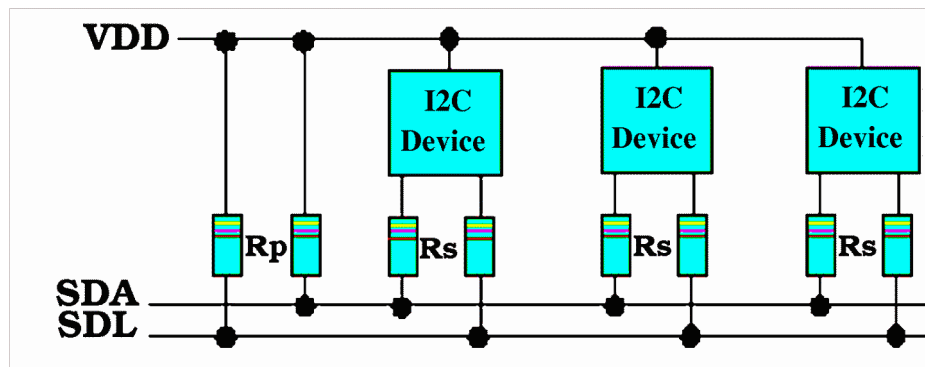


I2cRSX1.RSX

The RSX for the I2c-Interface

Ref-Manual



Inhaltsverzeichnis

1. Introduction.....	1	• Function Declaration:.....	4
1.1. I2c-Testadapter.....	1	• Example:.....	4
2. Compiling i2crsx1... and others.....	1	5. Functions in iwirefun.h.....	4
3. Function Reference.....	1	5.1. Function I2cClrBuff().....	4
4. Functions in iwireddef.h.....	2	• Function Declaration:.....	4
6.1. Function I2cInit().....	2	• Example:.....	5
• Function Declaration:.....	2	5.2. Function I2cCpyBuff().....	5
• Example:.....	2	• Function Declaration:.....	5
6.1. Function HelloI2c().....	2	• Example:.....	5
• Function Declaration:.....	3	5.3. Function I2cRdBuff().....	6
• Example:.....	3	• Function Declaration:.....	6
6.2. Function GetI2cVer().....	3	• Example:.....	6
• Function Declaration:.....	3	5.4. Function I2cWrBuff().....	6
• Example:.....	3	• Function Declaration:.....	7
6.3. Function GetI2cName().....	3	• Example:.....	7

RSX Call-Name:	Func.-No.:	ASM-Function:
I_Hello	0	Hello
I_RSXVersion	1	RSXVersion
I_RSXName	2	RSXName
I_I2c_Init	3	I2c_Init
I_I2c_ClrBuff	4	ClrBuff
I_I2c_CpyBuff	5	CpyBuff
I_I2c_WrBuff	6	WrBuff
I_I2c_RdBuff	7	RdBuff

1. Introduction

This RSX is pure software solution, using the bit-banging concept. This driver can operate only in Master-Slave mode. The behaviour is a streaming-mode like operation, what means the data to be send to or is received from the slave is handled by the user supplied buffer before the transfer or receiving data starts.

1.1. I2c-Testadapter

For testing the driver I've used a PCF8574A module from eBay (see picture). It receives the incoming byte and displays it on the connected 8 LED's. The last send byte can be read back, so both transfer directions can be tested. The first driver version was written in BASIC with the I2c-Speed reduced to 10Hz – BASIC isn't very speedy... So the tester was very helpful, especially for understanding the (in the beginning) confusing way the controller signals has to be checked by software to keep track with it's activities. Beside this a digital oscilloscope with build-in protocol

Picture 1: Ready to use Test-Adapter

Picture 2: The bare Module from eBay

decoder is very handy. This combo gave me a good overview about the activities of my „I2c-Baby“ when trying to confuse me.

2. Compiling i2crsx1... and others

For compiling all *.c and *.asm sources z-system makros as com-files are presented that automate this process:

mksetrtc.com, *mksetsys.com*, *mkshwrtc.com* and *clean.com*. Clean.com deletes all intermediate files produced by the assembler or MESCC-Compiler. That saves RAM-space on the RAM-Disk.

3. Function Reference

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

- IwireDef.h Constant definitions & RSX Base-Functions
- IwireFunc.h Functions that handle the I2c-communication

4. Functions in iwireddef.h

- I2cInit()
- HelloI2c()
- KillRSX()
- GetI2cVer()
- GetI2cName()

6.1. Function I2cInit()

Initialilze the I2c Port-Addr. Array.

- **Function Declaration:**

```
InitI2c(PORT)
{
    i_par[0] = 0;           /* Dummy Value */
    i_par[1] = 0;           /* Dummy Value */
    i_par[2] = PORT;        /* Port-Addr. Of I2c-Hardware */
    i_par[3] = 0;           /* Dummy Value */
    i_par[4] = 0;           /* Dummy Value */
    i_par[5] = 0;           /* Dummy Value */
    i_par[6] = 0;           /* Dummy Value */

    i_func  = i_I2cInit;
    i_call();
}
```

- **Example:**

```
[...]
    InitI2c(PORT);           /* Copy Port addr. to RSX */
[...]
```

6.1. Function HelloI2c()

If called checks whether the RSX is installed. If not the function returns

-1. The user programm should abort further execution.

- **Function Declaration:**

```
HelloI2c()
{
    i_par[0] = 0;                /* Don't touch Removal-Flag of RSX */
    i_func = I_Hello;
    return i_call() == SIGNATURE;
}
```

- **Example:**

```
[...]
RTRN = HelloI2c();              /* Check that RSX is in memory. */
if RTRN {
    return -1;
}
[...]
```

6.2. Function GetI2cVer()

Returns the Vers.-No. binary coded. Example: Rtn = 0x64 := 100₁₀.

- **Function Declaration:**

```
GetI2cVer()
{
    i_func = i_RSXVersion;
    return i_call();
}
```

- **Example:**

```
[...]
Version = GetI2cVer();          /* Get Version of RSX */
[...]
```

6.3. Function GetI2cName()

Get pointer as return value of zero terminated name string.

- **Function Declaration:**

```
GetI2cName()
{
    i_func = i_RSXName;
    return i_call();
}
```

- **Example:**

```
[...]
NameStr = GetI2cName();           /* Get Name of RSX */
[...]
```

5. Functions in iwirefun.h

- I2cClrBuff()
- I2cCpyBuff()
- I2cRdBuff()
- I2cWrBuff()

5.1. Function I2cClrBuff()

This function is ment to clear the buffer defined by BuffPtr or BuffPtr2. The buffer length is needed, therefore ByteCNT or ByteCNT2 has to be defined. If ByteCNT or ByteCNT2 is zero, the function will silently abort, so the user programm should check for a correct value. The allowed value range is 1..0xFFFF. No return value is returned in any case.

- **Function Declaration:**

```
I2cClrBuff(ByteCNT, BuffPtr)
int    ByteCNT;
BYTE *BuffPtr;
{
    i_par[0] = 0x00;    /* 'BYTE' to clear with */
    i_par[1] = 0;
    i_par[2] = 0;
    i_par[3] = ByteCNT;
    i_par[4] = BuffPtr;
    i_par[5] = 0;
    i_par[6] = 0;
```



```

    i_func    = I_ClrBuff;
    i_call();
}

```

- **Example:**

```

[...]
    I2cClrBuff(ByteCNT, BuffPtr);           /* Clear buffer */
[...]
```

5.2. Function I2cCpyBuff()

Copy data from Buffer-A to Buffer-B. „I2cCpyBuff()“ can handle overlapping source and destination situations. It's not really needed here, but who knows... If „ByteCNT“ is zero, the function will silently terminate and do nothing. No return parameter is delivered.

- **Function Declaration:**

```

I2cCpyBuff(ByteCNT, BuffPtr, BuffPtr2)
int    ByteCNT;
BYTE  *BuffPtr,*BuffPtr2;
{
    i_par[0] = 0;
    i_par[1] = 0;
    i_par[2] = 0;
    i_par[3] = ByteCNT;
    i_par[4] = BuffPtr;
    i_par[5] = 0;
    i_par[6] = BuffPtr2;

    i_func    = I_CpyBuff;
    i_call();
}

```

- **Example:**

```

[...]
    I2cCpyBuff(ByteCNT, BuffPtr, BuffPtr2); /* Copy buffer1 to buffer2 */
[...]
```

5.3. Function I2cRdBuff()

Reads ‚ByteCNT‘ data from slave to buffer pointed to by ‚BuffPtr‘. If an acknowledge-error is detected during receiving data, further action is aborted, the driver is STOPped and „FLAG" is returned, else „0". „FLAG" holds the value of the remaining number of bytes that could not be received from the slave. So it should be possible to determine the error point in the data stream. If the send address doesn't point to an existing slave, the return value is „ByteCNT". If „ByteCNT" is zero the function will terminate without doing anything and the return value is „-1". The range of „ByteCNT" is 1..0xFFFF.

- **Function Declaration:**

```
I2cRdBuff(I2cAddr, ByteCNT, BuffPtr)
int  I2cAddr, ByteCNT;
BYTE *BuffPtr;
{
    i_par[0] = 0;                /* Dummy-value */
    i_par[1] = I2cAddr;
    i_par[2] = 0;                /* Dummy-value */
    i_par[3] = ByteCNT;
    i_par[4] = BuffPtr;
    i_par[5] = ByteCNT;
    i_par[6] = BuffPtr;

    i_func  = I_RdBuff;
    return i_call();
}
```

- **Example:**

```
[...]
    ERROR = I2cRdBuff(I2cAddr, ByteCNT, BuffPtr); /* Read data from Slave */
[...]
```

5.4. Function I2cWrBuff()

Send contence of buffer „BuffPtr" is pointing to to slave. If an acknowledge-error is detected during reading in the data, further reading is aborted, the controller is STOPped and „FLAG" is returned, else „0". „FLAG" holds the value of the remaining number of bytes that could not be read-in. So it should be possible to determine the error point in the data stream. If the send i2c-address doesn't point to an existing slave, the returned value is „ByteCNT". If „ByteCNT" is zero

the function will terminate without doing anything and the return value is „-1“. The range of „ByteCNT“ is 1..0xFFFF.

- **Function Declaration:**

```
I2cWrBuff(I2cAddr, ByteCNT, BuffPtr)
int  I2cAddr, ByteCNT;
BYTE *BuffPtr;
{
    i_par[0] = ENA1WR; /* 'ENA1WR' = LOW-Byte, 'ENA1WR2' = HIGH-Byte = 0 */
    i_par[1] = I2cAddr;
    i_par[2] = I2cAddr;
    i_par[3] = ByteCNT;
    i_par[4] = BuffPtr;
    i_par[5] = ByteCNT;
    i_par[6] = BuffPtr;

    i_func  = I_StartRW;
    return i_call();
}
```

- **Example:**

```
[...]
    ERROR = I2cWrBuff(I2cAddr, ByteCNT, BuffPtr); /* Send data to Slave */
[...]
```