A6                      $C0BA09
A7 xx                   $C0BA14
A8                      $C0BA51     Show Floating Island soaring into the
sky
A9                      $C0B966     Show title screen
AA                      $C0B992     Show intro with Magitek Armor walking
through snowfields
AB                      $C0B91B     Invoke game loading screen
AC                      $C0B95E
AD                      $C0BA69     Show world getting torn apart
AE                      $C0B9C5     Show train car ride out of the Magitek
```

```
Factory
AF                      $C0A503     Invoke random Coliseum battle
```

Execution (B0-BF)

```
B0 xx                   $C0B138     Repeat the the following commands (until
B1 is encountered) xx times
B1                      $C0B145     End block of commands to repeat
B2 aaaaaa               $C0B1A1     Call subroutine $aaaaaa+$CA0000
B3 nn aaaaaa            $C0B1DF     Call subroutine $aaaaaa+$CA0000, nn
times
B4 xx                   $C0B21D     Pause for xx/60 seconds
B5 xx                   $C0B227     Pause for xx/4 seconds
B6 aaaaaa ...           $C0B6D3     Jump to the nth address following B6,
where n is the last item selected from a
                                    multiple-choice dialogue window.  Each
address is 3 bytes, added to $CA0000/$0A0200.
B7 xx aaaaaa            $C0B299     Jump to $aaaaaa+$CA0000 if bit $1DC9 +
$xx is clear
B8 xx                   $C0B6AB     Set bit $1DC9 + $xx
B9 xx                   $C0B6BF     Clear bit $1DC9 + $xx
BA xx                   $C0BA31
BB                      $C0B9BE
BC _xxx                 $C0B16F     Return if event bit $xxx is clear
BD aaaaaa               $C0B271     Randomly jump to $aaaaaa+$CA0000
BE nn caaaaa ...        $C0B6F7     Switch/case based on the value of
CaseWord ($1EB4-5)
                                    Number of parameters (bytes) = (nn * 3)
+ 1 (+1 accounts for the nn)
                                    "c" represents the bit to test in
CaseWord; if set, the jump will occur
BF                      $C0B9E7     Show airship scene from the ending
```

Conditionals (C0-CF)

```
If MSB of bit to test is 0, the bit will be compared to 0, else it will be
compared to 1.
[In a nutshell, if(tx & 0x8000) -> if(*tx); if(!(tx & 0x8000)) ->
if(!(*tx))]
If result of all comparisons is true, jump will occur; otherwise, execution
will occur at the next command.
Each condition to test (the bit value) is 2 bytes--the address to jump to is
3 bytes, and is added to $CA0000.

C0-C7: $C0B2C8, C8-CF: $C0B32D

C0 t1 addr                          if(t1) jump; else continue;
```

```
C1 t1 t2 addr                    if(t1 || t2) jump; else continue;
C2 t1 t2 t3 addr                 if(t1 || t2 || t3) jump; else continue;
C3 t1 t2 t3 t4 addr              if(t1 || t2 || t3 || t4) jump; else
continue;
C4 t1 t2 t3 t4 t5 addr           if(t1 || t2 || t3 || t4 || t5) jump;
else continue;
C5 t1 t2 t3 t4 t5 t6 addr        if(t1 || t2 || t3 || t4 || t5 || t6)
jump; else continue;
C6 t1 t2 t3 t4 t5 t6 t7 addr     if(t1 || t2 || t3 || t4 || t5 || t6 ||
t7) jump; else continue;
C7 t1 t2 t3 t4 t5 t6 t7 t8 addr  if(t1 || t2 || t3 || t4 || t5 || t6 ||
t7 || t8) jump; else continue;
C8 t1 addr                       if(t1) jump; else continue;
C9 t1 t2 addr                    if(t1 && t2) jump; else continue;
CA t1 t2 t3 addr                 if(t1 && t2 && t3) jump; else continue;
CB t1 t2 t3 t4 addr              if(t1 && t2 && t3 && t4) jump; else
continue;
CC t1 t2 t3 t4 t5 addr           if(t1 && t2 && t3 && t4 && t5) jump;
else continue;
CD t1 t2 t3 t4 t5 t6 addr        if(t1 && t2 && t3 && t4 && t5 && t6)
jump; else continue;
CE t1 t2 t3 t4 t5 t6 t7 addr     if(t1 && t2 && t3 && t4 && t5 && t6 &&
t7) jump; else continue;
CF t1 t2 t3 t4 t5 t6 t7 t8 addr  if(t1 && t2 && t3 && t4 && t5 && t6 &&
t7 && t8) jump; else continue;
```

Event Bits (D0-EF)

```
D0 xx                   $C0B593     Set event bit $0xx
D1 xx                   $C0B5CF     Clear event bit $0xx
D2 xx                   $C0B5A7     Set event bit $1xx
D3 xx                   $C0B5E3     Clear event bit $1xx
D4 xx                   $C0B5BB     Set event bit $2xx
D5 xx                   $C0B5F7     Clear event bit $2xx
D6 xx                   $C0B60B     Set event bit $3xx
D7 xx                   $C0B65B     Clear event bit $3xx
D8 xx                   $C0B61F     Set event bit $4xx
D9 xx                   $C0B66F     Clear event bit $4xx
DA xx                   $C0B633     Set event bit $5xx
DB xx                   $C0B683     Clear event bit $5xx
DC xx                   $C0B647     Set event bit $6xx
DD xx                   $C0B697     Clear event bit $6xx
DE                      $C0B40B     Load CaseWord with the characters in the
active party
DF                      $C0B465
E0                      $C0B513     Load CaseWord with event bits $2E0-$2EF
(characters encountered so far)
E1                      $C0B51E     Load CaseWord with event bits $2F0-$2FF
E2                      $C0B4B9
E3                      $C0B3B7     Load CaseWord with the available
```

```
characters
E4                      $C0B39E     Load CaseWord with currently active
party
E7 xx                   $C0B394
E8 xx yyyy              $C0B529     Set event word xx to yyyy
E9 xx yyyy              $C0B53C     Increment event word xx by yyyy
EA xx yyyy              $C0B556     Decrement event word xx by yyyy
EB xx yyyy              $C0B571     If event word xx == yyyy then CaseWord =
$0001
                                    Else If event word xx > yyyy then
CaseWord = $0002
                                    Else If event word xx < yyyy then
CaseWord = $0004
EF xx yy                $C0B7AA
```

Sound / Music (F0-FC)

```
F0 xx                   $C0B780     Play song xx
F1 xx yy                $C0B7D4     Fade in song xx with speed yy (higher ==
slower)
F2 xx                   $C0B811     Fade out current song with speed xx
(higher == slower)
F3 xx                   $C0B827     Continue song that was previously
paused, xx is unknown
F4 xx                   $C0B854     Play sound effect xx
F5 xx yy zz             $C0B85E
F6 xx yy zz             $C0B889     Stop sound effect xx
F7                      $C0B8A1
F8                      $C0B8AF
F9 xx                   $C0B8BA
FA                      $C0B8C7     Stop temporarily played song
```

End Script (FD-FF)

```
FD                      $C0B8D2     NOP
FE                      $C0B8D7     Return
```

Unused General Actions

The following is a list of unused pointers from the general actions jump table ($C0/98C4).

```
C0/9926:    1A B9        (act. 66: ** NOT USED **)
C0/9928:    1A B9        (act. 67: ** NOT USED **)
C0/992A:    1A B9        (act. 68: ** NOT USED **)
C0/992C:    1A B9        (act. 69: ** NOT USED **)
```

```
C0/9934:    1A B9        (act. 6D: ** NOT USED **)
C0/9936:    1A B9        (act. 6E: ** NOT USED **)
C0/9938:    1A B9        (act. 6F: ** NOT USED **)

C0/9946:    1A B9        (act. 76: ** NOT USED **)

C0/9960:    1A B9        (act. 83: ** NOT USED **)

C0/9996:    1A B9        (act. 9E: ** NOT USED **)
C0/9998:    1A B9        (act. 9F: ** NOT USED **)

C0/99A0:    1A B9        (act. A3: ** NOT USED **)
C0/99A2:    1A B9        (act. A4: ** NOT USED **)
C0/99A4:    1A B9        (act. A5: ** NOT USED **)

C0/9A24:    1A B9        (act. E5: ** NOT USED **)
C0/9A26:    1A B9        (act. E6: ** NOT USED **)

C0/9A32:    1A B9        (act. EC: ** NOT USED **)
C0/9A34:    1A B9        (act. ED: ** NOT USED **)
C0/9A36:    1A B9        (act. EE: ** NOT USED **)

C0/9A52:    1A B9        (act. FC: ** NOT USED **)
C0/9A58:    1A B9        (act. FF: ** NOT USED **)
```