1 aXbo – PC Schnittstellen Protokoll

Im Folgenden wird die Schnittstelle zwischen dem Schlafphasenwecker und einer RS232 am PC beschrieben.
Für die RS232 gilt: 115200 Baud, 8N1 (8 Datenbits, keine Parität, 1 Stopbit)
Die Schnittstelle ist als Halfduplex Verbindung ausgeführt, und wird immer im Master- Slave Betrieb betrieben, aXbo ist immer der Slave.
Alle gesendeten Protokolle werden als Echo auch wieder empfangen!

1.1 Daten Frame: aXbo zu PC

	Start		Pollbit	Adresse	Datenbytes	Ende		BCC
Bezeichnung	DLE	STX				DLE	ETX	
Bits	8	8	1	7	~136	8	8	16
Typische Werte (hex)	10	02	01 oder 81		26 41 31 30 ...	10	03	XX YY h-Byte l-Byte

Jeder Datenframe beginnt mit der Startsequenz und endet mit der Checksumme. Das Pollbit wird bei jedem neuen Protokoll geändert. Bei einer Wiederholung des gleichen Frames bleibt es unverändert.

BCC Block Check Characters; Es wird ein eine 16 Bit Checksumme durch Addition beginnend mit dem Adressbyte (inkl. Pollbit) bis zum letzten Datenbyte ermittelt. Die beiden Bytes der Checksumme können jeden binären Wert annehmen.

DLE.....Data Link Escape
STX.....Start Of Text
ETX.....End Of Text

Zusätzlich ist ein **Byte Stuffing** notwendig. Da in den Datenbytes binäre Daten gesendet werden und keine Anzahl der Datenbytes an einer fixen Stelle übertragen wird, kann die Endsequenz beginnend mit DLE nicht garantiert erkannt werden. Deshalb wird für den Fall, das in den Datenbytes ein DLE vorkommt (0x10) ein zweites DLE gesendet.
D.h. wenn zwei DLE hintereinander empfangen werden war das ein Datenbyte mit dem Wert 0x10. Kommt nach einem DLE ein ETX so beginnt damit das Protokollende.

1.2 Protokoll Bestätigung

Die Protokoll Bestätigung vom PC zu aXbo entfällt. Wenn der PC eine fehlerhafte Antwort erhält muss er als Master eine Wiederholung starten.

Für das Pollbit gilt: Es muss für das ACK oder NAK Protokoll das Pollbit des gerade empfangenen (zu bestätigenden) Protokolls übernommen werden.

	Acknowledge	Pollbit	Adresse
Bezeichnung	ACK		
Bits	8	1	7
Typische Werte (hex)	06	01 oder 81	

Falls der PC den Frame nicht richtig empfangen hat (Checksumme falsch):

	Negative Acknowledge	Pollbit	Adresse
Bezeichnung	NAK		
Bits	8	1	7
Typische Werte (hex)	15	01 oder 81	

Wird ein Protokoll nicht innerhalb von 125 ms bestätigt muss vom PC eine Wiederholung erfolgen.

1.3 Datenbyte Format

Für die tatsächlichen Nutzdaten wurden folgendes Format vereinbart.

	Start	Person	Zeitstempel	Positions-wechsel	Impulse horizon.	Impulse vertikal	Protokoll Typ
Bezeichnung	&	A	YYMMDDhhmmss	P	H	V	T
Format	ASCII	ASCII	ASCII	ASCII	BINÄR	BINÄR	BINÄR
Bits	8	8	96	8	8/16	8/16	8
Typische Werte (hex)	26	41	31 30 35 30 35 30	50 oder 20			

& .....Das kaufmännische Und kennzeichnet den Beginn der Daten.
A .....kennzeichnet die Person. Hier könne die Buchstaben A bis Z ausgegeben werden.
Zeit .....Zeitstempel des Ereignisses im Format Jahr, Monat, Tag, Stunden, Minuten, Sekunden - YYMMDDhhmmss
P .....Positionswechsel erkannt oder sonst ein Blank (Leerzeichen)
H .....Anzahl der Impulse des horizontalen Sensors seit dem letzten Ereignis
V.....Anzahl der Impulse des vertikalen Sensors seit dem letzten Ereignis
T .....Protokolltyp: Das letzten Zeichen im Protokoll definiert den Protokoll Typ.

- T = Taste Am Sensor wurde die Taste betätigt
- B = Beginn Bewegungsdaten vom Sensor (von den beiden Neigungs-sensoren)
- N = Nächstes Nächste Bewegungsdaten (4sec nach dem letzten B oder N Protokoll)
- P = Personenwechsel Die Personenkennung am Sensor wurde geändert **A**, C, E, G bzw. **B**, D, F, H
- S = Sensor Sleep Sensor wurde deaktiviert (es werden keine weiteren Funkprotokolle gesandt)
- O = Sensor ON Sensor wurde wieder aktiviert
- W = Wake Letztmöglicher Alarm wurde ausgelöst
- D = Default *Ein Funkprotokoll mit undefinierten Daten wurde empfangen. Sollte nicht vorkommen!*

Neue Kommandos ab V1.80

- G = Good Wake Optimaler Alarm wurde ausgelöst
  Beispiel: C070704133911 G...

- Z = Weckzeitpunkt Der Weckzeitpunkt wird ausgegeben. Diese Protokoll wird am Beginn des Weckzeitbereichs ausgegeben. Der eigentliche eingestellt Weckzeitpunkt ist 30 min später. Die Personen Kennung ist immer A oder B!
  Beispiel: A070704133900 Z...

- C = Chillout Die Chillout Funktion wurde gestartet.
  Beispiel: D070704133330 C...

- F = Sound File Bei der Auswahl eines Soundfiles wird ein Protokoll mit der File Nummer übertragen. Die Sound Nummer wird als ASCII Zeichen (0x31 bis 0x36) an der Stelle der horizontalen Impulse nach dem Blank gesendet. Die Personen Kennung ist immer A oder B!
  Beispiele: A070704134021 1F...
  B070704134048 6F...

- K = PowerNapping Wenn die PowerNapping Funktion von einer Person gestartet wird erfolgt eine Protokoll Ausgabe. Die Personen Kennung ist immer A oder B!

- R = Random Zufälliger Weckzeitpunkt im Weckzeitbereich ist erreicht. Ab V200

- I = Snooze Snooze Funktion wurde nach erstem Wecken aktiviert. Ab V210

- X = Ende Die Ausgabe der Protokoll Daten wurde beendet (letztes Protokoll)

Zusätzliche Protokolle für den TESTMODE bei der aXbo Fertigung

- b = Back Taste Die BACK Taste wurde am Wecker betätigt.
- h = Home Taste Die HOME Taste wurde am Wecker betätigt.
- c = Click Taste Die Scrollrad CLICK Taste wurde am Wecker betätigt.
- u= Scrollrad Up Das Scrollrad wurde in Richtung UP gedreht
- d= Scrollrad Down Das Scrollrad wurde in Richtung DOWN gedreht

Bei den Impulsen von den Sensoren werden die Anzahl der Kontakt Öffnungen bzw. Kontakt Schließungen in den beiden Nibblen des jeweiligen Bytes angezeigt. Oberer Nibble zeigt Öffnen an – unterer Nibble zeigt Schließen an.

ZB: 0x10 = Kontakt ist geöffnet worden
0x01 = Kontakt ist geschlossen worden
0x11 = Kontakt wurde innerhalb der Zeit (0,3 bzw. 4 sec) geöffnet und geschlossen
0x12 = Kontakt wurde innerhalb der Zeit (0,3 bzw. 4 sec) geöffnet und 2x geschlossen

1.4 Logdaten

Ab Version V20 hat der Schlafphasenwecker auch eine Datenspeicherung der Bewegungen im Flash integriert.
Für den Online betrieb werden die Daten nach wie vor per RS232 gesendet gleichzeitig aber auch in das Flash im Wecker gespeichert.

Für die Speicherung der Logdaten sind 128kByte im Flash reserviert. Damit sollte für 2 Personen die Speicherung für mindestens 14 Tage möglich sein (1 Monat für eine Person).

Zur Unterscheidung dieser vielen Datensätze wurde auch eine Datumsberechnung in den Wecker eingebaut. Diese ist nur über die Datenprotokolle sichtbar. Für die Einstellung des Datums ist ein Kommando eingebaut worden mit dem über die RS232 das Datum gesetzt werden kann.

1.5 Datum setzen

Das Kommando zum Setzen des Datums ist aufgebaut wie unter Punkt 18.1 beschrieben.
Das Kommando ist 0xC8 die Datenbytes sehen wie folgt aus:

C8 YYMMDDhhmmss zB: C8 050429134700
Datum: 2005 April 29 Uhrzeit: 13:47:00

zB: 0x10 0x02 0x01
0xC8 0x00 0x05 0x00 0x04 0x02 0x09 0x01 0x03 0x04 0x07 0x00 0x00
0x10 0x03 0x00 0xEC

1.6 Log Daten auslesen

Zum Auslesen der Logdaten ist ein weiterer Befehl notwendig. Dieser ist auch wie unter Punkt 18.1 beschrieben aufgebaut. Das Datenbyte bestehen nur aus dem Kommando 0xCD

zB: 0x10 0x02 0x01
0xCD
0x10 0x03 0x00 0xCE

Die Ausgabe der Daten Protokolle erfolgt ohne Pause zwischen den einzelnen Datensätzen. Die Ausgabe der ca. 14500 Datensätze dauert somit ca. 40 sec.

Beim Auslesen der werden alle Daten im Flash beginnenden mit den ältesten ausgegeben. Das Ende wird durch eine Wiederholung des letzen Datensatzes gekennzeichnet wobei der Protokolltyp auf ein „X“ gestellt wird.

Während dem Auslesen der Logdaten sind Funk und Audio des Weckers nicht verfügbar.

1.7 Log Daten löschen

Zum Löschen der Logdaten ist ein weiteres Kommando 0xCE implementiert worden. Die Sicherheitsabfragen vor dem Löschen etc. müssen von der Anwendung am PC erfolgen!

zB: 0x10 0x02 0x01
0xCE
0x10 0x03 0x00 0xCF

1.8 SPW Hard und Software Status ab V1.13

Für Software Updates ist ab der Version V1.13 ein Kommando zum Auslesen folgender Daten implementiert:

- Software Version
- Hardware Version
- RTC Kalibration Werte

Das Kommando sieht wie folgt aus:

zB: 0x10 0x02 0x01
0x36
0x10 0x03 0x00 0x37

Die Antwort des Weckers kann wie folgt aussehen:

0x10 0x02 0x00 0x20 0x20 0x20 0x56 0x31 0x31 0x36 0x20 0x20
0x20 0x31 0x30 0x20 0x00 0x05 0xF9 0x20 0x10 0x03 0x03 0x4D

Die darin enthaltenen Daten sind :

- Text der Software Version: „ V116 “
- Hardware Version: „10“
- RTC Kalibrations Wert: 0x00 0x05 0xF9
- Trennzeichen: 0x20

1.9 SPW Hard und Software Status ab V1.70

Seit V1.70 wurde zusätzlich noch das Schreiben der Seriennummer in den Wecker vorgesehen. Es können daher mit dem Kommando zum Hard- und Software Status folgende Daten ausgelesen werden:

1. Kommando an Wecker (Status Anfrage):

00 10 02 01 **39** *10 03* 00 3A

Mögliche Antwort:

10 02 00 **A4** *10 03* 00 A4

2. Kommando an Wecker (Status Anfrage):

00 10 02 81 **39** *10 03* 00 BA

Mögliche Antwort:

10 02 80 **A4** *10 03* 01 24

3. Kommando an Wecker:

00 10 02 01 **36** *10 03* 00 37

Mögliche Antwort:

10 02 00 20 20 20 56 31 37 30 20 20 20 31 30 20 00 05 F9 20 42 65 6E 65 31 32 33 34 20 *10 03* 05 B1

Beim Lesen der Wecker Kenndaten sind folgenden Daten enthaltenen:

- Text der Software Version: 20 20 56 31 37 30 20 20 „ V170 “
- Hardware Version: 31 30 „10“
- RTC Kalibrations Wert: 0x00 0x05 0xF9
- Seriennummer: 42 65 6E 65 31 32 33 34 „Bene1234“
- Trennzeichen: 0x20

1.10 Verfügbare Test Kommandos am Wecker ab V1.70

Es wurden Testkommandos per Schnittstelle für den einfacheren Fertigungstest eingebaut:

- Der Wecker lässt sich in einen Testmodus versetzen. Im Testmodus werden alle Tastenbetätigungen, sowie Scrollrad Aktionen als Datenprotokoll per Schnittstelle an einen PC gesandt.
- Alarm Sound Person B abspielen
- Funk deaktivieren zum besseren RTC Kalibrieren und 1:1 Tastverhältnis mit 8Hz
- Wecker Selbsthalte Schaltung beenden (Clear Aktive). Damit kann man die Tiefentlade Schaltung aktivieren ohne eine Batterie am Wecker herausnehmen zu müssen. Der Wecker schaltet sich komplett ab – es ist keine Anzeige am Display mehr zu sehen. Wenn dieses Kommando verwendet wird darf über den USB Umsetzer keine Versorgungsspannung zum Wecker geführt werden!
  (R813 auf diesem Test USB-Umsetzer nicht bestücken)
- Wecker Reset
- eine 8 stellige Seriennummer (0-9, A-Z, a-z) lässt sich in den Wecker schreiben.
- Auslesen der Wecker spezifischen Kenndaten (Software Version, Hardware Version, RTC Kalibrationswerte, Seriennummer)

	Protokoll Typ	Kommando an Wecker	Antwort von Wecker
1	Status Anfrage	*00 10 02 01* **39** *10 03* 00 3A	*10 02 00* **A4** *10 03 00 A4*
2	Status Anfrage	*00 10 02 81* **39** *10 03* 00 BA	*10 02 80* **A4** *10 03 01 24*
3	Test Mode	*00 10 02 01* **C0 00 0A** *10 03* 00 CB	*06 01*
4	Start Alarm	*00 10 02 01* **C0 01 0A** *10 03* 00 CC	*06 01*
5	RTC Kalibration	*00 10 02 01* **C0 02 0A** *10 03* 00 CD	*06 01*
6	Clear Active	*00 10 02 01* **C0 04 0A** *10 03* 00 CF	*06 01*
7	Software Reset	*00 10 02 01* **C0 08 0A** *10 03* 00 D3	*06 01*
8	Serien Nummer	*00 10 02 01* **C5** 42 65 6E 65 31 32 33 34 *10 03* 03 0A	*06*
9	Wecker Kenndaten auslesen	*00 10 02 01* **36** *10 03* 00 37	*10 02 00* 20 20 20 56 31 37 30 20 20 20 31 30 20 00 05 F9 20 42 65 6E 65 31 32 33 34 20 *10 03* 05 B1

Die beiden ersten Kommandos zur Statusanfrage dienen nur zum Synchronisieren der Schnittstelle. Danach kann eines der Kommandos 3 bis 8 an den Wecker abgesetzt werden.

C0 – Kommando Kennung
0A – Testmode Timeout
01 – Testkommando Kennung
10 – Protokoll Bytes
81 – Adresse
D3 – Checksummen Bytes

Beim Lesen der Wecker Kenndaten sind folgenden Daten enthaltenen:

- Text der Software Version: 20 20 56 31 37 30 20 20 „ V170 “
- Hardware Version: 31 30 „10“
- RTC Kalibrations Wert: 0x00 0x05 0xF9
- Seriennummer: 42 65 6E 65 31 32 33 34 „Bene1234“
- Trennzeichen: 0x20

1.11 Ausführliche Beispiele

Anbei eine Sequenz mit der das Auslesen der Daten auf jeden Fall funktionieren muss. Es werden dabei einfach zuerst zwei Dummy Kommandos gesendet und dabei das Togglebit geändert. Nach dem zweiten Dummy Kommando ist für das nächst (Nutz-)Kommando das Togglebit damit sicher klar gestellt.

1.11.1 Lese Hard- Software und RTC Werte (V1.13 bis V1.64)

1. Kommando zum Wecker: 00 10 02 01 39 10 03 00 3A
   Antwort vom Wecker: 10 02 00 A4 10 03 00 A4

2. Kommando zum Wecker: 00 10 02 81 39 10 03 00 BA
   Antwort vom Wecker: 10 02 80 A4 10 03 01 24

3. Kommando zum Wecker: 00 10 02 01 36 10 03 00 37
   Antwort vom Wecker: 10 02 00 20 20 20 56 31 31 36 20 20 20 31 30 20 00 05 F9 20
   10 03 03 4D

Das 1. Kommando beginnt mit 00 damit der Wecker sicher genug Zeit hat um in den UART Modus zu schalten. Danach folgt das Kommando beginnend mit 10 02 ... bis zur Checksumme ... 00 3A. Die dargestellten Werte sind alles HEX Zahlen!

Beim 2. Kommando ist nur das Togglebit geändert (81).

Das 3. Kommando ist jenes zum Lesen der Hardware, Software, RTC Daten. Dieses Kommando hat wieder ein gelöschtes Togglebit (01)

1.11.2 Lese Logbuch

1. Kommando zum Wecker: 00 10 02 01 39 10 03 00 3A
   Antwort vom Wecker: 10 02 00 A4 10 03 00 A4

2. Kommando zum Wecker: 00 10 02 81 39 10 03 00 BA
   Antwort vom Wecker: 10 02 80 A4 10 03 01 24

3. Kommando zum Wecker: 00 10 02 01 CD 10 03 00 CE
   Antwort vom Wecker: 06 01

10 02 80 26 41 30 36 30 32 31 31 30 36 32 35 32 31 20 10 10 00 42 10 03 03 B3
10 02 00 26 41 30 36 30 32 31 31 30 36 32 36 33 34 20 01 00 42 10 03 03 29
10 02 80 26 41 30 36 30 32 31 31 30 36 32 36 34 39 20 10 10 00 42 10 03 03 BE

Letztes Protokoll: Person A, 06 02 11, 06:26:49, Sensor A wurde geöffnet, Beginn

Der Wecker antwortet auf das 3. Protokoll mit einem ACK und beginnt dann mit der Datenausgabe. Dabei sieht man das sich Ändernde Togglebit. Die Länge der Datenzeilen ist unterschiedlich wegen des Byte Stuffings.

Wichtig: Die Daten kommen nun ohne Pause zwischen den einzelnen Protokollen.

2 Protokollbefehle

Das 1. Datenbyte ist immer der Protokollbefehl. Das Bit 7 im Protokollbefehl unterscheidet zwischen Schreibbefehl (Bit 7 = 1, Aktion ausführen) und Lesebefehl (Bit 7 = 0, Daten werden angefordert).

Die folgende Tabelle listet alle verfügbaren Befehle auf.

Befehlbezeichnung	HEX	Beschreibung	Parameter	Antwort
DUMMY	0xCB	Dummybefehl für Kommunikationsstart	keine	ACK
FLASH_PAGETOBUFFER	0xBB	Flashseite in RAM- Puffer übertragen	Siehe unten	ACK
FLASH_BUFFERTOPAGE	0xBD	RAM- Puffer in Flashseite übertragen	Siehe unten	ACK
FLASH_BUFFERWRITE	0xBC	Flash RAM- Puffer beschreiben	Siehe unten	ACK
FLASH_BUFFERREAD	0x3E	Flash RAM- Puffer auslesen	Siehe unten	Daten der Flash-Seite
FLASH_STATE	0x39	Flash Status lesen	<flashstatus>	Flashstatus
PLAY – TEST Kommandos	0xC0	Audiodatei abspielen	Siehe unten	ACK
SETTIME	0xC8	RTC setzen	Siehe unten	ACK
ERASE_LOG_DATA	0xCE	Log Daten löschen		ACK
READ_LOG_DATA	0xCD	Log Daten auslesen		Log Daten
SET_SER_NR	0xC5	Seriennummer setzen		ACK
SETRTCKORR	0xCC	RTC Kalibrierwerte senden		ACK
READ_RTC_CAL	0x36	RTC Kalibrierwerte lesen		RTC Kalibrierwerte
ENTER_BOOT	0x35	In Bootloader Betrieb wechseln		ACK

2.1 Audio Datenspeicherverwaltung in aXbo

Mit dem Befehl FLASH_STATE kann der Status und damit die Speichergröße des internen Datenspeichers (Flash) ausgelesen werden. Die Bits sind dabei folgendermaßen zu interpretieren:

Wert <flashstatus>	Anzahl Seiten	Seitengröße	Speichergröße
[binär]		[Bytes]	[MB]
0bxx100xxx *)	4096	264	1
0bxx101xxx	4096	528	2

*) Standardbestückung aXbo

Weiters werden die einzelnen Audiodateien im aXbo anhand einer Tabelle organisiert. Diese Tabelle belegt immer die komplette erste Flashseite des Datenspeichers. Je Datei werden 16 Bytes als Dateiheader verwendet, maximal 16 Dateien sind erlaubt. Die Position des Datenheaders innerhalb dieser Tabelle (0 .. 15) ergibt den Dateiindex.

Der Dateiheader ist dabei folgendermaßen aufgebaut:

Byte	Beschreibung
0 – Bit 7	Datei gültig (Bit = 0), sonst ungültig
0 – Bit 6, 5, 4	Audiocodierung (nur Bit 5 = 1 entspricht µLaw, Standardcodierung im aXbo)
0 – Bit 3, 2, 1	Abtastrate (nur Bit 1 = 1 entspricht 11025Hz, standard im aXbo; 0 entspricht 8000Hz)
0 – Bit 0	Mono (Bit = 0, standard), sonst Stereo (nicht unterstützt)

1, 2	Startseite Audiodaten *)
3, 4	Endseite Audiodaten *)
5	Reserve (= 0)
6 .. 15	Dateiname in ASCII- Zeichen

*) Das niedrigere Byte entspricht dem Low- Byte

Zwischen Start- und Endseite befinden sich die eigentlichen Audiodaten. Die letzte verwendete Seite im Datenspeicher muss für µLaw- Codierung mit dem Wert 255 aufgefüllt werden. **Für Audiodaten stehen die Seiten 0x001 bis 0xDFF zur Verfügung.**

Mithilfe dieser Informationen kann ein Abbild des Datenspeichers ausgelesen werden bzw. neue Audiodateien in den Datenspeicher übertragen werden.

Die Audiodaten selber müssen dabei 8-Bit µLaw codiert sein, wozu verschiedene Tools/Codecs verwendet werden können. Unter Microsoft Windows® ist ein Audiodatei im WAV- Format gespeichert. Hier stehen die einzelnen Formateinstellungen im Dateiheader, nach dem Schlüsselwort „data“ steht die Anzahl der Audiodatenbytes (4 Byte lang), direkt danach folgen die eigentlichen Audiodaten, welche in den Datenspeicher des aXbo übertragen werden müssen.

2.2 Kommunikationsablauf

2.2.1 Kommunikationsbeginn

Ein Datenkommunikation wird immer mit dem Befehl DUMMY (0xCB) begonnen, Togglebit = 0. Bei korrektem Empfang wird der Befehl in jedem Fall mit ACK bestätigt und die Togglebits zwischen Master und Slave sind synchronisiert.

Nun können beliebige weitere Befehle ausgeführt werden.

Für eine stabile Kommunikation muss der Master Wiederholungen von Protokollen durchführen, falls aXbo mit NAK antwortet, und erst nach mehreren gescheiterten Versuchen die Kommunikation abbrechen.

2.2.2 Lesen einer Flashseite

Mit dem Befehl FLASH_PAGETOBUFFER muss zunächst die gewünschte Seite in den RAM-Puffer übertragen werden. Dazu wird als Parameter die Flashseite angegeben, bestehend aus 2 Byte, das 1. Byte ist das Hi- Byte.
Nach einer Wartezeit von einer Millisekunde kann mit dem Befehl FLASH_BUFFERREAD die Seite ausgelesen werden. Hier sind 3 Byte Parameter zu übergeben, die ersten beiden Bytes spezifizieren die Byteadresse innerhalb des Puffers, ab welcher gelesen werden soll (Hi- Byte zuerst), dahinter folgt die Anzahl der zu übertragenden Bytes (bei aXbo max. 66).

2.2.3 Schreiben einer Flashseite

Mit dem Befehl FLASH_BUFFERWRITE werden Daten zunächst in einen von 2 RAM- Puffern geschrieben. Hier spezifiziert der erste Parameter bestehend aus 2 Bytes die Byteadresse innerhalb des Puffers, ab welcher geschrieben werden soll (Hi- Byte zuerst), dahinter folgt die Anzahl der zu übertragenden Bytes (bei aXbo max. 66). Dann folgen max. 66 Datenbytes. Im höchstwertigen Bit der 16-Bit Byteadresse (Bit 15) kann der gewünschte RAM- Puffer gewählt werden.
Ist der RAM- Puffer vollständig geschrieben, kann er mit dem Befehl FLASH_BUFFERTOPAGE ins Flash übertragen werden. Dazu wird als Parameter die Flashseite angegeben, bestehend aus 2 Byte, das 1. Byte ist das Hi- Byte.

Achtung:
Beim Schreiben mehrerer Flashseiten hintereinander muss der verwendete RAM- Puffer immer abgewechselt werden, damit kann ein RAM- Puffer ins Flash übertragen werden, während der andere mit neuen Daten beschrieben wird.
Wird eine Flashseite nur teilweise beschrieben (Dateiheader), so muss die Seite zunächst vom Flash in den RAM- Puffer übertragen werden und nach einer Wartezeit von einer Millisekunde kann der Schreibvorgang durchgeführt werden.