025/11/12

Pico Systems Co., Ltd.

コインメック・ビルバリ エミュレータ
通信プロトコル仕様書

１．　概要

コインメック・ビルバリ エミュレータ（以後、本ドキュメント内では単にエミュレータと呼称） は自動販売機内の低速バス（VCCS バス）上でコインメックおよびビルバリをエミュレーションするボードです。使用方法は自動販売機のコインメックおよびビルバリを取り外してエミュレータを代わりに取り付けます。一般的な飲料自動販売機ではコインメックを取り外すと起動することが出来ませんが、エミュレータを取り付ける事によって起動する事が可能となります。（注：VCCS バスでは 2 台以上の同じ種類のデバイスは許可されていませんので、エミュレータを VCCS バスに接続する場合は必ず他のコインメックとビルバリを取り外す必要があります。）

エミュレータ は自動販売機の主制御機との通信をコインメック、ビルバリに代わって行います。この時、内部レジスタの値に応じて通信を行うため、実際にコインや紙幣を投入しなくても内部レジスタを変更する事で自動販売機はコインや紙幣が投入されたものとして動作します。

エミュレータ は RS-232C シリアルインターフェースを持ち、外部 PC やマイクロコントローラー（以後、本ドキュメントではホストＰＣと呼称）と接続し独自プロトコルによるコマンドで通信を行うことが出来ます。これらのコマンドは内部レジスタを読み書きするもので、これによりホストＰＣは間接的に自動販売機を制御することが可能です。

本ドキュメントではエミュレータとホストＰＣ間の通信プロトコルについて解説します。

２．　内部レジスタ

エミュレータ は以下に解説する４つの内部レジスタを持っており、ホストＰＣからすべてのレジスタは、シリアル通信により読み出しを行うことが出来ます。またいくつかのレジスタは書き込みも可能です。（これらのレジスタの値は自動販売機の主制御機に返されるデータが格納されており、書き込み可能なレジスタは自動販売機の制御に使用可能ですが、読み出し専用レジスタはあまり意味を持ちません。）

エミュレータ は常に自動販売機の主制御機と通信をしています。そのため、シリアル通信によるコマンドでレジスタの値が変更されるとその情報はエミュレータ と主制御機の次の通信で主制御機に伝わる事になります。 結果として主制御機はコイン投入枚数が変化した事や紙幣投入枚数が変化した事、返却レバーが操作された事などを知り、それに対応する動作を行う事になります。この様にしてホストＰＣは間接的に自動販売機を制御する事が可能となります。

レジスター一覧

Register Name	Length	Explanations		R/W	Remarks
Number of Coins Accepted	4 Bytes	1st Byte	Number of ¥10 coin	Read Write	2 digit BCD
		2nd Bye	Number of ¥50 coin	Read Write	2 digit BCD
		3rd Byte	Number of ¥100 coin	Read Write	2 digit BCD
		4th Byte	Number of ¥500 coin	Read Write	2 digit BCD
Coin Mech Status	1 Byte	bit 0	Enabled	Read Only	
		bit 1	(Inventory Flag)	Read Only	* can be ignored
		bit 2	(Payout Enabled)	Read Only	* can be ignored
		bit 3	Cancel Lever	Read Write	
		bit 4	(Payout Completed)	Read Only	* can be ignored
		bit 5	Transaction Cleard	Read Only	
		bit 6	(Inventry Disabled)	Read Only	* can be ignored
		bit 7	(Change Controle SW)	Read Only	* can be ignored
Number of Bills Accepted	5 Bytes	Byte 1	Number of ¥1000 bill	Read Write	can be written only '1'
		Byte 2	N/A	Read Only	
		Byte 3	N/A	Read Only	
		Byte 4	N/A	Read Only	
		Byte 5	Number of ¥2000 bill	Read Write	can be written only '1'
Bill Vali Status	3 Byte	Byte 1 bit 0	Enabled	Read Only	
		Byte 1 bit 1	(Inventory Flag)	Read Only	* can be ignored
		Byte 1 bit 2	(Another bill none)	Read Only	* can be ignored
		Byte 1 bit 3	(Payout Completed)	Read Only	* can be ignored
		Byte 1 bit 4	Transaction Cleard	Read Only	
		Byte 1 bit 5	(Detection)	Read Only	* can be ignored
		Byte 1 bit 6	(Stacking)	Read Only	* can be ignored
		Byte 1 bit 7	(Payout)	Read Only	* can be ignored
		Byte 2 bit 0	Accept ¥1000 bill	Read Only	
		Byte 2 bit 1	N/A	Read Only	Allways '0'
		Byte 2 bit 2	N/A	Read Only	Allways '0'
		Byte 2 bit 3	N/A	Read Only	Allways '0'
		Byte 2 bit 4	Accept ¥2000 bill	Read Only	
		Byte 2 bit 5	N/A	Read Only	Allways '0'
		Byte 2 bit 6	N/A	Read Only	Allways '0'
		Byte 2 bit 7	(Pulling Out detected)	Read Only	* can be ignored
		Byte 3	N/A	Read Only	Allways '0'

３．　レジスタを介した自動販売機制御の流れ

- コインメックステータス(CoinMech Status)レジスタまたはビルバリステータス(BillVali Status)レジスタの Enabled ビットは自販機が動作していて硬貨、紙幣の受け入れ中である事を示します。本ビットが立っている時にのみ Number of Coin Accepted（受入硬貨枚数）レジスタ、Number of Bill Accepted（受入紙幣枚数）レジスタに書き込むことが出来ます。

- Number of Coin Accepted レジスタ、Number of Bill Accepted register に書き込んだ値は自動販売機の投入金額に反映され、金額に応じて販売動作を継続します。

- 自販機が販売を終了しビルバリ、コインメックに投入金クリア指令を送ると CoinMech Status レジスタの Transaction Cleared ビット、BillVali Status レジスタの Transaction Cleared ビットがセットされ、一連の販売動作が終了します。

- 販売動作中、自販機とエミュレータの通信が発生する度にエミュレータ内のレジスタ値に反映されます。ただし実際の釣銭返却動作などは行われません。またレジスタに書き込んだ値は自販機に反映されますのでこれを利用して返却レバー操作なども行わせる事が出来ます。

４．　シリアル通信プロトコル

a. 通信手順

- 全ての通信はホストＰＣがコマンドを送る事により始まり、エミュレータは例外なく必ずレスポンスを返す事で一つの通信トランザクションが完了します。
- コマンドとレスポンスは後に記載するフレーム形式で送られます。
- ホストＰＣはレスポンスフレーム受信時にエラーを検出した場合、フレームは処理する事なく破棄、再度コマンドを送信する事とします。
- ホストＰＣは一度に１つのコマンドしか送信する事は出来ず、レスポンスを受信するかタイムアウト時間が経過するまで次のコマンドを送信する事は出来ません。

b. シリアル通信パラメータ

Serial Line　　: RS-232C, Async, Half Duplex
Baud Rate　　: 38,400 [bps]
Start Bit　　: 1
Data Bit　　: 8
Stop Bit　　: 1
Parity Bit　　: None

c. 通信フレーム

コマンドおよびレスポンスは以下フレームの DATAGRAM として送信されます。

STX	SID	RID	DATAGRAM	CRC	ETX

STX: Start Mark [02h]
SID: Sender ID [1 Byte Binary] 0x00: Host PC, 0x80: EMULATOR
RID: Receiver ID [1 Byte Binary] 0x00: Host PC, 0x80: EMULATOR
DATAGRAM: Data Packet (MAX: 2048 Bytes Max)
CRC: CRC-16-CCITT (from SID to DATAGRAM) [2 Byte Binary HL]
ETX: End Mark [03h]

フレーム間タイムアウト：500 [msec]

データエスケープ処理

通信フレーム内に特定のデータが現れた場合、以下のルールによってエスケープ処理を行ってから送信するものとします。従って受信フレームは本ルールに則って復号する必要があります。

- STX(0x02)または ETX(0x03)が SID から CRC の間に現れた場合、以下の通りエスケープします。（CRC 計算はエスケープ処理する前に行われるものとします。）

0x02 は、0x7D, 0xE2　２バイトにエスケープ	(STX escape)
0x03 は、0x7D, 0xE3　２バイトにエスケープ	(ETX escape)
0x7D は、0x7D, 0x5D　２バイトにエスケープ	(Control escape)

CRC について

CRC 算出には CRC-16-CCITT を採用していますが、CRC-16-CCITT には複数の実装が知られておりエミュレータには最も一般的な実装を採用しています。これは“123456789”という文字列を計算した際に 0x29B1 を返す物ですが、他の実装では別の値を返す物が存在するので気を付けて下さい。巻末に C 言語と VB による実装例を添付します。

参考サイト： http://www.zorc.breitbandkatze.de/crc.html

５．　コマンド

コマンドは以下のフォーマットによる連続したバイト列です。

< command > < data >

< command > は１バイトの制御コードです。

0x20: Read Register (Number of Coins Accepted)
0x21: Read Register (Coin Mech Status)
0x22: Read Register (Number of Bills Accepted)
0x23: Read Register (Bill Vali Status)

0x30: Write Register (Number of Coins Accepted)
0x31: Write Register (Coin Mech Status)
0x32: Write Register (Number of Bills Accepted)
0x33: Write Register (Bill Vali Status)

0x40: Get Firmware Version string
0x41: Get Mode Byte

< data > コマンドに依存する複数バイトで 0x30 – 0x33 のコマンドに必要です。

６．　レスポンス

レスポンスは以下に定義される連続したバイト列です。

< ackcode > < result >

< ackcode > はコマンドに対する結果を表す１バイトコードです。

0x00: Success
0xFF: Other Error
0xFE: Frame Error
0xFD: Command Error

< result > は< ackcode >が success の場合のみ現れ得るデータバイトです。

７．　参考資料

CRC-16-CCITT（Ｃ言語によるサンプル）

```
#ifdef ROM_TABLE
/*****************************************************************************
*  ROM Table
*****************************************************************************/
      const unsigned short CrcTable[256] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
```

```
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0 };
#else
    unsigned short CrcTable[256];
#endif

/*****************************************************************************
* Function: void crc_create_table(void);
* Overview: Create CRC-16 CCITT Table
* Input: none
* Output: none
*****************************************************************************/
void crc_create_table(void)
{
#ifndef ROM_TABLE
#define polynomial 0x1021

    unsigned int i, j;
          unsigned short a;

    for (i = 0; i < 256; i++)
    {
        a = (i << 8);
        for (j = 0; j < 8; j++)
        {
            if (0 == (a & 0x8000)) a = (a << 1);
            else a = polynomial ^ (a << 1);
        }
        CrcTable[i] = a;
    }
#endif
}
```

```
/*****************************************************************************
* Function: unsigned short crc_1byte(unsigned char in, unsigned short crc);
* Overview: Calc CRC (1 byte)
* Input: in - data
*       crc - Initial Value
* Output: crc value
*****************************************************************************/
      unsigned short crc_1byte(unsigned char in, unsigned short crc)
{
    return CrcTable[((crc >> 8) ^ in) & 0xFF] ^ (crc << 8);
}

/*****************************************************************************
* Function: unsigned short crc_block(unsigned char buff[], unsigned short
size, unsigned crc);
* Overview: Calc CRC (block)
* Input: in - data
*       crc - Initial Value （初期値は通常、0xFFFF を設定して呼び出します。）
* Output: crc value
*****************************************************************************/
      unsigned short crc_block(unsigned char buff[], unsigned short size,
      unsigned short crc)
{
          unsigned short pos;

    for(pos=0; pos!=size; pos++)
    {
        crc = crc_1byte(buff[pos], crc);
    }

    return crc;
}
```

CRC-16-CCITT（VisualBasic によるサンプル）

```
Private CRC_CCITT_TABLE(255) As Integer

Private Sub CreateCRCTable()
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim Value As Integer
    For i = 1 To 256
        j = i * &HFF
        Value = &H0
        For k = 0 To 7
            If ((Value Xor j) And &H8000) = &H8000 Then
                Value = ((Value * 2) Xor &H1021) And &HFFFF
            Else
                Value = (Value * 2) And &HFFFF
            End If
            j = j * 2
            CRC_CCITT_TABLE(i - 1) = Value
        Next
    Next
End Sub

  Private Function CRC16_Calculate(ByRef bytes() As Byte, nmbr As Integer) As
Integer
    Dim CRC_Value As Integer = &HFFFF
    Dim i As Integer
    For i = 0 To nmbr - 1
        Dim Value As Integer = (CRC_Value >> 8) Xor bytes(i)
                    CRC_Value = ((CRC_Value << 8) And &HFFFF) Xor
CRC_CCITT_TABLE(Value)
    Next
    Return CRC_Value
End Function
```