General overview

The MP3 cartridge contains an AtMega644 processor together with a VS1011 MP3 decoder. All registers of the decoder are shadowed by the AtMega and can be read by the MSX without additional wait states. After the MSX reads a register from the shadow memory, the AtMega reads the (new) contents of this register from the VS1011. If the shadow would be updated continiously, VS1011 registers which have auto incremented pointers will give unpredictable data. This update after read will give one byte delay when using user code or reading memory from a VS1011E.

All these registers are as described in the VS1011 datasheet, expect for 6 additional bits. These bits are not used by the VS1011 and are now used to switch between audio chip, enable interrupts and audio buffer warning level.

&H21 Any write to this address will reset the cartridge

&H22 This is the data port for registers read/write and audio write

&H23 Mode port:

- 7: When this bit it set in register mode, all written data is send to the register written in bits 0-4
- 6: 1=Switch to register mode *
  0=Switch to audio mode *
- 5: When set, auto increment is disabled when writing registers
- 4-0: Address of register to access

&H24 I2C Status, for reading back I2C errors, wait for finishing of last action
&H25 I2C Data, used for writing slave addresses and data
&H26 I2C Own address, not used in this application
&H27 I2C Control, for generating start/stop conditions

* Always read out the busy flag after an update to &H23

The I2C part is a completely separated part of the cartridge, this will be described in detail later on. There are two analog audio chips which both have their own audio registers, and of course the I2C controller itself which is directly controlled by the I/O addresses has registers as well.

To prevent any confusions, the AtMega (and VS1011) registers will be called ‘MP3 registers’. The audio chip registers ‘Audio registers’ and the I2C controller ‘I2C registers’.

Up to chapter ‘Audio settings and AUX input’ the document only describes the MP3 registers, IO &H22 and &H23.

MP3 Register description

Register:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 Mode	SetAUX	INTEnable	Keep0	Keep0	SNew	SShare	SDat	SClk	Keep0	Stream	TEST	Keep0	StopWAV	Reset	Keep0	InvLeft
1 Status					Buffer					StVer			StPD2	StPD1	StVol	
2 Bass									BassEnh				BassFrq			
3 ClockF	ClkD	ClkFreq														
4 DecodeTime	Playtime															
5 AuData	SmpFrq															Chans
6 WRAM	RAMData															
7 WRAMAddr	RAMAddress															
8 HDAT0	MP3BRate				MP3SRate		MP3Pad	MP3Priv	MP3Mode		MP3Ext		MP3CRgt	MP3Copy	MP3Emph	
9 HDAT1	MP3Header											MP3ID		MP3Lay		MP3Prot
10 AIAddr	UserSA															
11 Volume	VolLeft								VolRight							
12 AICTRL0	UserCtrl0															
13 AICTRL1	UserCtrl1															
14 AICTRL2	UserCtrl2															
15 AICTRL3	UserCtrl3															

🟩 VS1011 bits which might be usefull and are possible to modify by MSX user

▨ Should be kept 0, according to the VS1011 datasheet

🟨 Additional bits used by AtMega only. AtMega will mask the bits when sending the register sontent to the VS1011

🟥 Settings for communication between AtMega and VS1011. Changing thee bits will make the transferring of audiodata impossible.

SetAUX When set, audio settings written by I2C will affect the AUX input.

INTEnable When set, an interrupt will be generated when buffer level is reached.

SNew SDI bus. Audio data and register data are send trough the same bus.

SShare SDI bus. One chip select is used.

SDat SDI bus. When set, data should be LSB first

SClk SDI bus. When this bit is set, data is read at falling edge.

The SDI and SCI settings are affecting communication between AtMega and VS1011. Changing these bits will make further communication impossible.

Stream Set for stream mode, playback speed is adjusted depending on buffer.

It is not supported by the AtMega.

TEST When set, some test functions of the VS1011 can be executed

StopWAV Set this bit to terminate playback of a wav file.

Reset Setting this bit will reset the VS1011 player.

InvLeft When set, audio of left channel is inverted (180deg phase shifted).

Buffer In these 4 bits the buffer warning level is specified.

StVer (Read only) VS1011 device version.

StPD2 Power down VS1011 output, can be used to avoid transient at reset.

StPD1 Power down VS1011 analog.

StVOL Ouput 0=0dB, 1=-6dB, 3=-12dB. This is not meant for volume.

St bits are meant for power down, not useful for MSX.

BassEnh Bass enhancement, active when BassFrq>0.

BassFrq Lowest frequency the audio system can reproduce.

ClkD Clock doubler. When set, the crystal frequency is doubled.

ClkFreq Crystal/2000 ➔&H1800 by default.

For pitch bending the formula 100*&H1800 / Pitch can be used.

Playtime Setting the playtime to 0 should be done by writing 0 two times.

SmpFrq Current samplefrequency, can be changed while playing.

Chans The number of channels used. 0=Mono, 1=Stereo.

RAMData Data for VS1011 RAM,, written to address specified in RAMAddress

RAMAddress VS1011 RAM address pointer for data write.

MP3*** HDAT is the MP3 package header, found by the decoder, read only.

UserSA Activate or deactivate user code by writing its start address or 0 here.

VolLeft Volume left channel, this value times -0.5dB attenuation.

VolRight Volume right channel, this value times -0.5dB attenuation.

It is recommended to adjust volume by the I2C audio chips instead.

UserCtrl* These 4 registers are settings and feedback for the uploaded user code.

The highlighted bits can be useful for MSX applications.

MP3 packages

MP3 files are divided into packages, a package is recognized by the 11 ones at the beginning. For more detailed information, refer to ISO11172-3.

MP3Header All set to one.

MP3ID
00 – MPEG Version 2.5 (unofficial)
01 – reserved
10 – MPEG Version 2 (ISO/IEC 13818-3)
11 – MPEG Version 1 (ISO/IEC 11172-3)

MP3Lay
00 – reserved
01 – Layer III
10 – Layer II
11 – Layer I

MP3Prot When set, the file is protected by CRC

MP3BRate

Bits	Version1 Lay1	Version1 Lay2	Version1 Lay3	Version2 Lay1	Version2 Lay2 & Lay3
0000	Free	free	free	free	Free
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	Bad	bad	bad	bad	Bad

MP3SRate

bits	MPEG1	MPEG2	MPEG2.5
00	44100	22050	11025
01	48000	24000	12000
10	32000	16000	8000
11	reserv.	reserv.	reserv.

MP3Pad When set, the frame is padded

MP3Priv Free to use bit for MP3 encoders

MP3Mode
00 – Stereo
01 – Joint stereo (Stereo)
10 – Dual channel (2 mono channels)
11 – Single channel (Mono)

MP3Ext In joint-stereo mode, these bits indicate the real stereo subbands.

MP3CRgt When set, the file is copyrighted.

MP3Copy When set, the file is the original.

MP3Emph
00 – No emphasis

01 – 50/15 microseconds emphasis
10 – Reserved
11 – CCITT J17

Initialisation

After power up the AtMega processor enters a bootloader program which gives the possibility of flashing the main program. The bootloader part is protected at production to prevent the boot part to be damaged.

For normal operation, first write a zero to &H22… Or any value different from &HAA. In this case the AtMega will execute the normal program.

Accessing the cartridge

Data is send and read through port &H22, to distinguish between data and MP3 registers access to &H23 is used.

After a write action to &H23, the programmer should wait a while before sending new data. In case of changing register read pointer the wait time will be around 300us roughly. When updating registers with data send to &H22 previously, the wait time can be up to 8 milliseconds. The AtMega is not allowed to send MP3 register data in between audio data.

Writing MP3 registers

Writing only register 3:

```
        LD    A,&H40          ;Go to register mode
        OUT   (&H23),A
        CALL  ATWAIT

        OUT   (&H22),A        ;High byte
        OUT   (&H22),A        ;Low byte
        LD    A,&HC3          ;Update registers, offset is register 3
        OUT   (&H23),A
        CALL  ATWAIT          ;This wait can be milliseconds

ATWAIT: IN    A,(&H23)
        AND   A,16
        JR    NZ,ATWAIT
        RET
```

Writing 9 registers, starting from 0:

```
        LD    A,&H40          ;Go to register mode
        OUT   (&H23),A
        CALL  ATWAIT

        LD    HL,REGTAB
        LD    BC,&H1222       ;9 registers = 18 bytes
        OTIR

        LD    A,&HC0          ;Update registers, offset is register 0
        OUT   (&H23),A
        CALL  ATWAIT          ;This wait can be milliseconds

ATWAIT: IN    A,(&H23)
        AND   A,16
        JR    NZ,ATWAIT
        RET
```

Reading MP3 registers

Reading register 4:

```
LD    A,&H04           ;Can stay in audio mode, only update pointer
OUT   (&H23),A
CALL  ATWAIT

IN    A,(&H22)         ;High byte
…
IN    A,(&H22)         ;Low byte
```

Reading 16 registers, starting from 0:

```
LD    A,&H03           ;Can stay in audio mode, only update pointer
OUT   (&H23),A
CALL  ATWAIT

LD    HL,REGTAB
LD    BC,&H2022        ;16 registers = 32 bytes
INIR
```

Sending audio data

Audio data is send as raw MP3 of WAV data.

While playing audio, the buffer level will decrease, the cartridge sets bit 7 of IO &H23 when the audio buffer reaches its minimum level programmed by bits 8-11 of MP3 register 1. This bit will be cleared when the audio buffer is filled again to an amount of the minimum level plus 512 bytes.

It is possible to let the cartridge generate an interrupt by setting bit 14 of MP3 register 0, this interrupt will be generated once until the hardware detects an interrupt acknowledge from the Z80, or after a write to &H22. The interrupt service routine can recognize the interrupt also by checking bit 7 of IO &H23.

When the minimum buffer level at bits 8-11 of register 1 is set to 0, the warning bit is never set. When set to 1, the warning bit will be set when the buffer level is smaller than 256 bytes and reset when the buffer level is higher than 768.

The total buffer size is 3584 bytes. To choose the right buffer warning level, two values are important. The first one is the *number of bytes* that will be send each time and the second is the *period between two transfers*.

One clear rule is the buffer may not be overfilled because data will be lost. So when sending 1024 bytes each time, the highest possible buffer warning level will be 3584-1024=2560 bytes. When the warning level is set to a higher value e.g. 2816 and 1024 bytes will be send when bit 7 of &H23 is set, the buffer will overflow, 2816+1024 is more than the buffer size.

Second, if the time between transmitting data is high, the buffer warning should also be high. The exact time of the interval allowed depends on the bitrate.

In general, a higher warning level gives more safety. If the MSX for some reason needs more time between two transmissions, there is enough data left in the buffer to continue playing. It is recommended to send packages with a fixed time interval to achieve a known time shift between currently sent package and actual played part of the sound.

The cartridge always starts playing the audio data immediately after receiving the first complete MP3 package.

Status register

The information of currently read register, busy and data request can be checked by reading back I/O &H23.

&H23 Mode port:

7: When this bit it set, the programmed minimum buffer level is reached
Note: This bit will be reset some milliseconds after filling the buffer, don't check again too fast.

6: Not used, floating internally, might be 1 or 0

5: Not used, floating internally, might be 1 or 0

4: Busy. This should be checked after updating registers.

3-0: Address of register which data is available at I/O &H22 for reading

Writing VS1011 memory

For uploading and controlling user code, access to memory is needed. Writing is simply done by register 6 and 7. The first RAM address is written to register 7 and the data to 6. See next chapter.

The implementation of the AtMega for writing memory is only passing on data. One advantage, an amount of data smaller than 256 bytes meant for one register (&H06), can be written to &H22 at one time. Then send &HE6 to I/O register &H23. to send all data in one burst.

Reading VS1011 memory

For there is a difference between VS1011E and VS1011B devices. For VS1011E it is always possible to read memory at any time, the VS1011B does not support reading by default. The amplitude information of the spectrum analyzer requires memory read. Therefore the spectrum analyzer code is slightly different for these versions, in the VS1011B code the software creates a read possibility through register &H0F while the VS1011E uses the default register &H06.

The AtMega will read out 64 bytes of the VS1011 memory, starting at the last address written to register &H07. By default this should be part of the memory where the spectrum analyzer data is available because this is the most common application, for other future applications this value can be changed to another address.

VS1011 User code

The VS1011 has the possibility to execute small programs (plug-ins) written for a VS_DSP core. There are a number of plugins available at www.vlsi.fi, also a development kit with a C compiler is available. It is a small program which stays available in the VS1011 memory until reset or powerdown.

After compiling several files are created, the .bin files are not just binary but do also contain labels. A .cmd file can be used to upload to the VS1011. This is an example of a .cmd file:

```
W 2 7 8050
W 2 6 2800
W 2 6 2f40
W 2 6 0000
W 2 6 0024
W 2 7 8052
W 2 6 3e12
W 2 6 b817
W 2 6 3e12
W 2 6 3815
      …
      …
W 2 6 6800
W 2 6 0024
W 2 A 0050
```

The 3rd column is the MP3 register to write to, the last column the data, high byte first.

Register 7 is the address in RAM to write to, register 6 is the data and will de auto incremented by the VS1011. After uploading the code can be executed by writing the start address to register 10. Writing a 0 to register 10 will disable to code execution.

After a hard reset or write to I/O &H21, the spectrum analyzer is uploaded to the VS1011 by the AtMega as default user code.

To optimize the uploading (other) user code on MSX, after writing register 7 all data for register 6 can be send to the cartridge in packages of 128 words. When sending the ‘update registers’, set bit 5 to disable auto increment of the AtMega. In this case all data in the AtMega buffer will be sent to register 6 only.

Uploading the existing spectrum analyzer code by MSX, which has a size of 987 words, takes about 0.2 seconds in Z80 mode without this optimization.

Spectrum analyzer

At the moment of developing the cartridge there are two plug-ins available. A loudness and the spectrum analyzer. The loudness function works without any exceptions. For the spectrum analyzer reading back of registers is required, when using the VS1011B all spectrum data is available at register 15, this is covered by the AtMega. Since this is the only plug-in with data feedback, the AtMega software is tuned for this plug-in. The spectrum analyzer is automatically uploaded to the VS1011 at start up, any other user code tools can be written by MSX and will simply overwrite the spectrum analyzer.

To activate the spectrum analyzer, write its start address &H0050 to register &H0A.

When the plug in is active, the AtMega won't read out register 10 to update it's shadow memory. It seems to disturb the VS1011. All spectrum data of the VS1011 is read by the AtMega and stored in a different shadow memory behind the register shadow as described in chapter 'Read VS1011 memory'. Register 7 should be &H1384, also the startup value, because this is the first address where the spectrum data is available. The spectrum data can be read out by simply reading further than register 15 using an INIR or separate IN instructions.

All odd bytes are peak values, even bytes are actual values, the first pair of bytes is the lowest frequency and the last pair is the highest frequency. By default 14 frequency bands are selected, the mirror is updated every 20ms.

Reading 16 registers, starting from 0 including spectrum data:

```
LD     A,&H00          ;Can stay in audio mode, only update pointer (XOR A)
OUT    (&H23),A
CALL   ATWAIT

LD     HL,REGTAB
LD     BC,&H4022       ;16 registers = 32 bytes, 32 bytes of spectrum
INIR
```

Reading only spectrum data, start at register 15 and read e.g. 28 bytes of spectrum:

```
LD     A,&H0F          ;Can stay in audio mode, only update pointer (XOR A)
OUT    (&H23),A
CALL   ATWAIT

LD     HL,REGTAB
LD     BC,&H1E22       ;16 registers = 32 bytes, 32 bytes of spectrum
INIR
```

Audio settings and AUX input

Audio settings can be adjusted by two TDA8425 devices, they are controlled through a I2C controller PDA9564D. Audio settings are:

- Volume R*
- Volume L*
- Bass
- Treble
- Pseudo stereo
- Spatial stereo
- Forced mono / one channel

*Example code is included to create balance by software

The first audio chip is used for controlling the MP3 music, the second audio chip is connected to an auxiliary input and is mixed with the MP3 audio. Since both TDA8425 devices have the same I2C slave addresses, switching between the chips is done by bit 15 of MP3 register 0. This is one of the unused VS1011 bits. When this bit is set, the AUX audio chip is enabled.

Sending all settings for one audio chip using I2C takes 1.5ms for Z80 and 0.8ms in R800 mode. Changing only one register of an audio chip will take 0.4ms for an MSX2.

I2C

The IO addresses &H24 to &H27 controlled by the MSX are the I2C registers of the PCA9564D. The audio chips are accessed through this controller, according to the I2C protocol, a number of actions must be taken to change one single audio register.

Writing data over the two wire I2C bus is done by writing the data to I2C register &H25, poll I2C register &H24 to check for errors or finishing data transfer. Initiating and closing a data transmission is done via I2C register &H27.

```
TDAINI: LD    A,&H44          ;After power-up initialize oscillator
        OUT   (&H27),A        ; communication is 88kHz
        LD    B,0
WAIT:   DJNZ  WAIT

        LD    A,&H64          ;Initiate start condition
        OUT   (&H27),A        ; For I2C, all communication starts with a start
        LD    C,&H08          ;  condition. The initiator (master) keeps SDA (Data)
        CALL  STAT            ;  low, just before it keeps SCL (Clock) low.
        JP    C,I2CERR        ;Wait for code 8 in I2C status register
                              ;In case of collisions or shorted lines, exit
        LD    A,&H82          ;Load audio chip's slave address in buffer
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data.
        OUT   (&H27),A        ; All devices on a I2C bus have their own address.
        LD    C,&H18          ;  just like the address bus in the MSX, peripherals
        CALL  STAT            ;  can see for which one the data is meant.
        JP    C,I2CERR        ;In case of no acknowledge, exit

        XOR   A               ;Load audio register number in buffer (number 0)
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A        ; After the of the device address, the address of
        LD    C,&H28          ; the register of the device to write to is send.
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,(HL)          ;Load data for audio register 0 in buffer
        INC   HL
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,(HL)          ;Load data for audio register 1 in buffer
        INC   HL              ; I2C devices are usually auto incrementing the register
        OUT   (&H25),A        ;  number
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,(HL)          ;Load data for audio register 2 in buffer
        INC   HL
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit
        LD    A,(HL)          ;Load data for audio register 3 in buffer
        INC   HL
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,&H64          ;Generate (repeated) start condition
        OUT   (&H27),A        ; After reinitializing the bus, slave address
        LD    C,&H10          ;  and register must be send again. Now can we
        CALL  STAT            ;  only send 2 bytes to reach register 8
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,&H82          ;Load audio chip’s slave address in buffer
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H18
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,&H08          ;Load audio register number in buffer (number 8)
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,(HL)          ;Load data for audio register 8 in buffer
        INC   HL
        OUT   (&H25),A
        LD    A,&H44          ;Send buffer data
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR        ;In case of no acknowledge, exit

        LD    A,&H54          ;Generate stop condition
        OUT   (&H27),A        ; Other I2C masters now know they can send their
        LD    C,&HF8          ;  data.
        CALL  STAT
        JR    C,I2CERR        ;In case of any bus error, exit

        LD    A,&H44
        OUT   (&H27),A

        RET

STAT:   LD    B,0
STAL:   IN    A,(&H27)
        AND   8
        JR    NZ,STAC
        DJNZ  STAL
        SCF
        RET
STAC:   IN    A,(&H24)
        CP    C
        JR    Z,STAOK
        SCF
STAOK:  RET
```

Balance

```
TDAINI: LD    A,&H64
        OUT   (&H27),A
        LD    C,&H08
        CALL  STAT
        JP    C,I2CERR

        LD    A,&H82
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H18
        CALL  STAT
        JP    C,I2CERR

        XOR   A
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR

        LD    E,(HL)          ;Volume
        INC   HL
        LD    A,(HL)          ;Balance
        PUSH  HL
        CP    128
        JR    C,BALL
        LD    A,E
        JR    NOBALL
BALL:   SLA   E               ;Vol*2
        CALL  MUL8            ;HL=A*E
        LD    A,H             ;A=HL/256
NOBALL: OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        POP   HL
        JP    C,I2CERR

        DEC   HL
        LD    E,(HL)          ;Volume
        INC   HL
        LD    A,(HL)          ;Balance
        PUSH  HL
        CPL
        CP    127
        JR    C,BALR
        LD    A,E
        JR    NOBALR
BALR:   SLA   E               ;Vol*2
        CALL  MUL8            ;HL=A*E
        LD    A,H             ;A=HL/256
NOBALR: OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        POP   HL
        JP    C,I2CERR

        INC   HL
        LD    A,(HL)
        INC   HL
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR

        LD    A,(HL)
        INC   HL
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR

        LD    A,&H64
        OUT   (&H27),A
        LD    C,&H10
        CALL  STAT
        JP    C,I2CERR

        LD    A,&H82
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H18
        CALL  STAT
        JP    C,I2CERR

        LD    A,&H08
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR

        LD    A,(HL)
        INC   HL
        OUT   (&H25),A
        LD    A,&H44
        OUT   (&H27),A
        LD    C,&H28
        CALL  STAT
        JP    C,I2CERR

        LD    A,&H54
        OUT   (&H27),A
        LD    C,&HF8
        CALL  STAT
        JR    C,I2CERR

        RET

STAT:   LD    B,0
STAL:   IN    A,(&H27)
        AND   8
        JR    NZ,STAC
        DJNZ  STAL
        SCF
        RET
STAC:   IN    A,(&H24)
        CP    C
        JR    Z,STAOK
        SCF
STAOK:  RET
```

Updating firmware

1 Generate a hardware reset by writing to I/O address &H21

2 Write to I/O address &H22:

&HAA
&H55
&HAA
Size of block to program, high byte
Size of block to program, low byte
Program data byte by byte

After each transmitted byte, the value read back on I/O &H22 will de incremented by one when the cartridge is ready to receive a new byte. Bytes which are written when the cartridge is not ready will be lost.

The FLASH consists of pages of 256 bytes, so after each 256 data bytes the AtMega starts flashing one block. This takes more processing time than other bytes.

The data can be verified by reading back the program data as described in the next chapter.

Reading program memory

1 Generate a hardware reset by writing to I/O address &H21

2 Write to I/O address &H22:

&HAA
&H55
&H55

All data can be read sequentially at I/O address &H22.

Known issues

Hardware interrupt acknowledge does not work in R800 mode. The interrupt request is also ended after a write to &H22, the interrupt service routine must take care of that.