1. #144 キーパッド 8 のプログラム例（I2C）

#144 キーパッド 8 を制御する際のホスト（マスタ）側の C 言語プログラム例を説明します。

＜リレー、LED の制御＞

次のような I2C の制御関数があるとします。BYTE 型は unsigned char 型を typedef で定義したものです。

```
void  I2CStart(BYTE opt);              // スタート・コンディション発行
void  I2CStop(void);                   // ストップ・コンディション発行
BYTE  I2CRead(BYTE noack);             // 1 バイト リード
BYTE  I2CWrite(BYTE dat);              // 1 バイト ライト
```

変数、マクロは次のように定義してあります。

```
BYTE  CBData;                          // I2C コントロール・バイト
BYTE  Cmd;                             // #144 に与えるコマンド・コード
BYTE  I2CAdrs;                         // 送信先の I2C スレーブ・アドレス
BYTE  Key;                             // 読み出したキー・コード
#define  CB_W  0                       // コントロール・バイト R/W フラグ =W
#define  CB_R  1                       // コントロール・バイト R/W フラグ =R
#define  ON  0x10                      // LED/ リレー ON
#define  OFF  0x00                     // LED/ リレー OFF
#define  LED_SEL  0x00                 // LED 制御
#define  RELAY_SEL  0x08               // リレー制御
#define  LED1  0                       // LED1
#define  LED2  1                       // LED2
#define  LED3  2                       // LED3
#define  LED4  3                       // LED4
#define  RLY1  0                       // リレー 1
#define  RLY2  1                       // リレー 2
#define  RLY3  2                       // リレー 3
```

LED1 を点灯させるには、次のようにプログラムします。

```
I2CAdrs = 0x44;                        // #144 のスレーブ・アドレス
CBData = I2CAdrs << 1;                 // コントロール・バイト (R/W フラグはブランク )
Cmd = 0xE0 ¦ LED_SEL ¦ ON ¦ LED1;      // cmd = 0xF0（表 1 参照）
I2CStart( 0 );                         // ノーマル・スタート・コンディション
I2CWrite( CBData ¦ CB_W );             // コントロール・バイト送信 (W)
I2CWrite( Cmd );                       // I2C コマンド送信
I2CStop();                             // ストップ・コンディション
```

これで、I2C バスにコマンドが送出され、それを受けた #144 ボードが作動します。
上記プログラムで送信するコマンドの内容 (Cmd 変数の設定値) を変えれば、制御する対象を変更できます。コマンド・コードは表 1 を参照してください。

LED1 を消灯させるとき

```
Cmd = 0xE0 ¦ LED_SEL ¦ OFF ¦ LED1;
```

LED2 を点灯させるとき

```
Cmd = 0xE0 ¦ LED_SEL ¦ ON ¦ LED2;
```

リレー 2 を ON させるとき

```
Cmd = 0xE0 ¦ RELAY_SEL ¦ ON ¦ RLY2;
```

＜キー・パッドの読み出し＞

キー・パッドの状態を読み出すには次のようにプログラムします。

```
I2CAdrs = 0x44;                        // #144 のスレーブ・アドレス
CBData = I2CAdrs << 1;                 // コントロール・バイト (R/W フラグはブランク )
I2CStart(0);                           // ノーマル・スタート・コンディション
I2CWrite(CBData ¦ CB_R);               // コントロール・バイト送信 (R)
key = I2CRead(1);                      // キー・ステータス読み出し (NOACK で応答 )
I2CStop();                             // ストップ・コンディション
```

これで、キー・パッドの状態が読み出せます。キーがどれも押されていないときは 0xFF が返ります。それ以外は押されたキーに応じて 0 ～ 7 の値が返ります。

この読み出し処理を 20 ～ 50ms 程度の周期で実行して、チャタリング防止処理やリピート処理に渡せば、通常のキー・マトリクスからセンス・データを取り込んで処理するのと同様に扱えます。

表 1 #144 のコマンド・コード

#144 キーパッド8										
コマンド名	R/W	b7	b6	b5	b4	b3	b2	b1	b0	説 明
リレー制御	W	1	1	1	RON	1	0	RY#		RY#: リレー番号 0～3 RON 1:リレーON
LED制御	W	1	1	1	LON	0	LED#			LED#: LED番号 0～7 LON 1: LED点灯
（入力データ）	R	KEY								KEY: キー入力コード 0～7, 0xFF

2. #134/157 4ch ディジタル調光器のプログラム例（I2C）

#134 ディジタル調光器を制御する際のホスト（マスタ）側の C 言語プログラム例を説明します（#157 と共用）。調光器では、チャネル番号と明るさデータを 1 バイトのデータとして送信するだけですが、明るさデータは 0 のときは OFF、1 のときが最も明るく、60 が最も暗い（ほとんど OFF）というようになっています。1 ～ 60 の値は、ゼロクロス・ポイントからトライアックのゲート信号を ON するまでの時間を指定しています。

次のような I2C の制御関数があるとします。BYTE 型は unsigned char 型を typedef で定義したものです。

```
void  I2CStart(BYTE opt);          // スタート・コンディション発行
void  I2CStop(void);               // ストップ・コンディション発行
BYTE  I2CRead(BYTE noack);         // 1 バイト リード
BYTE  I2CWrite(BYTE dat);          // 1 バイト ライト
```

変数、マクロは次のように定義してあります。

```
BYTE  CBData;                      // I2C コントロール・バイト
BYTE  Cmd;                         // #134 に与えるコマンド・コード
BYTE  I2CAdrs;                     // 送信先の I2C スレーブ・アドレス
#define  CB_W  0                   // コントロール・バイト R/W フラグ =W
#define  CB_R  1                   // コントロール・バイト R/W フラグ =R
#define  CH0  0x00                 // チャネル 0
#define  CH1  0x40                 // チャネル 1
#define  CH2  0x80                 // チャネル 2
#define  CH3  0xC0                 // チャネル 3
```

チャネル 0 に明るさデータ 30 を設定するには、次のようにプログラムします。

```
I2CAdrs = 0x20;                    // #134 のスレーブ・アドレス
CBData = I2CAdrs << 1;             // コントロール・バイト (R/W フラグはブランク )
Cmd = CH0 ¦ 30;                    // cmd = 0x1E（表 2 参照）
I2CStart( 0 );                     // ノーマル・スタート・コンディション
I2CWrite( CBData ¦ CB_W );         // コントロール・バイト送信 (W)
I2CWrite( Cmd );                   // I2C コマンド送信（複数コマンドを連続して送信可能）
I2CStop();                         // ストップ・コンディション
```

上記プログラムで送信するコマンドの内容 (Cmd 変数の設定値) を変えれば、制御する内容を変更できます。コマンド・コードは表 2 を参照してください。なお、コマンドは複数、連続で送信することもできます。

チャネル 2 を OFF に設定する場合

```
Cmd = CH2 ¦ 0;
```

チャネル 3 に 50 を設定する場合

```
Cmd = CH3 ¦ 50;
```

表 2 #134/157 のコマンド・コード

#134 4ch ディジタル調光器										
コマンド名	R/W	b7	b6	b5	b4	b3	b2	b1	b0	説 明
明るさ設定	W	CH		VAL						CH: チャネル 0～3 VAL: 明るさデータ 1(明)～60(暗)/0 (OFF)

3. #129 6 桁 7 セグメント LED 表示器のプログラム例（I2C）

#129 数値表示器を制御する際のホスト（マスタ）側の C 言語プログラム例を説明します。なお、数値、個別 LED 表示関係のコマンドは #133 3 桁 7 セグメント LED 表示器のコマンドと、桁数指定の範囲が違う以外は互換性があります。

次のような I2C の制御関数があるとします。BYTE 型は unsigned char 型を typedef で定義したものです。

```
void  I2CStart(BYTE opt);              // スタート・コンディション発行
void  I2CStop(void);                   // ストップ・コンディション発行
BYTE  I2CRead(BYTE noack);             // 1 バイト リード
BYTE  I2CWrite(BYTE dat);              // 1 バイト ライト
```

変数、マクロは次のように定義してあります。

```
BYTE  CBData;                          // I2C コントロール・バイト
BYTE  Cmd;                             // #129 に与えるコマンド・コード
BYTE  I2CAdrs;                         // 送信先の I2C スレーブ・アドレス
#define  CB_W  0                       // コントロール・バイト R/W フラグ =W
#define  CB_R  1                       // コントロール・バイト R/W フラグ =R
#define  CMD_CLEAR  0x80               // データ・クリア・コマンド
#define  CMD_DATASET  0x00             // データ設定コマンド
#define  CMD_DSPOPT  0xA0              // 表示切り替えコマンド
#define  CMD_LEDSET  0xC0              // 個別 LED 設定コマンド
#define  CPM0  0x00                    // データ設定コマンド パラメータ
#define  CPM1  0x10                    // データ設定コマンド パラメータ
#define  CPM2  0x20                    // データ設定コマンド パラメータ
#define  CPM3  0x30                    // データ設定コマンド パラメータ
#define  BLK  0x10                     // 個別 LED 設定コマンド パラメータ
#define  ON  0x80                      // 個別 LED 設定コマンド パラメータ
#define  DSP1  0x02                    // 表示切り替えコマンド パラメータ
#define  DSP2  0x01                    // 表示切り替えコマンド パラメータ
#define  BLK1  0x04                    // 表示切り替えコマンド パラメータ
```

数値表示に左桁から "12A" と表示させる場合の例を示します。データ設定コマンド（表示データ）は CPM=0（省略）に設定されているため、数値設定（表示）後にカレント・ポインタを右に 1 つ移動させます。つまり連続して表示データを送信すると、左から順番に数字が表示されていきます。

```
I2CAdrs = 0x10;                        // #129 のスレーブ・アドレス
CBData = I2CAdrs << 1;                 // コントロール・バイト (R/W フラグはブランク )
I2CStart( 0 );                         // ノーマル・スタート・コンディション
I2CWrite( CBData ¦ CB_W );             // コントロール・バイト送信 (W)
Cmd = CMD_CLEAR;                       // データ・クリア・コマンド（表示クリア）（表 3 参照）
I2CWrite( Cmd );                       // I2C コマンド送信
Cmd = 1;                               // データ設定コマンド（表示データ）（表 3 参照）
I2CWrite( Cmd );                       // I2C コマンド送信
Cmd = 2;                               // データ設定コマンド（表示データ）（表 3 参照）
I2CWrite( Cmd );                       // I2C コマンド送信
Cmd = 0xA;                             // データ設定コマンド（表示データ）（表 3 参照）
I2CWrite( Cmd );                       // I2C コマンド送信（複数コマンドを連続して送信可能）
I2CStop();                             // ストップ・コンディション
```

送信コマンドを変えることで、カレント・ポインタの位置を指定したり、個別 LED を点消灯、点滅させるなどの操作が可能です。コマンド・コードの詳細は表 3 を参照してください。

個別 LED 3 を点滅させるとき

```
Cmd = CMD_LEDSET ¦ BLK ¦ ON ¦ 3;       // 個別 LED 設定コマンド
```

数値表示を点滅させるとき

```
Cmd = CMD_DSPOPT ¦ BLK1 ¦ DSP2 ¦ DSP1;
                                       // 表示切り替えコマンド
```

表 3 #129 のコマンド・コード

#129 6桁 7セグメントLED表示器										
コマンド名	R/W	b7	b6	b5	b4	b3	b2	b1	b0	説 明
データ設定	W	0	0	CPM(0～3)		VAL(0～F)				VAL: 表示する数字(16進数1桁) CPM 0: 設定後カレント・ポインタ右移動 CPM 1: 設定後カレント・ポインタ左移動 CPM 2: マイナス表示の設定後右移動 CPM 3: カレント・ポインタのデータをクリア
カレント・ポインタ設定	W	0	1	0	0	0	CPOS(0～5)			CPOS: 数値表示の位置指定 0が左端
小数点位置設定	W	0	1	1	EX	DP	DPOS(0～5)			DPOS: 小数点の位置指定 0が左端 DP: 1: 小数点表示 EX 1: 排他表示
データ・クリア	W	1	0	0	0	0	CTP(0～3)			CTP 0: 数値表示をブランク(小数点含む) CTP 1: 最下位'0'、それ以外ブランク CTP 2: オール'0'表示 CTP 3: 個別LEDをブランク
表示切り替え	W	1	0	1	0	0	BLK1	DSP2	DSP1	DSP1 1: 数字、小数点表示 DSP2 1: 個別LED表示 BLK1 1: 全桁、小数点点滅
個別LED設定	W	1	1	0	BLK	ON	LED#(0～7)			LED#: 操作対象の個別LED番号 ON 1: 個別LED点灯 BLK 1: 個別LED点滅
リザーブ	W	1	1	1						

4. I2C 制御関数

I2C マスタのドライバとして次の 5 つの関数を用意しています。

(1) void I2CMsInit(void);. . . . . I2C マスタモードで初期化

(2) void I2CStart(BYTE opt); . . . . . スタート・コンディション発行

(3) void I2CStop(void); . . . . . ストップ・コンディション発行

(4) BYTE I2CRead(BYTE noack); . . . 1 バイト リード

(5) BYTE I2CWrite(BYTE dat); . . . . 1 バイト ライト

(1) の I2CMsInit() は、I2C 関係の初期化処理で、他の 4 つの関数を使う前に、初期化処理などでコールしておく必要があります。ハードウェア制御のドライバでは PIC MSSP が I2C モードで初期化されます。ソフトウェア制御のドライバでは I2C 関連の I/O ポートが初期化されます。ハードウェア制御かソフトウェア制御かはリンク(CCS-C の場合はインクルード) するファイルで切り替えます。

これらの関数は、書籍「マイコンの 1 線、2 線、3 線インターフェース活用入門」(CQ 出版刊) に掲載しているものですが、CCS-C のライブラリにも同等のものが用意されていますので、少しの手直しで置き換えが可能です。次に関数の引数について簡単に説明しますので、参考にしてください。

(1) void I2CMsInit(void); I2C マスタモードで初期化

引数はありません。

(2) void I2CStart(BYTE opt); スタート・コンディション発行

"opt" が "0" のときはノーマル・スタート・コンディション、"1" のときはリピートスタート・コンディションを発行します。

(3) void I2CStop(void); ストップ・コンディション発行

引数はありません。

(4) BYTE I2CRead(BYTE noack); 1 バイト リード

"noack" が "1" のときはリード後に NOACK をスレーブへ返します。読み出したデータは関数の戻り値で得られます。

(5) BYTE I2CWrite(BYTE dat); 1 バイト ライト

"dat" は出力する 8 ビットのデータです。ライト後にスレーブから送られる ACK/NOACK のビット値は関数の戻り値で得られます。戻り値が "0" のときが ACK(SDA="L") です。