H8/OS Ver 3.3 システムコール仕様書 (gcc用)

1. 標準出力 sys_write

   SH2-7045/H8-3067/3068/3069/3048/3052/3664対応

```
#include <h8/syscall.h>

int sys_write(char *data, int size)
```

   第1引数 data : 出力するデータのポインタ

   第2引数 size : 出力するデータの長さ (バイト)

2. 標準文字列出力 write_string

   SH2-7045/H8-3067/3068/3069/3048/3052/3664対応

```
#include <h8/syscall.h>

int write_string(char *string)
```

   第1引数 data : 出力する文字列のポインタ

3. 出力モード指定 write_mode

   SH2-7045/H8-3067/3068/3069/3048/3052/3664対応

```
#include <h8/syscall.h>

int write_mode(int mode)
```

   第1引数 mode : 出力モード指定

   write_mode(BINARY) : バイナリデータ出力にする

   write_mode(ASCII) : 文字出力にする

   write_mode(SIO) : シリアル端末出力にする

   write_mode(LCD) : キャラクタ液晶出力にする

4. 標準入力 sys_read

   SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int sys_read(char *data, int size)
```

   第 1 引数 data : 入力データを格納する領域のポインタ

   第 2 引数 size : 入力データを格納する最大数

   戻り値 : 入力データのサイズ (バイト)

   入力データがない場合は、戻り値が 0 で即時に戻る

   入力データがある場合には、戻り値に入力データの大きさ (バイト数)

5. 標準文字列入力 read_string

   SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int sys_read(char *data, int size)
```

   第 1 引数 data : 入力データを格納する領域のポインタ

   第 2 引数 size : 入力データを格納する領域のサイズ (バイト数)

   戻り値 : 入力データのサイズ (バイト数)

   改行が入力されるまで入力を受け付ける。

   戻り値に入力データの大きさ (バイト数)

   このシステムコールは、改行が入力されるまで固まるので、あまりお勧めしない。

6. シリアル通信速度設定 sio_speed

   SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int sio_speed(int speed, int freq)
```

第 1 引数 speed：シリアル通信速度［bps］
第 2 引数 freq：システムクロック［HZ］
戻り値：設定成功 0、設定失敗 -1
シリアル通信速度は、以下の通信速度の指定のみ有効である
1200/2400/4800/9600/19200/31250/38400/57600 ［bps］
それ以外の通信速度を指定すると、設定は失敗する。

7. 16 進数文字列を数値に変換 atohex
   SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int atohex(char *string)
```

   第 1 引数 string：16 進数文字列のポインタ
   戻り値：変換された数値、失敗のときは -1

8. LCD 表示の初期化 lcd_clear
   SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

void lcd_clear()
```

   引数なし

9. LCD の初期設定 lcd_setup
   H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

void lcd_setup(int row, int column, volatile char* port, char* bits)
```

第 1 引数 row：LCD の縦文字数

第 2 引数 column：LCD の横文字数

第 3 引数 port：LCD が接続されている I/O ポートのポインタ

第 4 引数 bits：LCD の端子とポートのビット位置情報の配列のポインタ

設定例：PB ポートに接続された 20x4 文字 LCD の初期設定

```
unsigned char bits[] = {0x80, 0x40, 0x20, 0x02, 0x01, 0x08, 0x04};

PBDDR = 0xef;

lcd_setup(4, 20, &PBDR, bits);
```

bits は、7 要素の文字配列で、それぞれ以下の信号のビット位置を記述する。

配列 bits の 0 番目：RS

配列 bits の 1 番目：RW

配列 bits の 2 番目：CS

配列 bits の 3 番目：D7/D3

配列 bits の 4 番目：D6/D2

配列 bits の 5 番目：D5/D1

配列 bits の 6 番目：D4/D1

LCD は 4 ビットデータバスで I/O ポートに接続する。

10. LCD の初期設定 lcd_setupw

SH2-7045 対応

```
#include <h8/syscall.h>

void lcd_setupw(int row, int column, volatile short* port, short* bits)
```

第 1 引数 row：LCD の縦文字数

第 2 引数 column：LCD の横文字数

第 3 引数 port：LCD が接続されている I/O ポートのポインタ

第 4 引数 bits：LCD の端子とポートのビット位置情報の配列のポインタ

11. 16 ビットタイマーの割り込みハンドラの登録 timer?_regist

    H8-3067/3068/3069 対応

```
#include <h8/syscall.h>

int timer1_regist(void (*proc)(), unsigned int count)
int timer2_regist(void (*proc)(), unsigned int count)
int timer3_regist(void (*proc)(), unsigned int count)
```

    第 1 引数 proc : 割り込みハンドラの先頭アドレス

    第 2 引数 count : タイマーの割り込みカウンタ数

    16 ビットタイマー 1 番/2 番/3 番 に対する割り込みをユーザーで使用できる。

    16 ビットタイマー 0 番は、システムで使用するのでユーザーは使用できない。

    タイマーのカウントは、クロック× 8192 のタイミングで行う。

    例えば、クロックが 20MHz の場合のカウントは、0.4096 ミリ秒ごとになる。

    従って、割り込み周期は、0.4096 と count の積の時間間隔となる。

12. 16 ビットタイマーの割り込みハンドラの解除 timer?_unregist

    H8-3067/3068/3069 対応

```
#include <h8/syscall.h>

int timer1_unregist()
int timer2_unregist()
int timer3_unregist()
```

    timer?_regist で登録した割り込みハンドラを解除する。

13. 時間待ち sleep

    SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int sleep(int time)
```

第 1 引数 time : 待ち時間を 0.1 秒単位で指定する
待ち時間は、クロックが 20MHz のときに正確に計時する。
それ以外のクロックの場合は、待ち時間とクロック周波数は反比例する。

14. NIC のバッファへの書き込み ether_write
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_write(char *data, int offset, int size)
```

第 1 引数 data : 書き込むデータのポインタ
第 2 引数 offset : NIC のバッファの先頭のオフセット
第 3 引数 size : 書き込むデータの大きさ (バイト)
戻り値 : 成功のときは 0、失敗のときは -1
NIC のバッファに書き込むだけでパケット送信は行われない。
パケット送信は、ether_flush で行う。
H8/OS の IP ドライバを使用する場合は使用しないこと。

15. NIC のバッファからの読み込み ether_read
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_read(char *data, int offset, int size)
```

第 1 引数 data : 読み込むデータのポインタ
第 2 引数 offset : NIC のバッファの先頭のオフセット
第 3 引数 size : 読み込むデータの大きさ (バイト)
戻り値 : 読み込んだデータの大きさ (バイト)、失敗のときは -1
ether_get で受信したパケットのデータを NIC のバッファから読み込む。
H8/OS の IP ドライバを使用する場合は使用しないこと。

16. ネットワークへパケットを送信する ether_flush
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_flush(int size)
```

    第 1 引数 size : 送信するパケットの大きさ (バイト)
    戻り値 : 送信されたパケットの大きさ、失敗のときは -1
    H8/OS の IP ドライバを使用する場合は使用しないこと。

17. ネットワークからパケットを受信する ether_get
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_get()
```

    戻り値 : 受信されたパケットの大きさ、失敗のときは -1
    H8/OS の IP ドライバを使用する場合は使用しないこと。

18. NIC のセットアップ ether_setup
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_setup()
```

    戻り値 : NIC が有効でセットアップされたとき 0、失敗のときは -1
    NIC 関連のシステムコールを使用する場合は ether_setup で初期化すること。

19. NIC の状態を取得する get_ether_status
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int get_ether_status()
```

戻り値：NIC が有効 1、NIC が無効 0

20. NIC の mac アドレスを取得する get_mac_address
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

char* get_mac_address()
```

戻り値：mac アドレスの領域のポインタ
mac アドレスは、6 バイトの文字配列で格納されている。
NIC が無効のときは、「ff ff ff ff ff ff」の値となる。

21. TCP/IP のセットアップ ip_setup
    SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ip_setup(unsigned int ip, unsigned int netmask)
```

第 1 引数 ip：IP アドレス
第 2 引数 netmask：ネットマスク
TCP/IP ドライバの初期化を行い、IP アドレスとネットマスクを設定する。
ip_setup を使う場合は、NIC 関連のシステムコールを用いてはならない。
設定例：IP アドレスを 192.168.0.1、ネットマスクを 255.255.255.0 に設定

```
ip_setup(IPADDR(192,168,0,1), IPADDR(255,255,255,0));
```

22. ゲートウェイの設定 set_gateway
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int set_gateway(unsigned int ip)
```

第 1 引数 ip : ゲートウェイの IP アドレス
ローカル LAN の環境では設定は不要です。
ルータを通して他の LAN と接続する場合やマイコンをインターネットに接続する場合は、ゲートウェイとなるルータを指定してください。
set_gateway を使う場合は、NIC 関連のシステムコールを用いてはならない。
設定例 : ゲートウェイの IP アドレスを 192.168.0.2 に設定

```
set_gateway(IPADDR(192,168,0,2));
```

23. ネットワークに接続されたマシンの mac アドレスを IP アドレスから取得 query_mac
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

char* query_mac(unsigned int ip)
```

第 1 引数 ip : mac アドレスを取得したいマシンの IP アドレス
戻り値 : mac アドレスの領域のポインタ、取得できないときは 0
mac アドレスは、6 バイトの文字配列で格納されている。
使用例 : IP アドレス 192.168.0.10 のマシンの mac アドレスを得る

```
char *mac;

mac = query_mac(IPADDR(192,168,0,10));
```

24. 数値を 10 進数で出力する write_decimal
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int write_decimal(unsigned int dec)
```

第 1 引数 dec : 出力する数値

25. 10 進数文字列を数値に変換 atodec
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int atodec(char *string)
```

第 1 引数 string : 10 進数文字列のポインタ
戻り値 : 変換された数値、失敗のときは -1

26. 設定されている IP アドレスとネットマスクを得る get_ipinfo
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int get_ipinfo(unsigned int *ip, unsigned int *netmask);
```

第 1 引数 ip : IP アドレスを格納する変数のポインタ
第 2 引数 netmask : ネットマスクを格納する変数のポインタ
戻り値 : IP 設定が有効 1、IP 設定が無効 0

27. EEPROM に mac アドレスを設定 ether_setmac
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_setmac(char *mac)
```

第 1 引数 mac : 設定したい mac アドレスのポインタ
戻り値 : 設定された場合は 0、失敗の場合は -1

28. EEPROM の mac アドレスを取得 ether_getmac
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_getmac(char *mac)
```

第 1 引数 mac : mac アドレスを格納する領域のポインタ
戻り値 : 取得できた場合は 0、失敗の場合は -1

29. 受信した UDP データを読み込む udp_read
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int udp_read(char *data, int size)
```

第 1 引数 data : 読み込む領域のポインタ
第 2 引数 size : 読み込むデータの大きさ (バイト)
戻り値 : 読み込んだデータの大きさ (バイト)、失敗の場合は -1
このシステムコールは、udp_regport で登録されたハンドラ内で使用する。

30. UDP データを送信する udp_write

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int udp_write(char *data, int size)
```

第 1 引数 data : 送信するデータのポインタ

第 2 引数 size : 送信するデータの大きさ (バイト)

戻り値 : 送信したデータの大きさ (バイト)、失敗の場合は -1

udp_regport で登録されたハンドラ内で使用した場合、UDP パケットを送信した相手に対してパケットを送信する。

もし、送信先を変更したい場合は、set_udpaddr で変更することができる。

udp_regport で登録されたハンドラ外で使用する場合、あらかじめ set_udpaddr で送信先を指定すること

31. UDP データを受信する割り込みハンドラを登録する udp_regport

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int udp_regport(int (*proc)(unsigned short port, struct sockaddr_in *addr))
```

第 1 引数 proc : 登録する割り込みハンドラ

戻り値 : 割り込みハンドラの id、失敗の場合は -1

割り込みハンドラの第 1 引数 port : ターゲットのポート番号

割り込みハンドラの第 2 引数 addr : 送信元の IP アドレスとポート番号の構造体のポインタ

割り込みハンドラの戻り値 : パケットを受理した場合は 0、受理しない場合は -1 を返すこと

割り込みハンドラでは、ポート番号にかかわらず UDP データを受信した都度、呼び出される。

ポート番号からパケットを受理するかしないかは、割り込みハンドラ内で判断する。

割り込みハンドラでは、戻り値としてパケットを受理した場合は 0、受理しない場合は -1 を返すこと

送信元の IP アドレスは addr-＞sin_addr、ポート番号は addr-＞sin_port に格納される。

32. UDP データを受信する割り込みハンドラを解除する udp_unregport

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int udp_unregport(int (*proc)(unsigned short, struct sockaddr_in*))
```

第 1 引数 proc : 登録を解除する割り込みハンドラ

戻り値 : 成功の場合は 0、失敗の場合は -1

33. UDP データ受信の割り込みハンドラの登録状態を保存 save_regport

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

void save_regport()
```

\item UDP データ受信の割り込みハンドラの登録状態を回復 recover_regport

SH2-7045/H8-3067/3068/3069/3048/3052 対応

\begin{verbatim}

```
#include <h8/syscall.h>

void recover_regport()
```

save_regport で保存した状態に戻す。

34. UDP データの送信先を指定する set_udpaddr

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

void set_udpaddr(struct sockaddr_in *addr, unsigned short port)
```

第 1 引数 addr : 送信先の IP アドレスとポート番号の構造体のポインタ

第 2 引数 port : 送信元のポート番号

送信先の IP アドレスは addr.sin_addr、ポート番号は addr.sin_port で指定する。

送信元のポート番号を指定しない場合は 0 とする。

その場合は、適当なエメフェラルポートが割り当てられる。

35. TCP データを受信する割り込みハンドラを登録する tcp_regport

HSH2-7045/8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcp_regport(int (*proc)(unsigned short port, struct sockaddr_in
```

*addr, unsigned char flag, int seq))

第 1 引数 proc : 登録する割り込みハンドラ

戻り値 : 割り込みハンドラの id、失敗の場合は -1

割り込みハンドラの第 1 引数 port : ターゲットのポート番号

割り込みハンドラの第 2 引数 addr : 送信元の IP アドレスとポート番号の構造体のポインタ

割り込みハンドラの第 3 引数 flag : 送信元の TCP パケットのセッションフラグ

割り込みハンドラの第 4 引数 seq : 送信元の TCP パケットのシーケンス番号

割り込みハンドラの戻り値 : パケットを受理した場合は 0、受理しない場合は -1 を返すこと

割り込みハンドラでは、ポート番号にかかわらず TCP データを受信した都度、呼び出される。

ポート番号からパケットを受理するかしないかは、割り込みハンドラ内で判断する。

割り込みハンドラでは、戻り値としてパケットを受理した場合は 0、受理しない場合は -1 を返すこと

送信元の IP アドレスは addr-＞sin_addr、ポート番号は addr-＞sin_port に格納される。

この割り込みハンドラでは、単に TCP のパケットを受け取るだけで、TCP コネクションをサポートしない。

ここでは、TCP コネクション要求を受け付け、tcpsocket で TCP コネクションを処理する割り込みハンドラを登録する。

セッションフラグは、以下のような種類がある。

緊急フラグ：EXP

有効応答フラグ：ACK

プッシュフラグ：PSH

リセットフラグ：RST

同期フラグ：SYN

最終処理フラグ：FIN

通常、TCP ソケットでコネクションを確立するには、このハンドラで以下の判定をする。

(1) ポート番号が対象となる番号の場合以外は、受理しない。

(2) セッションフラグが同期フラグ以外の場合は、受理しない。

36. TCP データを受信する割り込みハンドラを解除する tcp_unregport

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcp_unregport(int (*proc)(unsigned short, struct sockaddr_in*, int))
```

第 1 引数 proc：登録を解除する割り込みハンドラ

戻り値：成功の場合は 0、失敗の場合は -1

37. TCP データの送信先を指定する set_tcpaddr

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

void set_tcpaddr(struct sockaddr_in *addr, unsigned short port)
```

第 1 引数 addr : 送信先の IP アドレスとポート番号の構造体のポインタ
第 2 引数 port : 送信元のポート番号
送信先の IP アドレスは addr.sin_addr、ポート番号は addr.sin_port で指定する。
送信元のポート番号を指定しない場合は 0 とする。
その場合は、適当なエメフェラルポートが割り当てられる。

38. 受信した TCP データを読み込む tcp_read
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcp_read(char *data, int size)
```

第 1 引数 data : 読み込む領域のポインタ
第 2 引数 size : 読み込むデータの大きさ (バイト)
戻り値 : 読み込んだデータの大きさ (バイト)、失敗の場合は -1
このシステムコールは、tcpsocket で登録されたハンドラ内で使用する。

39. TCP データを送信する tcp_write
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcp_write(int socket, char *data, int size)
```

第 1 引数 socket : ソケット番号
第 2 引数 data : 送信するデータのポインタ
第 3 引数 size : 送信するデータの大きさ (バイト)

戻り値：送信したデータの大きさ (バイト)、失敗の場合は -1

tcpsocket で登録されたハンドラ内では、1 回以内の送信とすること

tcpsocket で登録されたハンドラ外で使用する場合、あらかじめ set_tcpaddr で送信先を指定すること

40. TCP コネクションを処理する割り込みハンドラを登録する tcpsocket

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcpsocket(int seq, int (*proc)(int flag, int socket,
```

struct sockaddr_in *addr))

第 1 引数 seq：シーケンス番号

第 2 引数 proc：登録する割り込みハンドラ

戻り値：ソケット番号、失敗の場合は -1

割り込みハンドラの第 1 引数 flag：TCP コネクションの種別

割り込みハンドラの第 2 引数 socket：ソケット番号

割り込みハンドラの第 3 引数 addr：送信元の IP アドレスとポート番号の構造体のポインタ

tcpsocket は、tcp_regport ハンドラ内で登録をする。

相手からの接続要求に応じる場合は、シーケンス番号に tcp_regport ハンドラの第 3 引数を指定する。

このマシンから接続要求をする場合は、シーケンス番号をゼロを指定する。

割り込みハンドラでは、TCP コネクションにおけるシーケンス番号パケットを受信した都度、呼び出される。

割り込みハンドラでは、以下のイベントで呼び出され、種別は flag で判別する。

TCP_DATA：TCP データの受信

TCP_ACK：直前に送信した TCP データに対しての応答

TCP_CLOSE：相手からのコネクション切断要求

TCP_BEGIN：TCP コネクションが確立

flag の内容が TCP_DATA のときのみ tcp_read でデータを読み込むことができる。

flag の内容が TCP_DATA/TCP_ACK/TCP_BEGIN のときのみ tcp_write でデータを送信することができる。

送信元の IP アドレスは addr-＞sin_addr、ポート番号は addr-＞sin_port に格納される。

41. TCP コネクションの切断 tcpclose

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int tcpclose(int socket)
```

第 1 引数 socket：ソケット番号

戻り値：ソケット番号、失敗の場合は -1

tcpclose は、tcpsocket で登録されたハンドラ内で使用すること

42. NIC の初期設定 ether_init

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ether_init(int addr)
```

第 1 引数 addr：NIC の先頭 I/O ポートアドレス

戻り値：NIC が有効で初期設定されたとき 0、失敗のときは -1

43. IDE ディスクインターフェースの初期化

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ata_setup(int addr1, int addr2)
```

第 1 引数 addr1 : CS0 でアクセスする ATA レジスタの先頭アドレス

第 1 引数 addr2 : CS1 でアクセスする ATA レジスタの先頭アドレス

戻り値 : IDE ドライブ接続情報、IDE ドライブが失敗のときは -1

通常は、以下のように指定する。

```
ata_setup(0x600000, 0x600020);
```

IDE ドライブ接続情報は、マスターのみ”1”、スレーブのみ”2”、両方”3”

44. 割り込みハンドラの登録

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int int_regist(int num, void (*proc)())
```

第 1 引数 num : 割り込みベクター番号

第 2 引数 proc : 割り込みハンドラ

戻り値 : 割り込みハンドラの登録されたとき 0、失敗のときは -1

H8 の割り込みを全てサポートします。

H8 の割り込みベクターは全て H8/OS で管理していますので、ユーザーでは設定しないでください。

H8 の割り込みを使いたい場合は、int_regist で登録してください。

すでに int_regist で登録された割り込みベクターは、int_unregist で解除されるまで使用できません。

H8/3067、H8/3068 では、H8_OS で 36,38,40,42 番の割り込みベクターを占有していますのでユーザーで使用できません。

H8/3048 では、H8_OS で 20 番の割り込みベクターを占有していますのでユーザーで使用できません。

もし、どうしても使用したい場合は、int_unregist で解除可能ですが、H8/OS の動作はそれ以後、保証されません。

45. 割り込みハンドラの登録解除

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int int_unregist(int num)
```

第 1 引数 num : 割り込みベクター番号

戻り値 : 割り込みハンドラの登録解除されたとき 0、失敗のときは -1

46. ハードディスクのドライブ選択

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ata_sel_drive(int drive)
```

第 1 引数 drive : ドライブ番号でマスターは”0”、スレーブは”1”

戻り値 : IDE ドライブ選択成功のとき 0、失敗のときは -1

47. IDE ディスクインターフェースのリセット

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ata_reset()
```

戻り値 : IDE ドライブがリセットされたとき 0、未接続のとき -1

48. データ転送なしの ATA コマンドの実行

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int non_data_command(short command, int count, int lba)
```

第 1 引数 command : ATA コマンドコード
第 2 引数 count : セクター数
第 3 引数 lba : 先頭 LBA(Logical Block Address) 番号
戻り値 : コマンドが成功したとき 0、失敗のときは -1
ATA コマンドコードで以下は定義済
o SetMultiple
セクター数、LBA 指定が不要なコマンドは適当な値を指定する。

49. ハードディスクドライブ詳細情報を取得
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ata_identify(short *data)
```

第 1 引数 data : 詳細情報を格納する領域のポインタ
戻り値 : 詳細情報が格納されたとき 0、失敗のときは -1
詳細情報の大きさは 512 バイト。

50. データ転送 (読み込み) の ATA コマンドの実行
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int data_in_command(short command, char *data, int count, int lba)
```

第 1 引数 command : ATA コマンドコード

第 2 引数 data : データを格納する領域のポインタ

第 3 引数 count : セクター数

第 4 引数 lba : 先頭 LBA(Logical Block Address) 番号

戻り値 : コマンドが成功したとき 0、失敗のときは -1

ATA コマンドコードで以下は定義済

o IdentifyDevice

o ReadMultiple

o ReadSector

セクター数、LBA 指定が不要なコマンドは適当な値を指定する。

51. データ転送 (書き込み) の ATA コマンドの実行

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int data_out_command(short command, char *data, int count, int lba)
```

第 1 引数 command : ATA コマンドコード

第 2 引数 data : 書き込むデータを格納されている領域のポインタ

第 3 引数 count : セクター数

第 4 引数 lba : 先頭 LBA(Logical Block Address) 番号

戻り値 : コマンドが成功したとき 0、失敗のときは -1

ATA コマンドコードで以下は定義済

o WriteMultiple

o WriteSector

セクター数、LBA 指定が不要なコマンドは適当な値を指定する。

52. ハードディスクドライブ主要情報を取得

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int ata_driveinfo(char *name, int *chs)
```

第 1 引数 name : ハードディスク名称を格納する領域のポインタ (40 バイト)
第 2 引数 chs : ヘッド数、シリンダ数、セクター数を格納する領域のポインタ
戻り値 : 情報が格納されたとき 0、失敗のときは -1
第 2 引数 chs は、chs

0

がヘッド数、chs

1

がシリンダ数、chs

2

がセクター数となります。

53. ハードディスクドライブの容量を取得
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int get_hddsize()
```

戻り値 : 情報が格納されたとき 0、失敗のときは -1
ハードディスクドライブの容量は、セクター数を取得することになります。
ハードディスクドライブの容量をバイト数は、セクター数に 512 をかけた値になります。

54. PS/2 キーボードの初期化 ps2_setup
H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int ps2_setup(volatile char *port, char *bits)
```

第 1 引数 port：PS/2 キーボードが接続されているポートアドレス

第 2 引数 bits：PS/2 の端子とポートのビット位置情報の配列のポインタ

設定例：P4 ポートに接続された PS/2 キーボードの初期設定

```
unsigned char bits[] = {0x80, 0x40, 0x20};

P4DDR = 0xc0;

ps2_setup(&P4DR, bits);
```

bits は、3 要素の文字配列で、それぞれ以下の信号のビット位置を記述する。

配列 bits の 0 番目：CLOCK 出力

配列 bits の 1 番目：DATA 出力

配列 bits の 2 番目：DATA 入力

なお、PS/2 の CLOCK 入力は NMI に接続し、NMI は立ち下がりエッジに設定しておくこと

55. PS/2 キーボードの初期化 ps2_setupw

SH2-7045 対応

```
#include <h8/syscall.h>

int ps2_setupw(volatile short *port, short *bits)
```

第 1 引数 port：PS/2 キーボードが接続されているポートアドレス

第 2 引数 bits：PS/2 の端子とポートのビット位置情報の配列のポインタ

56. キーボードからの読み込み keyread

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int keyread(char *buffer)
```

第 1 引数 buffer : キーボードからの文字を格納するためのポインタ
戻り値 : 情報が格納されたとき 1、キーが押されないとき 0
キーボードからの文字を 1 バイトだけ読みこみます。
キーが押されないとき 0 が戻り値となります。

57. NMI 割り込みエッジの設定 nmi_edge
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

void nmi_edge(int edge)
```

第 1 引数 edge : 立ち下がりは DownEdge、立ち上がりは UpEdge と指定します。

58. 入力モード指定 read_mode
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int read_mode(int mode)
```

第 1 引数 mode : 入力モード指定
write_mode(SIO) : シリアル端末入力にする
write_mode(PS2) : PS/2 キーボード入力にする

59. I2C シリアル EEPROM の初期化 eep_setup
H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

void eep_setup(volatile char *dr_reg, volatile char *ddr_reg, char *bits)
```

第 1 引数 dr_reg：I2C の SEEPROM が接続されている I/O ポートのポインタ

第 2 引数 ddr_reg：I2C の SEEPROM が接続されている I/O の制御レジスタのポインタ

第 3 引数 bits：I2C の SEEPROM 端子とポートのビット位置情報の配列のポインタ

設定例：P6 ポートに接続された 2C の SEEPROM 初期設定

```
unsigned char bits[] = {0x02,0x01};

eep_setup(&P6DR, &P6DDR, bits);
```

bits は、2 要素の文字配列で、それぞれ以下の信号のビット位置を記述する。

配列 bits の 0 番目：CLOCK

配列 bits の 1 番目：DATA

60. I2C シリアル EEPROM の初期化 eep_setupw

SH2-7045 対応

```
#include <h8/syscall.h>

void eep_setupw(volatile short *dr_reg, volatile short *ddr_reg, short *bits)
```

第 1 引数 dr_reg：I2C の SEEPROM が接続されている I/O ポートのポインタ

第 2 引数 ddr_reg：I2C の SEEPROM が接続されている I/O の制御レジスタのポインタ

第 3 引数 bits：I2C の SEEPROM 端子とポートのビット位置情報の配列のポインタ

61. I2C シリアル EEPROM への書き込み eep_write

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int eep_write(int dest, char *src, int size)
```

第 1 引数 dest：書き込む先の SEEPROM アドレス
第 2 引数 src：書き込む元のデータのポインタ
第 3 引数 size：書き込む元のデータの大きさ
戻り値：書き込んだバイト数、失敗の場合は-1

62. I2C シリアル EEPROM からの読み込み eep_read
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int eep_read(char *dest, int src, int size)
```

第 1 引数 dest：読み込む先のデータのポインタ
第 2 引数 src：読み込む元の SEEPROM アドレス
第 3 引数 size：読み込む先のデータの大きさ
戻り値：読み込んだバイト数、失敗の場合は-1

63. FAT 上のディレクトリ一覧取得の初期化 dirent_init
H8-3068/3069 対応

```
#include <h8/syscall.h>

int dirent_init(char *path, dirent *dir)
```

第 1 引数 path：対象となるディレクトリ名
第 2 引数 dir：dirent 構造体のポインタ
戻り値：対象となるディレクトリのファイル数、失敗の場合は-1

64. FAT 上のディレクトリ一覧取得 dirent_get
H8-3068/3069 対応

```
#include <h8/syscall.h>

int dirent_get(dirent *dir)
```

第 1 引数 dir : dirent 構造体のポインタ
戻り値 : 情報取得時は、0、失敗の場合は-1

65. FAT 上にディレクトリを新規作成 makedir
H8-3068/3069 対応

```
#include <h8/syscall.h>

int makedir(char *path)
```

第 1 引数 path : 新規作成するディレクトリ名
戻り値 : 成功の場合は 0、失敗の場合は-1

66. FAT 上のディレクトリ/ファイルを削除 delete
H8-3068/3069 対応

```
#include <h8/syscall.h>

int delete(char *path)
```

第 1 引数 path : 削除するディレクトリ/ファイル名
戻り値 : 成功の場合は 0、失敗の場合は-1

67. FAT 上のファイルを開く openfile
H8-3068/3069 対応

```
#include <h8/syscall.h>

int openfile(char *path, int opt)
```

第 1 引数 path : オープンするファイル名
第 2 引数 opt : オプション
戻り値 : 成功の場合はファイルディスクプリタ、失敗の場合は-1
第 2 引数 opt は、FILEREAD、FILEWRITE、FILEAPPEND の 3 つがあります。

68. FAT 上のファイルの内容を読みこむ readfile
H8-3068/3069 対応

```
#include <h8/syscall.h>

int readfile(char *data, int size, int fd)
```

第 1 引数 data : ファイルの内容を読みこむ先のポインタ
第 2 引数 size : ファイルの内容を読みこむ最大サイズ
第 3 引数 fd : ファイルディスクプリタ
戻り値 : 読みこんだデータのバイト数

69. FAT 上のファイルの内容を書きこむ writefile
H8-3068/3069 対応

```
#include <h8/syscall.h>

int writefile(char *data, int size, int fd)
```

第 1 引数 data : ファイルの内容を書き込む元のポインタ
第 2 引数 size : ファイルの内容を書き込む最大サイズ
第 3 引数 fd : ファイルディスクプリタ
戻り値 : 書き込んだデータのバイト数

70. FAT 上のファイル名を変更する renfile
H8-3068/3069 対応

```
#include <h8/syscall.h>

int renfile(char *path1, char *path2);
```

第 1 引数 path1 : 変更前のファイル名
第 2 引数 path2 : 変更後のファイル名
戻り値 : 成功の場合は 0、失敗の場合は-1

71. FAT 上のファイルを閉じる closefile
H8-3068/3069 対応

```
#include <h8/syscall.h>

int closefile(int fd)
```

第 1 引数 fd : 閉じる対象のファイルディスクプリタ
戻り値 : 成功の場合は 0、失敗の場合は-1

72. FAT 上のファイル長の取得 get_file_size
H8-3068/3069 対応

```
#include <h8/syscall.h>

int get_file_size(int fd)
```

第 1 引数 fd : 取得対象のファイルディスクプリタ
戻り値 : 成功の場合は 0、失敗の場合は-1

73. フォーマット標準出力 printf
SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int printf(char*, ... )
```

基本的に標準の printf 関数に準拠しています。

ただし、浮動小数はサポートしていません。

既存の C ライブラリにくらべて、H8/OS の printf は RAM 領域を消費しないというメリットがあります。

74. フォーマット出力 sprintf

SH2-7045/H8-3067/3068/3069/3048/3052 対応

```
#include <h8/syscall.h>

int sprintf(char*, char*, ... )
```

基本的に標準の sprintf 関数に準拠しています。

ただし、浮動小数はサポートしていません。

既存の C ライブラリにくらべて、H8/OS の sprintf は RAM 領域を消費しないというメリットがあります。

75. SEEPROM ディスクを FAT でフォーマット format

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int format(char *path)
```

第 1 引数 path : デバイスファイル名

戻り値 : 成功の場合は 0、失敗の場合は-1

76. RTC(EPSON R8564) の初期化 rtc_setup

H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

void rtc_setup(volatile char *dr_reg, volatile char *ddr_reg, char *bits)
```

第 1 引数 dr_reg：RTC が接続されている I/O ポートのポインタ
第 2 引数 ddr_reg：RTC が接続されている I/O の制御レジスタのポインタ
第 3 引数 bits：RTC 端子とポートのビット位置情報の配列のポインタ
設定例：P6 ポートに接続された RTC 初期設定

```
unsigned char bits[] = {0x02,0x01};

rtc_setup(&P6DR, &P6DDR, bits);
```

bits は、2 要素の文字配列で、それぞれ以下の信号のビット位置を記述する。
配列 bits の 0 番目：CLOCK
配列 bits の 1 番目：DATA

77. RTC(EPSON R8564) の初期化 rtc_setupw
SH2-7045 対応

```
#include <h8/syscall.h>

void rtc_setupw(volatile short *dr_reg, volatile short *ddr_reg, short *bits)
```

第 1 引数 dr_reg：RTC が接続されている I/O ポートのポインタ
第 2 引数 ddr_reg：RTC が接続されている I/O の制御レジスタのポインタ
第 3 引数 bits：RTC 端子とポートのビット位置情報の配列のポインタ

78. RTC(EPSON R8564) への書き込み rtc_write
SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int rtc_write(int addr, int data)
```

第 1 引数 dest : 書き込む先の RTC アドレス

第 2 引数 data : 書き込む元のデータ

戻り値 : 成功の場合は 0、失敗の場合は-1

79. RTC(EPSON R8564) からの読み込み rtc_read

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int rtc_read(int addr)
```

第 1 引数 addr : 読み込む元の RTC アドレス

戻り値 : 読み込んだデータ、失敗の場合は-1

80. シリアルポートの切替え sio_change

SH2-7045/H8-3067/3068/3069/3048/3052/3664 対応

```
#include <h8/syscall.h>

int rtc_read(int sci_port)
```

sci_port はシリアルポートの番号を指定し、SCI0 なら「0」SCI1 なら「1」というように指定します。

第 1 引数 sci_port : シリアルポートの番号

戻り値 : 成功の場合は 0、失敗の場合は-1