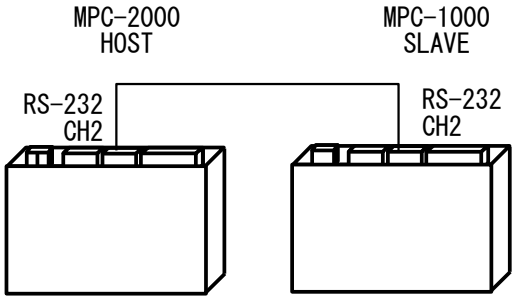
Application Note		Ref No: an2k-046	Last Modify 130809
テーマ	シリアル通信による MPC 間のメモリーシェア		
使用機器	MPC-1000, MPC-2000,MPC-2200 他		

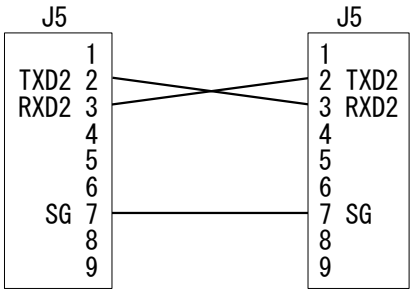
RS-232 メモリ共有

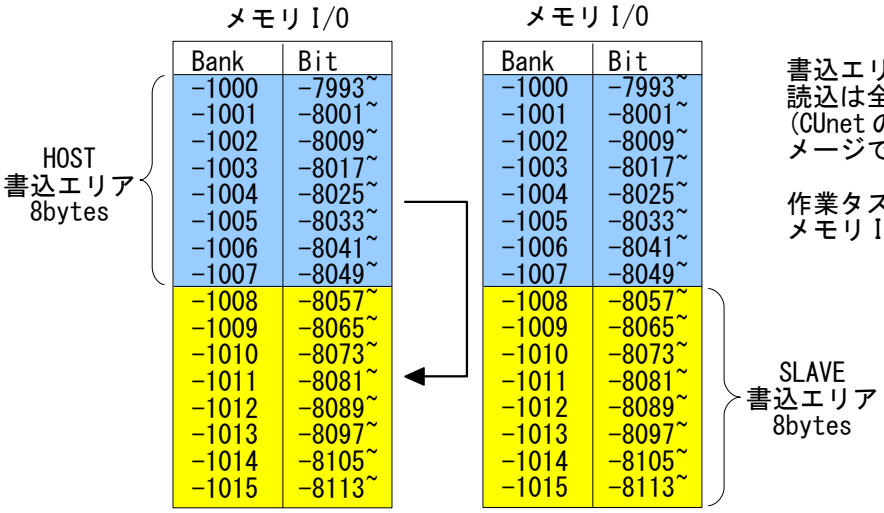
機器構成



MPC の RS-232 ポート同士を接続。

タスクを 1 つ使い、バックグラウンドで常時通信してメモリ I/O を共有します。通信の主導権は HOST が持ちます。





書込エリアは限定されますが、読込は全てのエリアが可能です。(CUnet のメモリー共有のようなイメージです)

作業タスクでは、I/O コマンドでメモリ I/O を操作します。

- 本サンプルは MPC-2000 と MPC-1000 を使っていますが、他の機種、RS-232 チャンネルでも可能です。RS-232 チャンネル、メモリ I/O エリア、通信タスクなどは適宜変更してください。
- サンプルのリフレッシュレートは約 0.1 秒です。
- HOST は SLAVE が停止しても自動復帰します。
- HOST、SLAVE ともタイムアウトチェック、サムチェックを行っており、それらの回数は点データに格納しています。
- 過去にタッチパネル通信(MEWNET)をしていた MPC を使用する場合は初期化(MPCINIT)してからプログラムを入れてください。
- サンプルソースのご請求はメールでサポートまでお願いします。

HOST プログラム

【使用方法】
*mem_sharing_host 以下を任意のタスクで実行します(バックグランドで通信するイメージです)。
作業タスクでは通常の I/O コマンドでメモリ I/O を扱います。

```
  QUIT_FORK 20 *mem_sharing_host    /* HOST 通信タスク
  TIME 500

  QUIT_FORK 1 *LOOP_BACK_TEST       /* 作業タスク
  END

*LOOP_BACK_TEST

  OFF -8025
  WAIT SW(-8089)==0
  I=0
  DO
    START_CLK=SYSCLK
    OUT I -1000~Lng                 /* 自己エリアにデータセット
    ON -8025                        /* データセット完了→Slave
    WAIT SW(-8089)==1               /* Slave の作業完了待ち
    A=IN(-1008~Lng)                 /* Slave エリア読込
    OFF -8025
    WAIT SW(-8089)==0
    ELAPSED=SYSCLK-START_CLK        /* 所要時間 このサンプルでは約 0.1 秒
    OUT A 0~Lng                     /* 実出力に出力
    PR I ELAPSED
    IF I<>A THEN
      PR "compare error"
      END
    END_IF
    I=I+1
  LOOP

*mem_sharing_host

  share_comport=2                                   /* COM ポート番号
  CNFG# share_comport "38400b8pns1NONE"             /* 通信ポート初期化
  share_base_adr=-1000                              /* シェアするメモリ I/O 先頭 Bank
  share_err_cnt=7000                               /* エラーカウントを格納する点データ番号
  SETP share_err_cnt 0 0 0 0                        /* エラーカウント初期化

  DO
    share_dt1=IN(share_base_adr~Lng)                /* 自己エリア入力
    share_dt2=IN((share_base_adr-4)~Lng)            /* 自己エリア入力
    share_sum=share_dt1^share_dt2                   /* チェックサム
    share_send$=HEX$(share_dt1)+","+HEX$(share_dt2)+","+HEX$(share_sum)+"¥r" /* 送信文字列
    INPUT# share_comport CLR_BUF                    /* 入力バッファクリア
    PRINT# share_comport share_send$                /* 送信

    INPUT# share_comport TMOUT|2 share_dt$          /* Slave からの応答
    IF (rse_==0)&(LEN(share_dt$)==26) THEN          /* タイムアウトじゃない & 正常文字数 なら
      share_dt1=HEX(share_dt$)                      /* Slave の IN((share_base_adr-8)~Lng)
      ptr_=ptr_+1
      share_dt2=HEX(8)                              /* Slave の IN((share_base_adr-12)~Lng)
      ptr_=ptr_+1
      share_sum=HEX(8)                              /* チェックサムパート
      IF share_dt1^share_dt2^share_sum==0 THEN      /* チェックサムをチェック
        OUT share_dt1 (share_base_adr-8)~Lng        /* Slave エリアに格納
        OUT share_dt2 (share_base_adr-12)~Lng       /* Slave エリアに格納
      ELSE
        X(share_err_cnt)=X(share_err_cnt)+1         /* チェックサムエラーカウンタ
      END_IF
      /* Y(share_err_cnt)=0                         /* タイムアウトカウンタクリア
```

```
    ELSE
      IF rse_<>0 THEN
        /*PR "TIMEOUT"
        Y(share_err_cnt)=Y(share_err_cnt)+1      /* タイムアウトカウンタ
      END_IF
    END_IF
    TIME 1
  LOOP
```

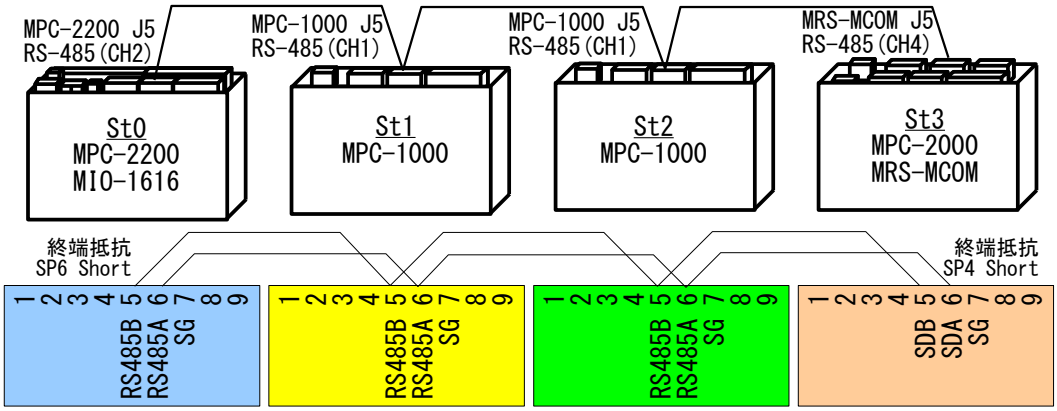
SLAVE プログラム

【使用方法】
*mem_sharing_slave 以下を任意のタスクで実行します(バックグランドで通信するイメージです)。
作業タスクでは通常の I/O コマンドでメモリ I/O を扱います。

```
  QUIT_FORK 20 *mem_sharing_slave  /* SLAVE 通信タスク
  TIME 500

  QUIT_FORK 1 *LOOP_BACK_TEST      /* 作業タスク
  END

*LOOP_BACK_TEST
  OFF -8089
  DO
    WAIT SW(-8025)==1            /* Host のデータセット完了待ち
    A=IN(-1000~Lng)              /* データ入力
    OUT A -1008~Lng              /* 自己エリアに書き込む
    OUT A 0~Lng                  /* 実出力に出力
    PR A
    ON -8089                     /* 作業完了->Host
    WAIT SW(-8025)==0
    OFF -8089
  LOOP

*mem_sharing_slave

  share_comport=2                                /* COM ポート番号
  CNFG# share_comport "38400b8pns1NONE"          /* 通信ポート初期化
  share_base_adr=-1000                           /* シェアするメモリ I/O 先頭 Bank
  share_err_cnt=7000                             /* エラーカウントを格納する点データ番号
  SETP share_err_cnt 0 0 0 0                     /* エラーカウント初期化

  DO
    INPUT# share_comport share_dt$               /* Host からのデータ待ち
    IF LEN(share_dt$)==26 THEN
      share_dt1=HEX(share_dt$)                   /* Host の IN(share_base_adr~Lng)
      ptr_=ptr_+1
      share_dt2=HEX(8)                           /* Host の IN((share_base_adr-4)~Lng)
      ptr_=ptr_+1
      share_sum=HEX(8)                           /* チェックサム
      IF share_dt1^share_dt2^share_sum==0 THEN   /* チェックサムをチェック
        OUT share_dt1 share_base_adr~Lng         /* Host エリアに格納
        OUT share_dt2 (share_base_adr-4)~Lng     /* Host エリアに格納
      ELSE
        X(share_err_cnt)=X(share_err_cnt)+1      /* チェックサムエラーカウンタ
      END_IF
    END_IF
    share_dt1=IN((share_base_adr-8)~Lng)       /* 自己エリア入力
    share_dt2=IN((share_base_adr-12)~Lng)      /* 自己エリア入力
    share_sum=share_dt1^share_dt2              /* チェックサム
    share_send$=HEX$(share_dt1)+","+HEX$(share_dt2)+","+HEX$(share_sum)+"¥r" /* 送信文字列
    INPUT# share_comport CLR_BUF               /* 入力バッファクリア
    PRINT# share_comport share_send$           /* 送信
    TIME 1
  LOOP
```

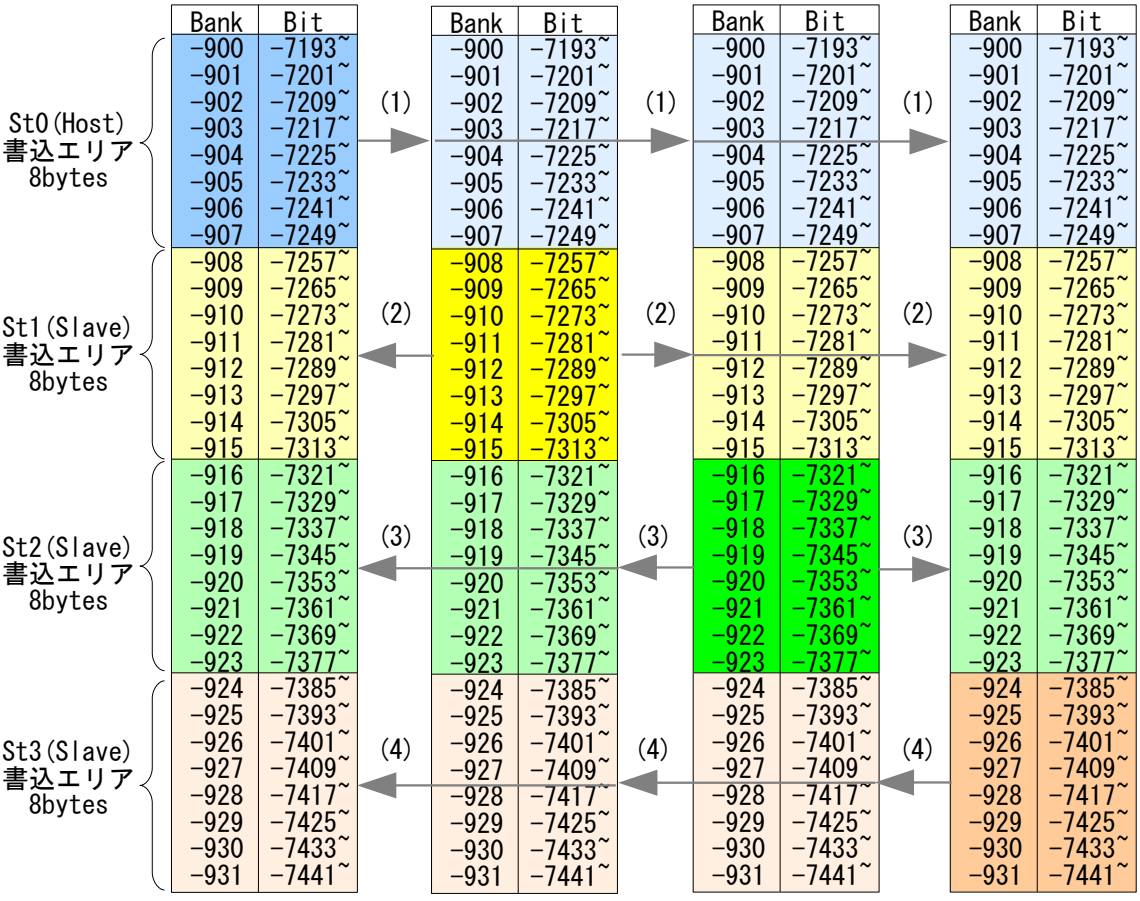
RS-485 メモリ共有

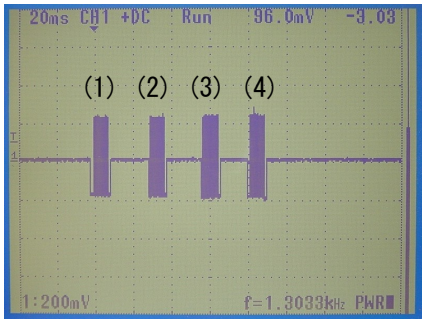
機器構成



※別電源使用などのとき、必要に応じて SG も接続

書込エリアは St 毎に限定されます。読込は全エリア可能です。（CUnet のようなイメージです）





1 セットの通信波形

Host の発信 (1) から一定の間隔で
Slave の発信 (2) (3) (4) が行われます。

他のタスクの実行状況により、
数 msec ぶれることがあります。

- RS-485 ブロードキャストでメモリー I/O を共有します。
- サンプルのリフレッシュレートは約 0.2 秒です。
- 各 St はタイムアウトチェック、サムチェックを行っており、それらの回数は点データに格納しています。
- 過去にタッチパネル通信(MEWNET)をしていた MPC を使用する場合は初期化(MPCINIT)してからプログラムを入れてください。
- サンプルソースのご請求はメールでサポートまでお願いします。

プログラム (各 St 共通)

【使用方法】
*mem_sharing 以下を任意のタスクで実行します(バックグランドで通信するイメージです)。
作業タスクでは通常の I/O コマンドでメモリ I/O を扱います。
通信タイミング生成に SYSCLK を使います。他のタスクでリセットや変更をしないで下さい。

```
  QUIT_FORK 20 *mem_sharing                    /* 通信タスク (各 St 共通)
  TIME 1000

  QUIT_FORK 1 *LOOP_BACK_TEST                  /* 作業タスク
  FOR i=2 TO 19
    QUIT_FORK i *DUMMY                         /* マルチタスク確認用のダミータスク
  NEXT

  END

*LOOP_BACK_TEST                                /* 動作確認用作業タスク
/* St0 が出力したデータを St1,St2,St3 は自己エリアにコピーする。
/* St0 は自分が出力したデータと St1,St2,St3 が出力したデータを比較する。

  outdt_=0
  DO
    IF share_myadrs==0 THEN
      start_clk=SYSCLK
      OUT outdt_ -900~Lng        /* データセット
      ON -7249                   /* データセット完了 -> St1,St2,St3

      WAIT SW(-7313)==1          /* St1 のデータセット完了待ち
      indt1_=IN(-908~Lng)        /* St1 のデータ読込

      WAIT SW(-7377)==1          /* St2 のデータセット完了待ち
      indt2_=IN(-916~Lng)        /* St2 のデータ読込

      WAIT SW(-7441)==1          /* St3 のデータセット完了待ち
      indt3_=IN(-924~Lng)        /* St3 のデータ読込

      OFF -7249
      WAIT SW(-7313)==0
      WAIT SW(-7377)==0
      WAIT SW(-7441)==0

      elapsed=SYSCLK-start_clk
      /* St0data,St1data,St2data,St3data,error stat, 経過時間
      PR outdt_ indt1_ indt2_ indt3_ P(7000) elapsed
      IF (outdt_<>indt1_)|(outdt_<>indt2_)|(outdt_<>indt3_) THEN
        PRINT "Compare error"
        END
      END_IF

      outdt_=outdt_+1
    END_IF

    IF share_myadrs==1 THEN
      WAIT SW(-7249)==1          /* St0 のデータセット完了待ち
      indt_=IN(-900~Lng)         /* St0 のデータ読込
```

```
      PR indt_ P(7000)          /* St0data, error stat
      OUT indt_ -908~Lng        /* 読み込んだデータを自己エリアにセット
      ON -7313                  /* St1 データセット完了 -> St0
      WAIT SW(-7249)==0
      OFF -7313
    END_IF

    IF share_myadrs==2 THEN
      WAIT SW(-7249)==1         /* St0 のデータセット完了待ち
      indt_=IN(-900~Lng)        /* St0 のデータ読込
      PR indt_ P(7000)          /* St0data, error stat
      OUT indt_ -916~Lng        /* 読み込んだデータを自己エリアにセット
      ON -7377                  /* St2 データセット完了 -> St0
      WAIT SW(-7249)==0
      OFF -7377
    END_IF

    IF share_myadrs==3 THEN
      WAIT SW(-7249)==1         /* St0 のデータセット完了待ち
      indt_=IN(-900~Lng)        /* St0 のデータ読込
      PR indt_ P(7000)          /* St0data, error stat
      OUT indt_ -924~Lng        /* 読み込んだデータを自己エリアにセット
      ON -7441                  /* St3 データセット完了 -> St0
      WAIT SW(-7249)==0
      OFF -7441
    END_IF

    SWAP

  LOOP

*DUMMY                          /* Debug のためのダミータスク
  DO
    AA=AA+1
    AA=AA+1
    AA=AA+1
    AA=AA+1
    AA=AA+1
    AA=0
    SWAP
  LOOP

/*
/* メモリ I/O 共有タスク
/*
*mem_sharing

  /* このサンプルは入力で St 番号,RS-485 ポートを設定しています。
  /* 実機に応じて変更してください。
  SELECT_CASE VOID
    CASE SW(192)
      share_myadrs=0
      share_comport=2
    CASE SW(193)
      share_myadrs=1
      share_comport=1
    CASE SW(194)
      share_myadrs=2
      share_comport=1
    CASE SW(195)
      share_myadrs=3
      share_comport=4
    CASE_ELSE
      PRINT "ADDRESS?"
      END
  END_SELECT
```

```
  PRINT "MY ADRS" share_myadrs "COM PORT" share_comport
  CNFG# share_comport 485 "38400b8pns1NONE"
  PRINT# share_comport "*"                          /* Dummy 出力 485 Activate

  share_basebank=-900                               /* 共有する先頭アドレス
  share_stcount=4                                   /* ステーション数
  share_interval=200                                /* リフレッシュインターバル
  share_delay=25                                    /* Slave の送信遅延係数
  share_oldclk=0                                    /* St0 が送信した前回の SYSCLK
  share_slavesend=0                                 /* Slave の送信時間
  share_errcnter=7000                               /* エラーカウンタに用いる点番号
    /* エラーカウントは点データに格納しています
    /*  X(share_errcnter): チェックサムエラーカウント
    /*  Y(share_errcnter): タイムアウトカウント
    /*  U(share_errcnter): St 番号エラーカウント
  SETP share_errcnter 0 0 0 0                       /* エラーカウンタ初期化
  share_topbank=share_basebank-(share_stcount*8-1)
  FOR i_=share_basebank TO share_topbank STEP -1
    OUT 0 i_                                        /* 使用するエリアを初期化
  NEXT
  FORMAT ""

  DO

    IF share_myadrs==0 THEN                         /* 自分が St0 なら
      IF SYSCLK>(share_oldclk+share_interval) THEN
        share_oldclk=SYSCLK
        share_dt1=IN(share_basebank~Lng)            /* 自己エリア読込
        share_dt2=IN((share_basebank-4)~Lng)        /* 自己エリア読込
        share_sum=share_dt1^share_dt2               /* チェックサム
        SEND$="$0"+HEX$(share_dt1)+","+HEX$(share_dt2)+","+HEX$(share_sum)+"¥r"
        INPUT# share_comport CLR_BUF
        PRINT# share_comport SEND$                  /* 送信
      END_IF
    ELSE                                            /* Slave
      IF (SYSCLK=>share_slavesend)&(share_slavesend<>0) THEN  /* 送信時間計算かどうか
        GOSUB *mem_share_setdata
        share_slavesend=0
      END_IF
    END_IF

    IF LOF(share_comport)>1 THEN
      INPUT# share_comport CHR_C|1 share_st$
      IF share_st$=="$" THEN
        INPUT# share_comport CHR_C|1 share_st$
        share_st=ASC(share_st$)-&H30                /* St 番号
        IF (share_st>=0)&(share_st<share_stcount) THEN
          IF share_st==0 THEN                       /* St0 から来たなら
            share_slavesend=SYSCLK+share_myadrs*share_delay  /* 送信時間計算
          END_IF
          share_area=share_basebank-share_st*8 /* エリアを計算
          GOSUB *mem_share_getdata
        ELSE
          U(share_errcnter)=U(share_errcnter)+1 /* St 番号異常
        END_IF
      END_IF
    END_IF
    SWAP
  LOOP

/*
/* 各 St から来たデータを自分の当該エリアに書き込む
/*
*mem_share_getdata
  INPUT# share_comport TMOUT|2 share_st$            /* cr まで読む
```

```
  IF rse_<>0 THEN                                   /* Timeout
    Y(share_errcnter)=Y(share_errcnter)+1           /* タイムアウトカウンタ
    RETURN
  END_IF
  share_dt1=HEX(share_st$)                          /* 下位 4bytes
  SERCH share_st$ ","
  share_dt2=HEX(8)                                  /* 上位 4Bytes
  ptr_=ptr_+1
  share_sum=HEX(8)                                  /* チェックサム
  IF share_dt1^share_dt2^share_sum==0 THEN
    OUT share_dt1 share_area~Lng                    /* 当該 St のエリアに書込む
    OUT share_dt2 (share_area-4)~Lng                /* 当該 St のエリアに書込む
  ELSE
    X(share_errcnter)=X(share_errcnter)+1           /* チェックサムエラーカウンタ
  END_IF
  RETURN

/*
/* 自己エリアのデータを送信する
/*
*mem_share_setdata
  share_area=share_basebank-share_myadrs*8          /* 自己エリアを計算
  share_dt1=IN(share_area~Lng)                      /* 自己エリア下位読込む
  share_dt2=IN((share_area-4)~Lng)                  /* 自己エリア上位読込む
  share_sum=share_dt1^share_dt2                     /* チェックサム
  SEND$="$"+CHR$(share_myadrs+&H30)
  SEND$=SEND$+HEX$(share_dt1)+","+HEX$(share_dt2)+","+HEX$(share_sum)+"¥r"
  INPUT# share_comport CLR_BUF
  PRINT# share_comport SEND$                        /* 送信
  RETURN
```

RS-485 垂れ流し

- 機器構成、配線は RS-485 メモリ共有と同じですが、こちらは Host が定期的に発信、Slave は受信のみです。Host の時計データなどを伝える、Host と同期を取るというような場合はこれでも有用です。
- サンプルでは、Host は 2 つの変数をカンマ区切りで 1 つの文字列に連結、Slave はそれを受信して変数に分離します。

プログラム (各 St 共通)

【使用方法】
*rs485_broadcasting 以下を任意のタスクで実行します(バックグランドで通信するイメージです)。
作業タスクで、Host は時間と変数を送信文字列に設定、Slave は得られた変数を参照します。
通信タイミング生成に SYSCLK を使います。他のタスクでリセットや変更をしないで下さい。

```
  QUIT_FORK 20 *rs485_broadcasting                  /* 送受信タスク
  TIME 1000

  IF broad_myadrs==0 THEN
    QUIT_FORK 1 *HOST_BROAD                         /* Host 作業タスク
  ELSE
    QUIT_FORK 1 *SLAVE_DISP                         /* Slave 作業タスク
  END_IF

  END

*HOST_BROAD
COUNT=0
DO
  broad_send1$=TIME$(1)                             /* 時計データ(文字列)
  broad_send2$=STR$(COUNT)                          /* 変数
  PRINT broad_send1$ broad_send2$
  COUNT=COUNT+1
  TIME 500
LOOP

*SLAVE_DISP
DO
  HOSTTIME$="[Host Time: "+broad_inp1$+"]"          /* このサンプルの 1 番目のデータは文字列
  IF LEN(broad_inp2$)<>0 THEN
    COUNT=VAL(broad_inp2$)                          /* このサンプルの 2 番目のデータは数値
  ELSE
    COUNT=0
  END_IF
  PRINT HOSTTIME$ "COUNT" COUNT "Err" P(broad_errcnter)
  TIME 500
LOOP


/*
/* RS-485 ブロードキャスト
/*   Host からの垂れ流し。Slave からの応答は無い。
/*
*rs485_broadcasting

  /* このサンプルは入力で St 番号,RS-485 ポートを設定しています。
  /* 実機に応じて変更してください。
  SELECT_CASE VOID
    CASE SW(192)
      broad_myadrs=0
      broad_comport=2
    CASE SW(193)
      broad_myadrs=1
```

```
      broad_comport=1
    CASE SW(194)
      broad_myadrs=2
      broad_comport=1
    CASE SW(195)
      broad_myadrs=3
      broad_comport=4
    CASE_ELSE
      PRINT "ADDRESS?"
      END
  END_SELECT

  PRINT "MY ADRS" broad_myadrs "COM PORT" broad_comport
  CNFG# broad_comport 485 "38400b8pns1NONE"

  IF broad_myadrs<>0 THEN                          /* Slave なら
    PRINT# broad_comport "*"                       /* Dummy 出力 for listener
  END_IF

  FORMAT ""
  broad_send1$=""                                  /* Host 送信用
  broad_send2$=""                                  /* Host 送信用
  broad_inp1$=""                                   /* Slave 受信用
  broad_inp2$=""                                   /* Slave 受信用
  broad_interval=200                               /* インターバル
  broad_errcnter=7000                              /* エラーカウンタに用いる点番号
  SETP broad_errcnter 0 0 0 0                      /* エラーカウンタ初期化
  INPUT# broad_comport CLR_BUF

  DO
    IF broad_myadrs==0 THEN                        /* Host
      IF SYSCLK>(broad_oldclk+broad_interval) THEN
        broad_oldclk=SYSCLK
        broad_send$=broad_send1$+","+broad_send2$
        GOSUB *broad_sum_calc                      /* sum 計算、送信文字列作成
        PRINT# broad_comport broad_send$ "¥r"      /* 送信
      END_IF
    ELSE                                           /* Slave
      IF LOF(broad_comport)>0 THEN                 /* 受信バッファチェック
        INPUT# broad_comport CHR_C|1 broad_inp$    /* 先頭のキャラクタ確認

        IF broad_inp$<>"$" THEN                    /* 先頭キャラチェック
          INPUT# broad_comport CLR_BUF
          GOTO *broad_salveerr
        END_IF

        INPUT# broad_comport TMOUT|2 broad_inp$    /* cr まで受信
        IF rse_<>0 THEN                            /* タイムアウトなら
          Y(broad_errcnter)=Y(broad_errcnter)+1    /* タイムアウトカウンタ
          INPUT# broad_comport CLR_BUF
          GOTO *broad_salveerr
        END_IF

        GOSUB *broad_sum_check                     /* サムチェック
        IF broad_inp$=="" THEN                     /* チェックサムエラーなら
          X(broad_errcnter)=X(broad_errcnter)+1    /* チェックサムエラーカウンタ
          GOTO *broad_salveerr
        END_IF

        GOSUB *broad_split                         /* カンマで文字列分割

      END_IF
*broad_salveerr
    END_IF
    SWAP
  LOOP
```

```
*broad_sum_calc                          /* Host のサム計算
  ptr_=broad_send$
  sum_=0
  FOR i_=0 TO LEN(broad_send$)-1
    sum_=sum_^ASC(PTR$(1))
    ptr_=ptr_+1
  NEXT
  FORMAT "00"
  broad_send$="$"+broad_send$+HEX$(sum_) /* 先頭に$、末尾にチェックサム(2 文字)付加
  FORMAT ""
  /*PRINT broad_send$
  RETURN

*broad_sum_check                         /* Slave のサムチェック
  ptr_=broad_inp$
  broad_inbuf$=PTR$(LEN(broad_inp$)-2)   /* チェックサム文字列の前までコピー(本文)
  sum_=0                                 /* サム計算用
  FOR i_=0 TO LEN(broad_inp$)-3
    sum_=sum_^ASC(PTR$(1))
    ptr_=ptr_+1
  NEXT
  sumstr_=HEX(PTR$(2))                   /* チェックサム文字列を数値にする
  IF sum_==sumstr_ THEN                  /* 計算したサムと受信サム比較
    broad_inp$=broad_inbuf$
  ELSE
    broad_inp$=""
  END_IF
  RETURN

*broad_split                             /* Slave 受信文字列をカンマで分割
  /* 受信した文字列は broad_inp1$ と broad_inp2$ に入る
  SERCH broad_inp$ ","                   /* ptr_に broad_inp$中のカンマの位置が入る
  comma_=ptr_                            /* comma_=カンマの位置
  ptr_=broad_inp$                        /* ptr_に broad_inp$の先頭位置が入る
  copylen_=comma_-ptr_                   /* カンマまでの文字数
  broad_inp1$=PTR$(copylen_-1)           /* カンマの前までコピー
  ptr_=comma_                            /* ポインタをカンマの位置にセットし直す
  copylen_=LEN(broad_inp$)-copylen_      /* カンマより後の文字数
  broad_inp2$=PTR$(copylen_)
  RETURN
```

実行結果(一部)

```
 Host               Slave
 13:38:40 1929      [Host Time: 13:38:40] COUNT 1929 Err 0 0 0 0
 13:38:40 1930      [Host Time: 13:38:40] COUNT 1930 Err 0 0 0 0
 13:38:41 1931      [Host Time: 13:38:41] COUNT 1931 Err 0 0 0 0
 13:38:41 1932      [Host Time: 13:38:41] COUNT 1932 Err 0 0 0 0
 13:38:42 1933      [Host Time: 13:38:42] COUNT 1933 Err 0 0 0 0
 13:38:42 1934      [Host Time: 13:38:42] COUNT 1934 Err 0 0 0 0
```

--- End Of Doc ---