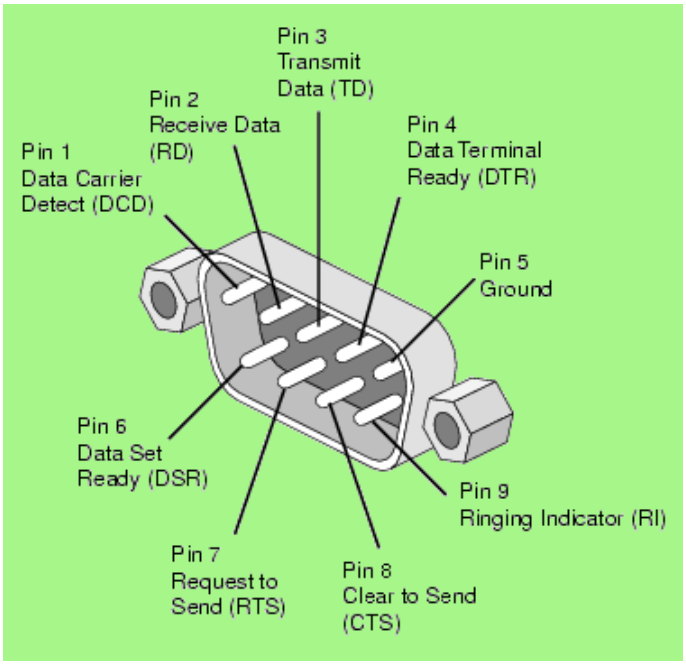
Zserial

The CP/M3 RSX for the Serial Interface

Ref-Manual



Inhaltsverzeichnis

RSX Call-Name:	Func.-No.:	ASM-Function:
Z_Hello	0	Hello
Z_RSXVersion	1	RSXVersion
Z_RSXName	2	RSXName
Z_SndChrStr	3	SndChrStr
Z_SendHLhex	4	SendHLhex
Z_SendLhex	5	SendLhex
Z_SndCrlf	6	SndCrlf
Z_GetHex	7	GetHex
Z_GetDec	8	GetDec
Z_TTYinput	9	TTYinput
Z_TTYoutput	10	TTYoutput
Z_TTYinit	11	TTYinit
Z_TTYres	12	TTYres
Z_SetBdRate	13	SetBdRate
Z_GetBdRate	14	GetBdRate

1. Introduction

This manual descrips the RSX library that supports the serial lines A...C on the Multicomp July2019. For Multicomp June2015 the port address for Serial-B and its BRG have to be adjusted to its actual values. Do this within the header file ZserDef.h. When initializing the RSX, the port address table is copied from the table within ZSerDef.h to the RSX internal address table. All further port access is done using this internal table. Consult the header file for more information about the table structure (look for ‚TTYcnf: … TTYcnf_end:‘).

As with the Xgraph-Package, this RSX can be used in a Stand-Alone way when disabling the RSX-Entry-Code and using the Test-Bench part for the user applicatione. How to do that is descriped in the asm-code section ‚tb_zserial:‘. There you can find a longer example-code which can be used as an entry. All functions that are listed in the jump-table preserve the values in the parameter field – helper subroutines do not !

2. Files in the ZSerial-Package

Text-Files:	***File-Size:***	***Comments:***
Ref-Manaul.pdf	284400 Byte	The ZSerial Ref-Manual as pdf
.h Header-Files:		
zserdef.h	6724 Byte	The definition file for constants
zserfunc.h	5451 Byte	The definitions of the c-functions
C Source-Files:		
zdemo1.c	4765 Byte	The c-source of ZSerial demo
CPM-Binaries:		
zserial.rsx	896 Byte	The ready to use RSX for CP/M3
zdemo1.com	2816 Byte	The compiled demo with attached RSX
Assembler-Sources:		
zserial.asm	29056 Byte	The RSX assembler source code

3. Const-Names of Func-Parameters

These values are function dependend used as parameters in ‚Zserial.h‘.

```
/* Baud-Rate indexes */
/* ================ */
#define ZB_B1200       0
#define ZB_B2400       1
#define ZB_B4800       2
#define ZB_B9600       3
#define ZB_B19200      4
#define ZB_B38400      5
```

```
#define ZB_B57600      6
#define ZB_B115200     7

/* Serial-A to C ACIA */
/* ================== */
#define ZS_SerialA     0  /* Serial-A Channel */
#define ZS_SerialB     1  /* Serial-B Channel */
#define ZS_SerialC     2  /* Serial-C Channel */
#define ZS_SerialD     3  /* Serial-D Wifi-Channel */

/* Serial-A to C Ctrl/Stat Reg. */
/* ============================ */
#define ZC_CtrlA       0  /* Serial-A Ctrl/Stat-Reg. */
#define ZC_CtrlB       1  /* Serial-B Ctrl/Stat-Reg. */
#define ZC_CtrlC       2  /* Serial-C Ctrl/Stat-Reg. */
#define ZC_CtrlD       3  /* Serial-D Ctrl/Stat-Reg. (Wifi) */

/* Serial-A to C Bd-Rate Reg. */
/* ========================== */
#define ZB_BRGA     0  /* Serial-A Bd-Rate Reg. */
#define ZB_BRGB     1  /* Serial-B Bd-Rate Reg. */
#define ZB_BRGC     2  /* Serial-C Bd-Rate Reg. */
#define ZB_BRGD     3  /* Serial-D Bd-Rate Reg. (Wifi) */

/* Serial Protocol */
/* =============== */
#define ZP_ACIA8N1      0x15    /* 8 Bit - No Parity - 1 Stop-Bit */

/* Others */
/* ====== */
#define Z_SIGNATURE    0xEBEB  /* RSX signature for Z_Hello function */
```

4. The I/O-Ports of the Ser.-A..-D & BRG-A..-D

It should be mentioned, that Serial-A/BRG-A are special. This line is managed by the BIOS and changing parameters has a direct feedback to the VGA-Terminal and its output speed. If logged-in on serial-A and this line is reprogrammed, the connection to a PC-Terminal app will break !

```
Data-Width: Port-Addr.: Comment:
===========+===========+==============================
Serial-A...-C Ports:
  1 Byte      080h      TTY-A: Ctrl/Stat port addr.
  1 Byte      090h      TTY-B: Ctrl/Stat port addr.
  1 Byte      084h      TTY-C: Ctrl/Stat port addr.
```

```
  1 Byte      086h      TTY-D: Ctrl/Stat port addr. (Wifi)

BRG-A...-C Ports:
  1 Byte      07Bh      TTY-A: Bd-Rate Gen. port addr.
  1 Byte      07Ch      TTY-B: Bd-Rate Gen. port addr.
  1 Byte      07Dh      TTY-C: Bd-Rate Gen. port addr.
  1 Byte      07Eh      TTY-D: Bd-Rate Gen. port addr. (Wifi)
```

5. Function Reference

All functions are collected in the following `.h`-files:

```
- ZSerDef.h      Constant definitions etc.
- ZSerfunc.h     Functions that handle the communication w. Serial-A...-D
```

---

6. Files in ZserDef.h

Function summary:

- `InitTTY()`
- `ResTTY()`
- `HelloZSerial()`
- `GetZSerVer()`
- `GetZSerName()`

6.1. Function `InitTTY()`

Initiailze the Serial Line Port-Addr. Array.

- **Function Declaration:**

```
InitTTY()
{
    z_func   = Z_TTYinit;
    z_par[0] = 0;       /* Dummy Value */
    z_par[1] = 0;       /* Dummy Value */
    z_par[2] = TTYcnf;  /* ptr to port addr. table */

    z_call();
}
```

- **Example:**

```
[…]
```

```
    Channel   = ZS_SerialC;
    Baud_Rate = ZB_B1200;
    DtaFormat = ZP_ACIA8N1;

    InitTTY();                              /* Set ACIA/BRG Port-Addr. */
    ResTTY(Channel,Baud_Rate,DtaFormat);  /* Reset Serial-Channel */
[…]
```

6.2. Function ResTTY()

Reset selected Serial Line Channel, initialisize Port-Addr, set Bd-Rate and empty Rec-Buffer.

- **Function Declaration:**

```
ResTTY(TTYchan,BdRate,TTYproto)
int TTYchan,BdRate,TTYproto;
{
    z_func   = Z_TTYres;
    z_par[0] = TTYchan;       /* TTY-Chan. */
    z_par[1] = BdRate;        /* Wanted Bd-Rate */
    z_par[2] = TTYproto;      /* Wanted Protocol-Format */

    z_call();
}
```

- **Example:**

See example for function `InitTTY()`.

6.3. Function HelloZSerial()

Check if the RSX is in memory. Returnvalue is `TRUE` or `FALSE`.

- **Function Declaration:**

```
HelloZSerial()
{
    z_func = Z_Hello;

    return z_call() == Z_SIGNATURE;
}
```

- **Example:**

```
if(!HelloZSerial()) {

    puts("The Zserial RSX is not in memory!");

    return -1;

}
```

6.4. Function `GetZSerVer()`

Readback RSX-Version No. Coded as 16-bit binary number.

- **Function Declaration:**

```
GetZSerVer()
{
    z_func = Z_RSXVersion;
    return z_call();
}
```

- **Example:**

```
data = GetZSerVer();
```

6.5. Function `GetZSerName()`

Readback RSX-Name as pointer to zero terminated ASCII-String.

- **Function Declaration:**

```
GetZSerName()
{
    z_func = Z_RSXName;
    return z_call();
}
```

- **Example:**

```
[…]
    Channel   = ZS_SerialC;
    Baud_Rate = ZB_B1200;
    DtaFormat = ZP_ACIA8N1;
    SendChrStr(Channel, GetZSerName());
[…]
```

7. Files in ZSerfunc.h

Function summary:

- SendChrStr();
- SendCRLF();
- SendWORD();
- SendBYTE();
- ReceiveHEX();
- ReceiveDEC();
- TTYinp();
- TTYout();
- SetBdRate();
- GetBdRate();

7.1. Function SendChrStr()

Send a Zero-terminated Chr-String to the serial line referenced in the function param. Field. Max. string length is 256 bytes. When this limit is reached the transmission silently terminates and the function returns. Bit7 of the chr. is NOT stripped. This must be done by the user app. The function is called with the serial line parameter and the address where the string is located as pointer.

- **Function Declaration:**

```
SendChrStr(TTYchan, TxtPtr)
int TTYchan; BYTE *TxtPtr;
{
    z_func   = Z_SndChrStr;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = TxtPtr;

    z_call();
}
```

- **Example:**

```
[…]
BYTE  *Message;
[…]
    Message   = "Hallo World"
    Channel   = ZS_SerialC;
```

```
    Baud_Rate = ZB_B1200;
    DtaFormat = ZP_ACIA8N1;
[…]
    InitTTY();                              /* Set ACIA/BRG Port-Addr. */
    ResTTY(Channel,Baud_Rate,DtaFormat);  /* Reset Serial-Channel    */
    SendCrlf(Channel);
    SendChrStr(Channel, "Send on Serial-C: ");
    SendChrStr(Channel, Message);
[…]
```

7.2. Function SendCRLF()

Sends a „CRLF“ on the specified serial line

- **Function Declaration:**

```
SendCRLF(TTYchan)
int TTYchan;
{
    z_func   = Z_SndCrlf;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    z_call();
}
```

- **Example:**

```
See Function SendChrStr().
```

7.3. Function SendWORD()

Send a 16-bit bin. number as a 4 charac. ASCII-Hex string. The HIGH-Byte of WORD is send first ! Any type of separater between the two 4 Byte ASCII-Hex strings must be inserted by the user app. Preferable separator is a <komma>.
Send hex-string will look like this: `'...,ABCD,1234,BB48,...'`
See also Function `RecWORD()` & `RecBYTE()`.

- **Function Declaration:**

```
SendWORD(TTYchan, data)
int TTYchan, data;
{
```

```
    z_func   = Z_SendHLhex;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = data;

    z_call();
}
```

- **Example:**

```
[…]
    Channel   = ZS_SerialC;
[…]
    SendWORD(Channel, data);
    SendChrStr(Channel, ",");
    SendWORD(Channel, data);
    SendChrStr(Channel, ",");
[…]
```

7.4. Function `SendBYTE()`

Send a 8-bit bin. number as 2 char. ASCII-Hex string. Any type of separater between 2 ASCII-Hex strings must be inserted by the user app. Preferable separator is a <komma>. See also Function `RecWORD()` & `RecBYTE()` & `SendWORD()`.

- **Function Declaration:**

```
SendBYTE(TTYchan, data)
int TTYchan, data;
{
    z_func   = Z_SendLhex;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = data;

    z_call();
}
```

- **Example:**

```
[…]
    Channel   = ZS_SerialC;
[…]
    SendBYTE(Channel, data);
    SendChrStr(Channel, ",");
    SendBYTE(Channel, data);
```

```
    SendChrStr(Channel, ",");
[…]
```

7.5. Function `ReceiveHEX()`

Receive a ASCII Hex Value (0..$FFFF). The function incorporates digits until a '$00' or **','** is detected. If almost 4 digits are accumulated the leftmost digit is shifted out to add the new one. Any non hex digit will produce erroneous results. The function will silently terminate after 8 consecutive received digits. The result will be returned as 16-bit binary in Reg. HL to the caller.

- **Function Declaration:**

```
ReceiveHEX(TTYchan)
int TTYchan;
{
    z_func   = Z_GetHex;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    return z_call();
}
```

- **Example:**

```
[…]
    Channel = ZS_SerialC;
    Data = ReceiveHEX(Channel);
    SendBYTE(Channel, data);
[…]
```

7.6. Function `ReceiveDEC()`

Receive a ASCII Decimal Value (0..65535) and return it as 16-bit binary. The function incorporates digits until a '$00' or ',' is detected. If almost 5 digits are accumulated any further received digit will give an out of range situation and Reg. HL contains $FF00 as the result to flag that. Any non decimal digit will produce erroneous results.

- **Function Declaration:**

```
ReceiveDEC(TTYchan)
int TTYchan;
{
```

```
    z_func   = Z_GetDec;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    return z_call();
}
```

- **Example:**

```
[…]
    Channel = ZS_SerialC;
    data = ReceiveDEC(Channel);
    SendBYTE(Channel, data);
[…]
```

7.7. Function `TTYinp()`

Receive Byte Data from TTY Channel. Returned value is 16-bit binary but only LByte will be valid, Hbyte is $00. No conversion is done, the received byte is return as is.

- **Function Declaration:**

```
TTYinp(TTYchan)
int TTYchan;
{
    z_func   = Z_TTYinput;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    return z_call();
}
```

- **Example:**

```
[…]
    Channel = ZS_SerialC;
    Data = TTYinp(Channel);
    SendBYTE(Channel, data);
[…]
```

7.8. Function `TTYout()`

Send Byte Data by selected TTY Channel. From 16-bit binary only LByte will be sendd, Hbyte is not used. No conversion is done, the byte is send as is.

- **Function Declaration:**

```
TTYout(TTYchan,data)
int TTYchan, data;
{
    z_func   = Z_TTYoutput;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = data;

    return z_call();
}
```

- **Example:**

```
[…]
    Channel = ZS_SerialC;
    data = 0x20;   /* Send a <blank> */
    TTYout(Channel, data);
[…]
```

7.9. Function `ChkTTYinp()`

Check for a received character in TTY Channel and return TRUE/FALSE depending on receives status of the checked channel.

- **Function Declaration:**

```
ChkTTYinpt(TTYchan)
int TTYchan;
{
    z_func   = Z_ChkTTYinp;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    return z_call() & 1;
}
```

- **Example:**

```
[…]
    Channel = ZS_SerialC;
    data = 0;
    if(ChkTTYinp(Channel)) then {
       data = TTYinp(TTYchan);
    }
[…]
```

7.10. Function `SetBdRate()`

Set selected BRG to the wanted Bd-Rate. Values are limited to 3 bit => 0..7

- **Function Declaration:**

```
SetBdRate(TTYchan, BdRate)
int TTYchan, BdRate;
{
    z_func   = Z_SetBdRate;
    z_par[0] = TTYchan;
    z_par[1] = BdRate;
    z_par[2] = 0;

    z_call();
}
```

- **Example:**

```
  […]
    Channel   = ZS_SerialC;
    Baud_Rate = ZB_B1200;
    SetBdRate(Channel,Baud_Rate);
  […]
```

7.11. Function `GetBdRate()`

Readback the act. Baud-Rate for TTY-Chan. The Bd-Rate index is returned in Reg. HL. Values are limited to 3 bit => 0..7, all other bits are zero. The readback of a BRG must be supported by the used hardware, otherwise the value is undefined.

- **Function Declaration:**

```
GetBdRate(TTYchan)
```

```
int TTYchan;
{
    z_func   = Z_GetBdRate;
    z_par[0] = TTYchan;
    z_par[1] = 0;
    z_par[2] = 0;

    return z_call();
}
```

- **Example:**

```
[…]
  Channel   = ZS_SerialC;
[…]
  data = GetBdRate(ZS_SerialB);      /* Read-Back Bd-Rate */
  SendCrlf(Channel);
  SendChrStr(Channel, "Readback Bd-Rate Serial-B: ");
  SendBYTE(Channel, data);
  SendChrStr(Channel, "h");
[…]
```

********** E N D **********