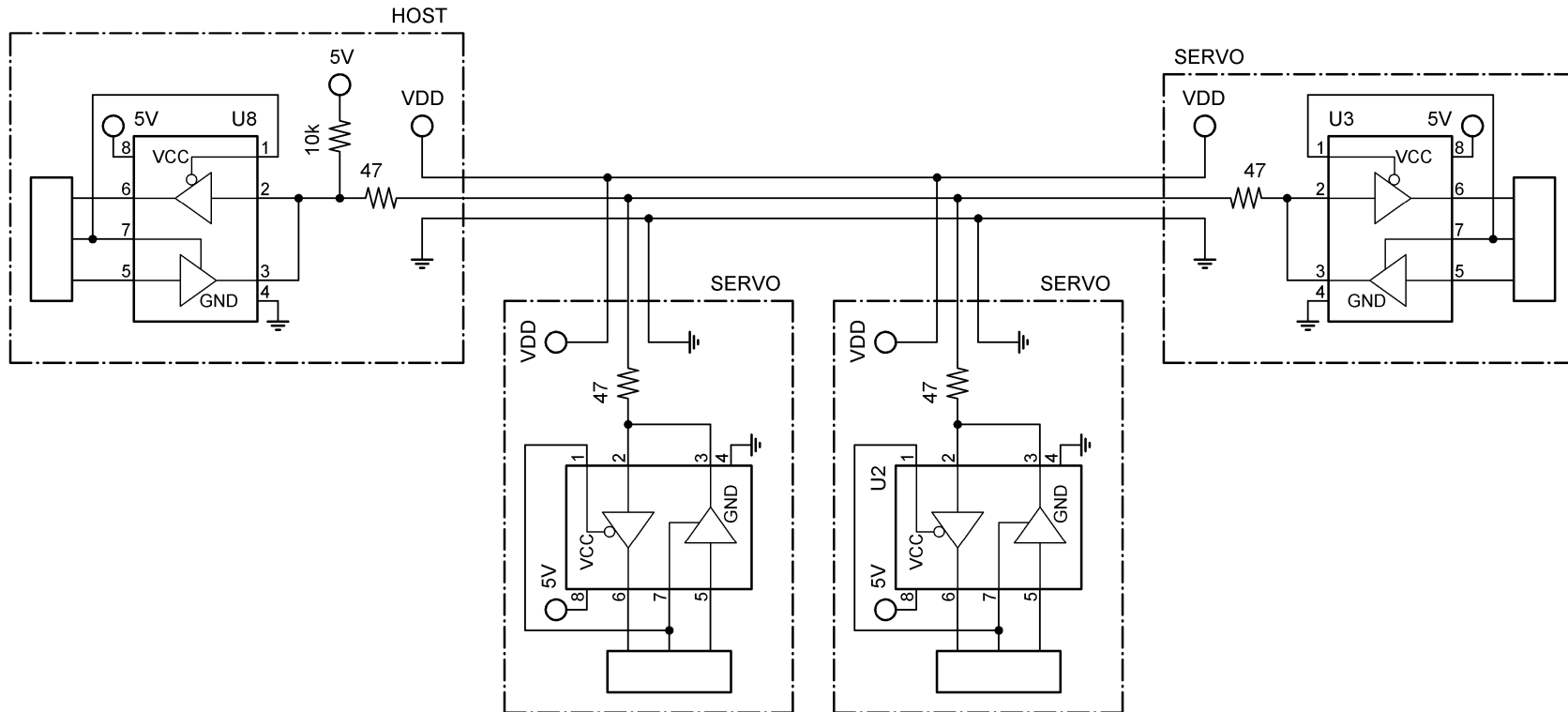
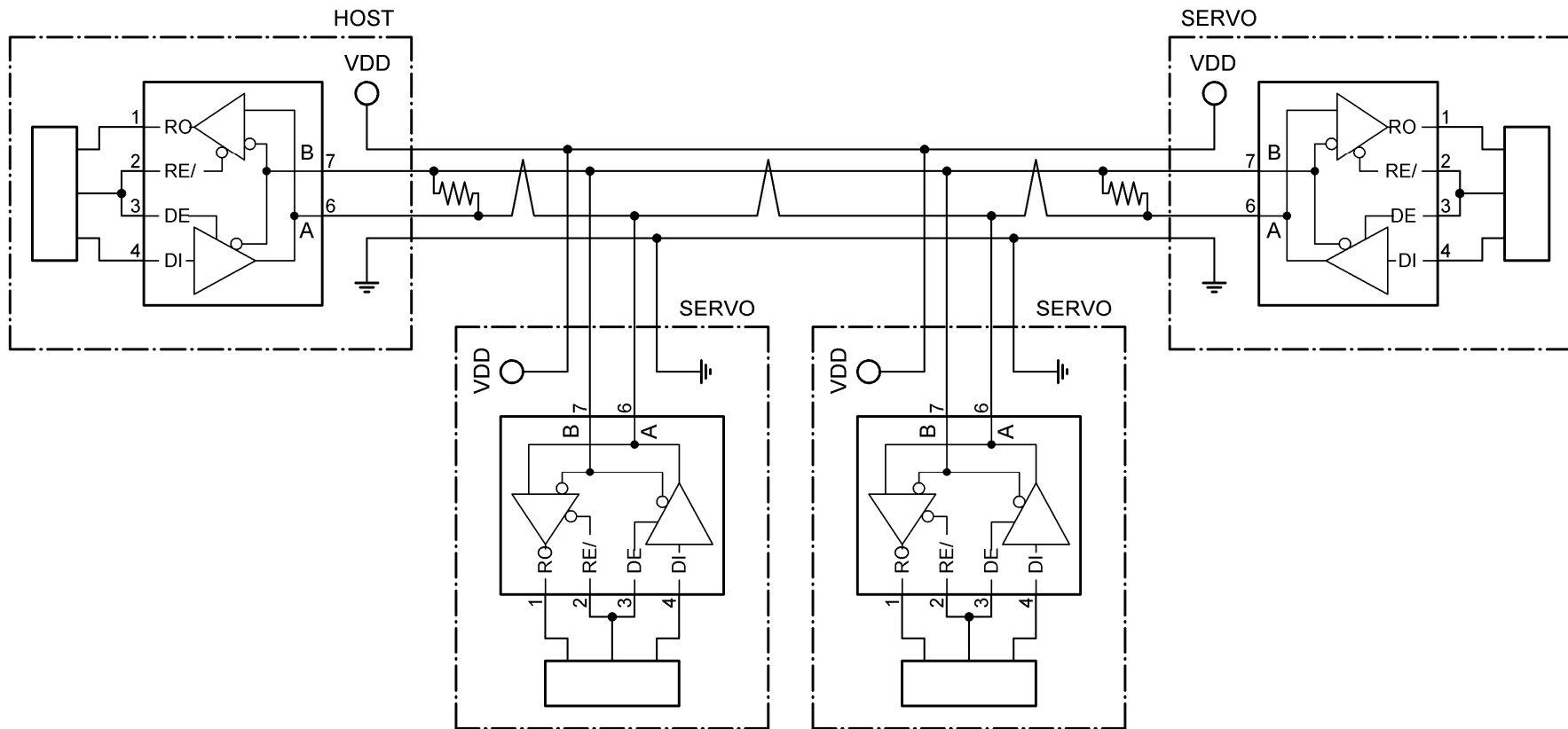


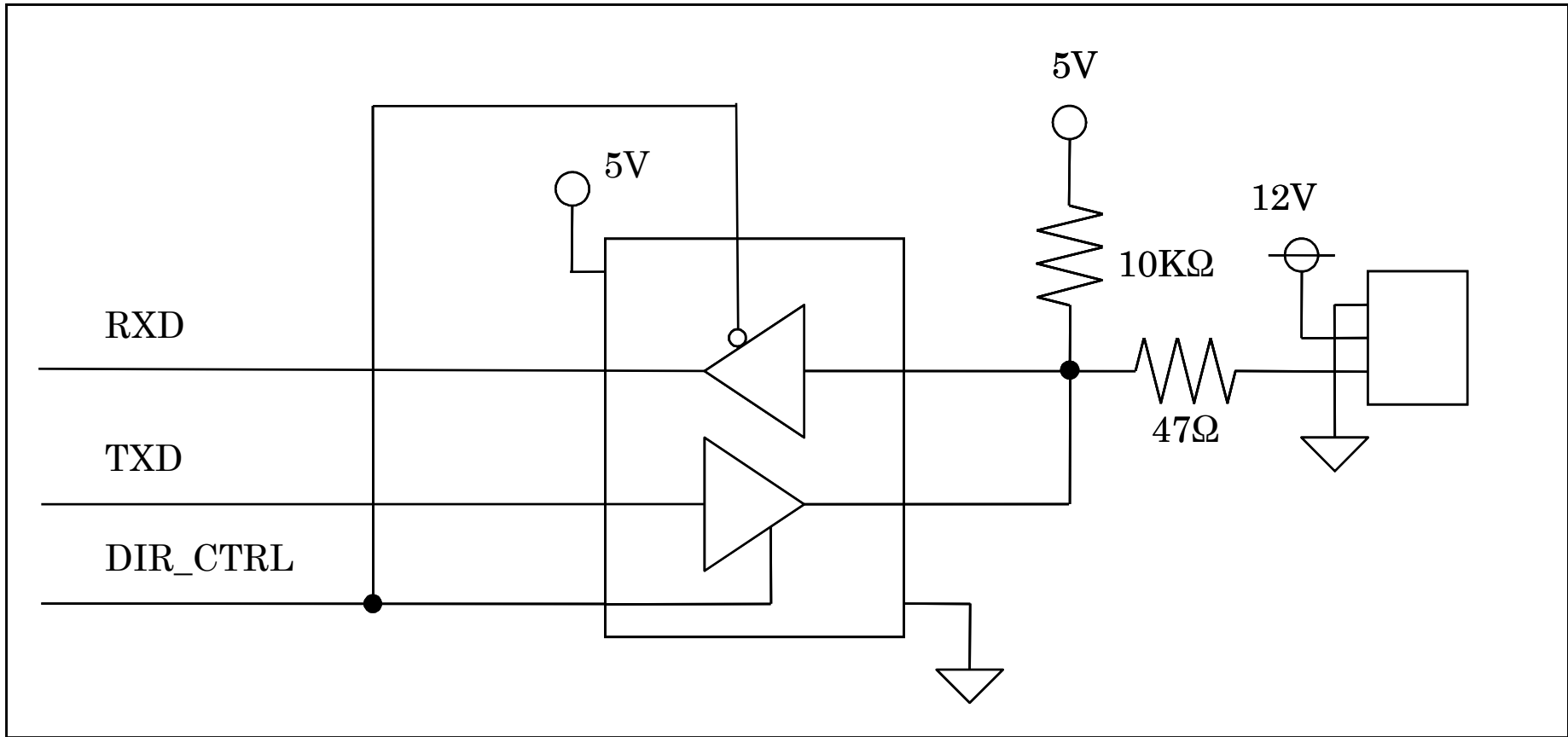
ホスト～アクチュエータ間接続イメージ(TTL)



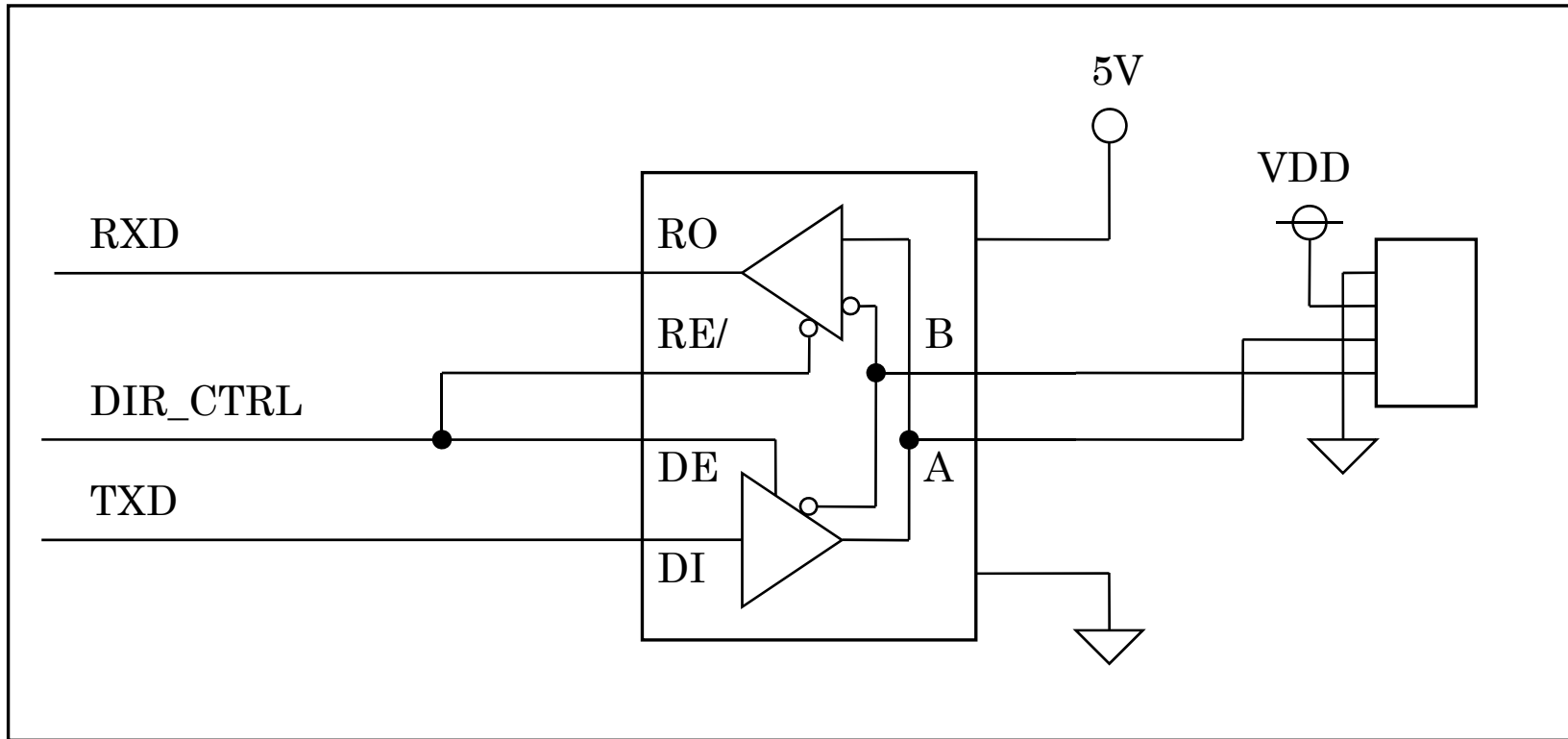
ホスト～アクチュエータ間接続イメージ(RS485)



双方向TTL



半二重RS485



SAM7S +DXROBO_SAM7Sライブラリ

アームロボで 簡単なデモ

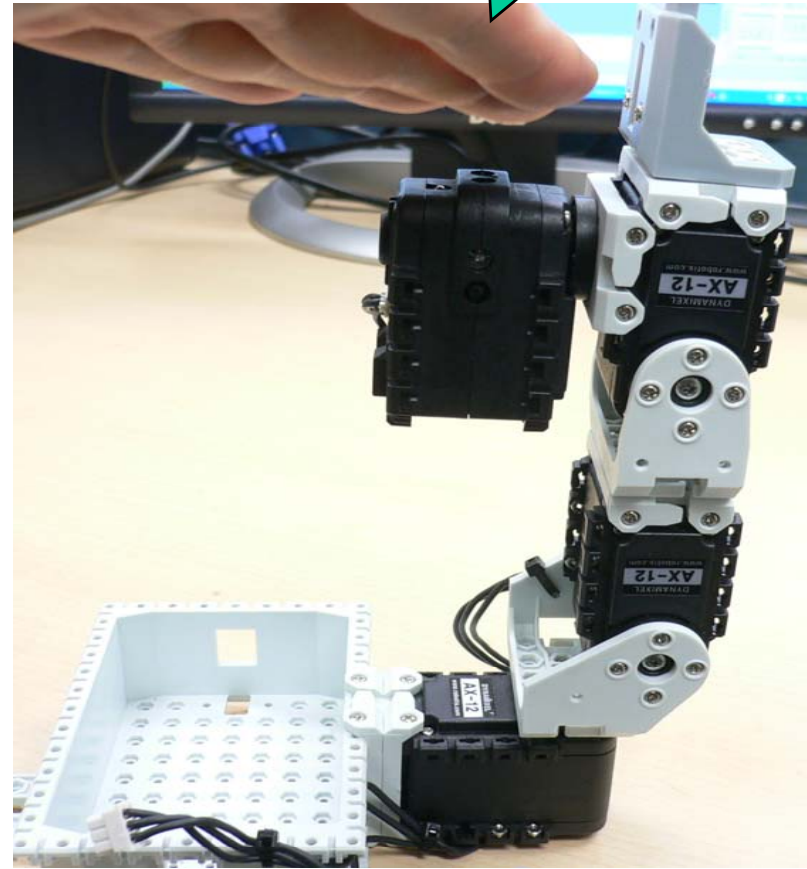
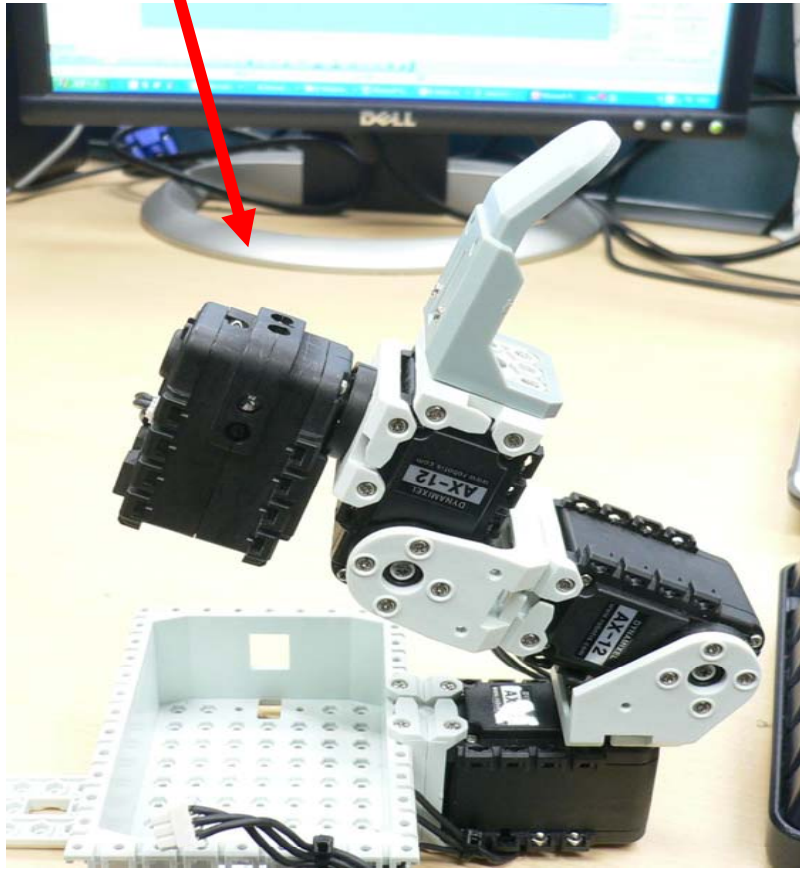
```
· #include <stdio.h>
· #include <us.h>
· #include <dx.h>
· #include <vic.h>

· void main(void)
· {
·     TSyncPosData pout[3];
·     unsigned char result;
·     DX_Init(1000000, 8, dbg_u_putc, NULL); // DX Library初期化
·     enableIRQ(); // 割り込み許可
·     while(1) {
·         DX_ReadByteData (100, 27, &result); // AX-S1の中央反射センサの情報を読む
·         if(result>64){
·             pout[0].ID=1; pout[0].GoalPosition = 512; //ID1に目標位置設定
·             pout[1].ID=2; pout[1].GoalPosition = 512; //ID2に目標位置設定
·             pout[2].ID=3; pout[2].GoalPosition = 512; //ID3に目標位置設定
·             DX_SendSyncPosition(pout, 3); // 目標位置の一括送信
·         }
·         else{
·             pout[0].ID=1; pout[0].GoalPosition = 512; //ID1に目標位置設定
·             pout[1].ID=2; pout[1].GoalPosition = 512-200; //ID2に目標位置設定
·             pout[2].ID=3; pout[2].GoalPosition = 512+300; //ID3に目標位置設定
·             DX_SendSyncPosition(pout, 3); // 目標位置の一括送信
·         }
·         DX_wait(100); // 100ms待ち
·     }
· }
```

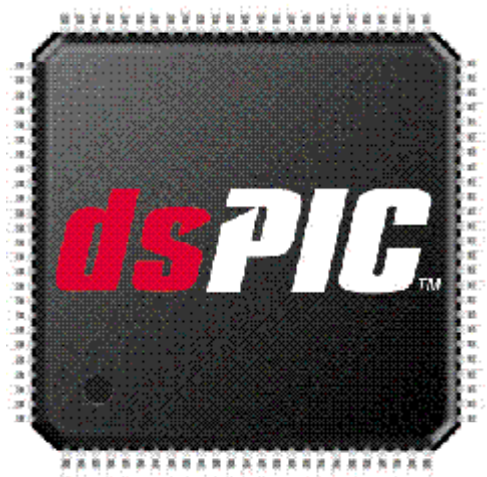
動作の様子

アタック!

反射センサ



dsPICでDX-117を動かす



dsPICとは？

- Microchip Technology社のマイコン
- いろいろな機能があり使いやすい
- 開発ツールが無料

dsPIC30F4011 でDX-117を動かしてみよう！



- 適当にボードを作る
- Cコンパイラは無料で手に入る
- シリアル通信関数は用意されている

dsPICでDX-117を動かす

```
#include <p30f4011.h>
#include "timer.h"
#include "uart.h"
```

～ 中略 ～

```
int main(void)
```

```
{
    tx_char[0] = 0xFF;
    tx_char[1] = 0xFF;
    tx_char[2] = 0x01;
    tx_char[3] = 0x05;
    tx_char[4] = 0x03;
    tx_char[5] = 0x1E;
    tx_char[6] = 0x00;
    tx_char[7] = 0x02;
    tx_char[8] = 0xD6;
```

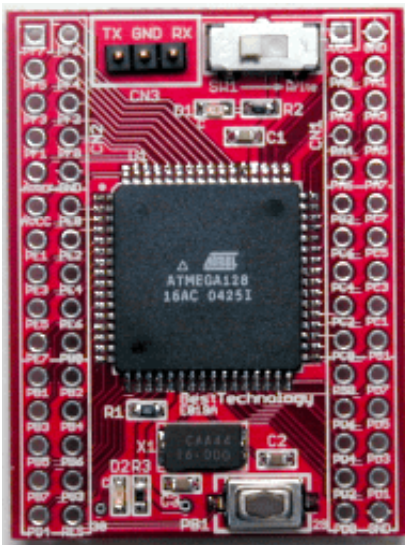
～ 中略 ～

```
OpenUART1(UMODEValue, USTAValue, 12);
OpenUART2(UMODEValue2, USTAValue2, 25);
```

```
while(1){
    putsbUART2(tx_char, 9);
    while(!DataRdyUART2());
    while(DataRdyUART2() && getcUART2() != 0xFF){i
= 0;}
    while(DataRdyUART2()){
        rx_char[i] = getcUART2();
        i++;
    }

    printf("%2x %2x %2x %2x¥n¥r",rx_char[0],rx_char[1]
,rx_char[2],rx_char[3]);
    while(BusyUART1());
    delay(500);
}
}
```

ATmega128でDX-117を動かす



ATmega128ボードとは？

- ・Atmel社のマイコンATmega128
- ・小型で外付けRAM付
- ・開発環境を無料で提供

ATmega128ボードでDX-117を動かしてみよう！

- ・AX-12/DX-117/RX-64用の関数が用意されている
- ・数命令を追加すれば動かせる

ATmega128でDX-117を動かす (DXROBO_AM128ライブラリ)

```
#include <avr/interrupt.h>
```

```
#include <rs.h>
```

```
#include "dx.h"
```

～ 中略 ～

```
void Position(_BYTE id, _WORD p)
```

```
{
```

```
    int j, l2;
```

```
    _BYTE param[10];
```

```
    param[0] = ADDRESS_GOAL_POSITION;
```

```
    param[1] = p & 0xff;
```

```
    param[2] = (p >> 8) & 0xff;
```

```
    j = DX_TxPacket(tbuf, id, INST_WRITE, param, 3);
```

```
    DX_PrintB(" <-", tbuf, j);
```

```
    j = DX_RxPacket(rbuf, &l2, _DX_READ_RESPONSE_TIME);
```

```
    if(id != BROADCASTING_ID) DX_PrintB(" ->", rbuf, j);
```

```
}
```

```
void main(void)
```

```
{
```

```
    rs0_init (br19200, txb, sizeof(txb), rxb,  
            sizeof(rxb));
```

```
    DX_Init(br9600);
```

```
    sei ();
```

```
    while (1) {
```




```
        Position(1, 512);
```

```
    }
```

```
}
```

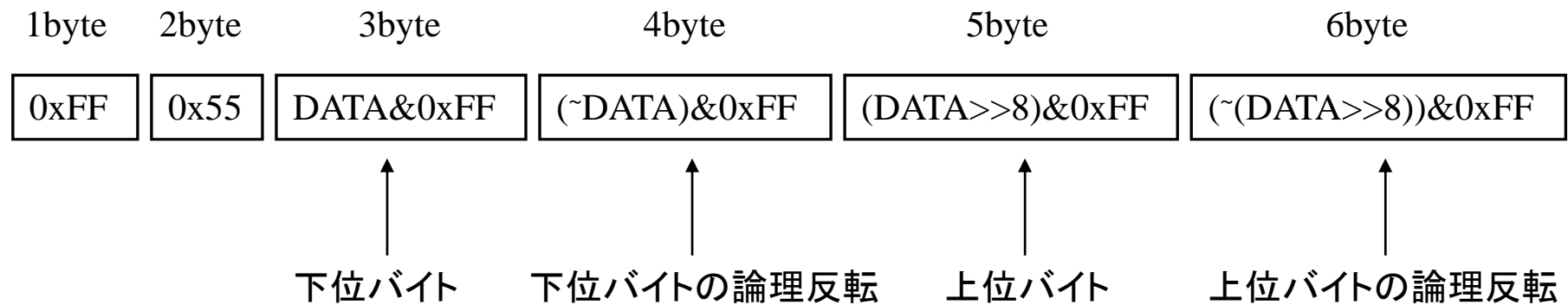
Bioloid同士がZigbeeで通信

送信側 コントローラ

1	[Label]	START							[Comment]		
2	[Label] Loop	IF	( Button	=	 8)	THEN	LOAD	 TX rem... (-	1
3	[Label]	ELSE IF	( Button	=	 4)	THEN	LOAD	 TX rem... (-	2
4	[Label]	ELSE IF	( Button	=	 2)	THEN	LOAD	 TX rem... (-	3
5	[Label]	ELSE IF	( Button	=	 1)	THEN	LOAD	 TX rem... (-	4
6	[Label]	ELSE IF	( Button	=	 16)	THEN	LOAD	 TX rem... (-	5
7	[Label]	JUMP	Loop						[Comment]		
8	[Label]	END							[Comment]		

BiolooidがZigbeeで通信するときのパケット

送信するデータを **DATA**(16bit: 0~65535) とすると



Bioloidを AT91SAM7S32とZigbeeで操作する

```
#include <AT91SAM7S.h>
#include <us.h>

#define SW1 (1<<15) // PA15
#define SW2 (1<<14) // PA14
#define SW3 (1<<19) // PA19
#define SW4 (1<<20) // PA20
#define SW5 (1<<18) // PA18

unsigned char txb[10];

void CM5_TxData(unsigned short data){
    int i;

    txb[0] = 0xff;
    txb[1] = 0x55;
    txb[2] = data & 0xff;
    txb[3] = (~data) & 0xff;
    txb[4] = (data >> 8) & 0xff;
    txb[5] = (~(data >> 8)) & 0xff;

    for(i = 0; i < 6; i++) dbg_u_putc(txb[i]);
}
```

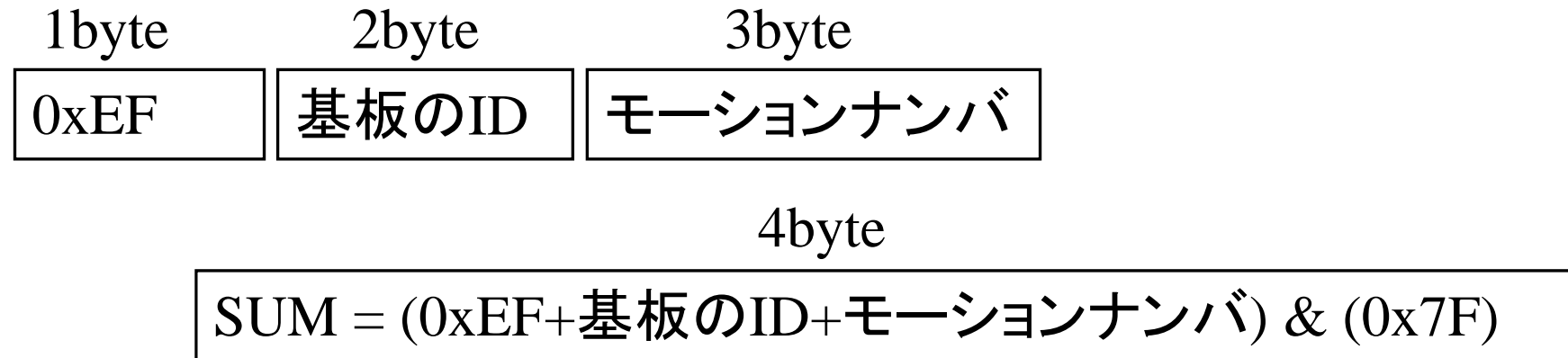
```
void main(void){
    dbg_u_init (115200);
    PMC.PCER = 1 << PIOA_PID;

    while(1){
        if (!(PIOA.PDSR & SW1)) CM5_TxData(1);
        if (!(PIOA.PDSR & SW2)) CM5_TxData(2);
        if (!(PIOA.PDSR & SW3)) CM5_TxData(3);
        if (!(PIOA.PDSR & SW4)) CM5_TxData(4);
        if (!(PIOA.PDSR & SW5)) CM5_TxData(5);
    }
}
```

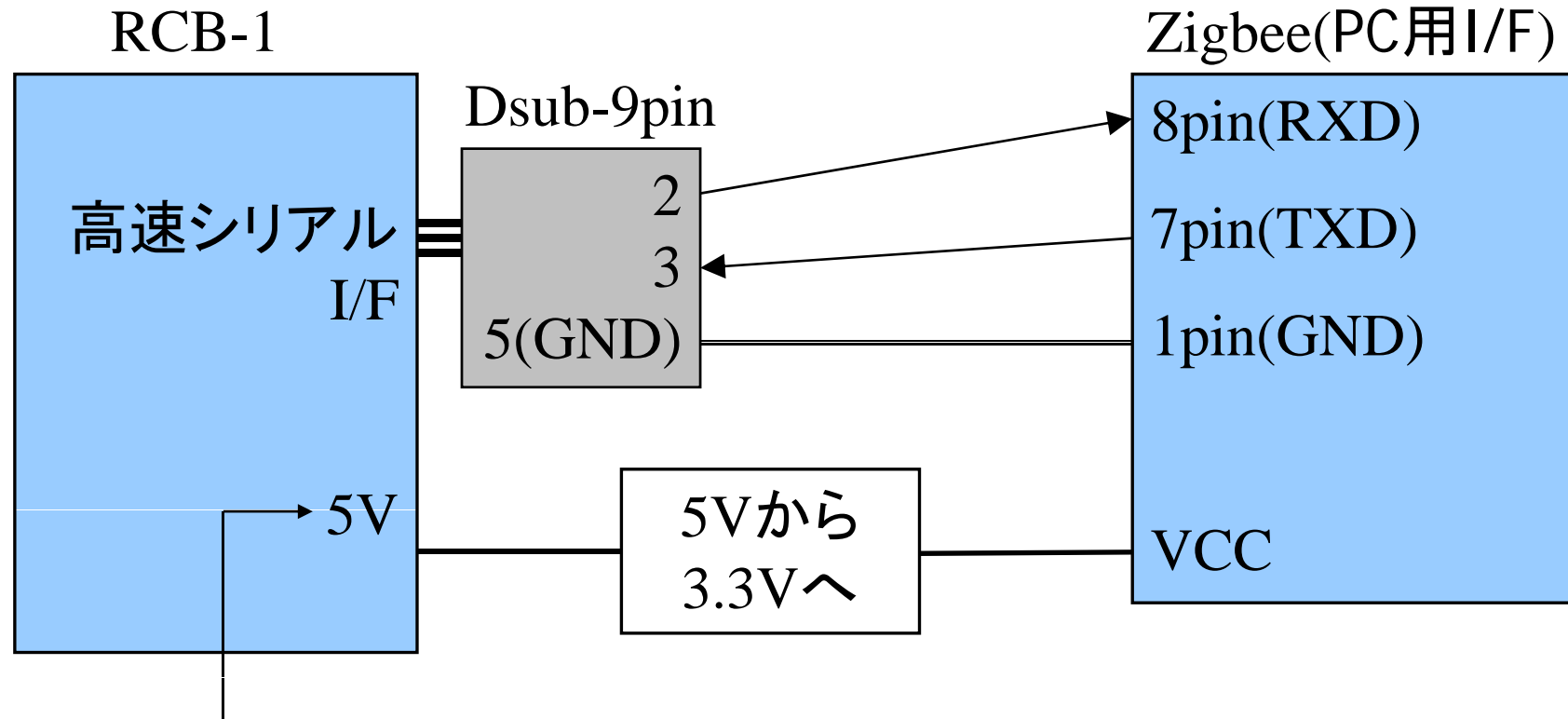
KHR-1のモーションを再生するコマンド

高速シリアル

通信速度115200bps



RCB-1をAT91SAM7S32とZigbeeで操作する



低速シリアルからとった

RCB-1をAT91SAM7S32とZigbeeで操作する

```
#include <AT91SAM7S.h>
#include <us.h>

#define SW1 (1<<15) // PA15
#define SW2 (1<<14) // PA14
#define SW3 (1<<19) // PA19
#define SW4 (1<<20) // PA20
#define SW5 (1<<18) // PA18

unsigned char txb[10];

void RCB1_TxData (unsigned char ID, unsigned
char MotionNO) {
    int i;

    txb[0] = 0xef;
    txb[1] = ID;
    txb[2] = MotionNO;
    txb[3] = (0xef + ID + MotionNO) & ~(1 << 7);
    for (i = 0; i < 4; i++) dbgu_putc (txb[i]);
}
```

```
void main (void){
    dbgu_init (115200);
    PMC.PCER = 1 << PIOA_PID;

    while (1) {
        if (!(PIOA.PDSR & SW1)) RCB1_TxData (0, 0);
        if (!(PIOA.PDSR & SW2)) RCB1_TxData (0, 1);
        if (!(PIOA.PDSR & SW3)) RCB1_TxData (0, 3);
        if (!(PIOA.PDSR & SW4)) RCB1_TxData (0, 2);
    }
}
```

SH7145FによるDX117の運転

RS485チップとSH7145Fのピンアサイン

- PE7/RXD2 MAX485 RO
- PE10/TXD2 MAX485 DI
- PE14 MAX485 RE (L=Enable)
- PE15 MAX485 DE (H=Enable)

SH7145FによるDX117の運転

SH7145F(I/Oクロック24.576MHz)時のボーレート例

• SH	DX	誤差率
• N=13 54857[bps]	55556[bps]	1.258%
• N=12 59077[bps]	58824[bps]	-0.43%
• N=11 64000[bps]	64516[bps]	0.8%
• N=10 69818[bps]	68966[bps]	-1.236%
• N=9 76800[bps]	76923[bps]	0.16%
• N=8 85333[bps]	86957[bps]	1.867%
• N=7 96000[bps]	95238[bps]	-0.8%
• N=6 109714[bps]	111111[bps]	1.257%
• N=4 153600[bps]	153846[bps]	0.16%

SH7145FによるDX117の運転

設計思想

- SCIの送受信は割り込みを使用
- ディレクションコントロールも割り込みで
- Dynamixelの基本パケット構成はサブルーチン化し、IDやパラメータのみを引数として処理
- SHからパケットの送信後、Dynamixelの応答パケットを必ず受信
- 何かあった場合はタイムアウトにて抜けエラ
- サンプルという事で通信ができる事を最優先にし、できるだけ簡素化
- パケットの内容はターミナルに表示可能

SH7145FによるDX117の運転

DX用基本サブルーチン

- 受信バッファクリア
void DX_RXPurge (void)
- Dynamixel用パケット送信
int DX_TxPacket (_BYTE *tbuf, _BYTE bID, _BYTE inst, _BYTE *pparam, _BYTE dlen)
- Dynamixel用パケット受信
int DX_RxPacket (_BYTE *rbuf, int *len, int timeout)
- パケット内容の確認用(デバッグ用)
void DX_PrintBuffer (char *cmt, unsigned char *pbuf, int len)

後は実際のソースを参考に

ちなみにどうしても外付けクロックなしで高速な通信を行いたいという場合は、CPUのクロックを32MHz(オーバクロック)にする事で500kbpsまで対応可能