

# FALCOM A2-3

## PROGRAMMING MANUAL



# FALCOM A2-3

## PROGRAMMING MANUAL

<b>1. General Description .....</b>	<b>3</b>
Summary schematics A2-3 .....	3
Detailed startup information .....	4
Downloading EXE files .....	5
Upgrading previous monitor versions .....	5
MON186 commands .....	6
<b>2. Programming guide .....</b>	<b>11</b>
Serial support functions .....	11
Environment support functions .....	13
Time and Date support functions .....	14
File and AUX port functions.....	15
<b>3. MON186 system services .....</b>	<b>16</b>
Serial support functions .....	16
Environment support functions .....	17
Time and Date support functions .....	18
Memory management functions.....	19
Process management functions .....	20
Console character input and output functions .....	20
File functions.....	22
Auxiliary io functions.....	22
Miscellaneous functions .....	23
<b>4. HARDWARE SUPPORT .....</b>	<b>24</b>
A1-3 hardware settings.....	24
A2-3 hardware settings.....	24
<b>5. Debug interface .....</b>	<b>27</b>
<b>6. Technical data .....</b>	<b>29</b>
<b>7. General hints .....</b>	<b>31</b>

# 1. General Description

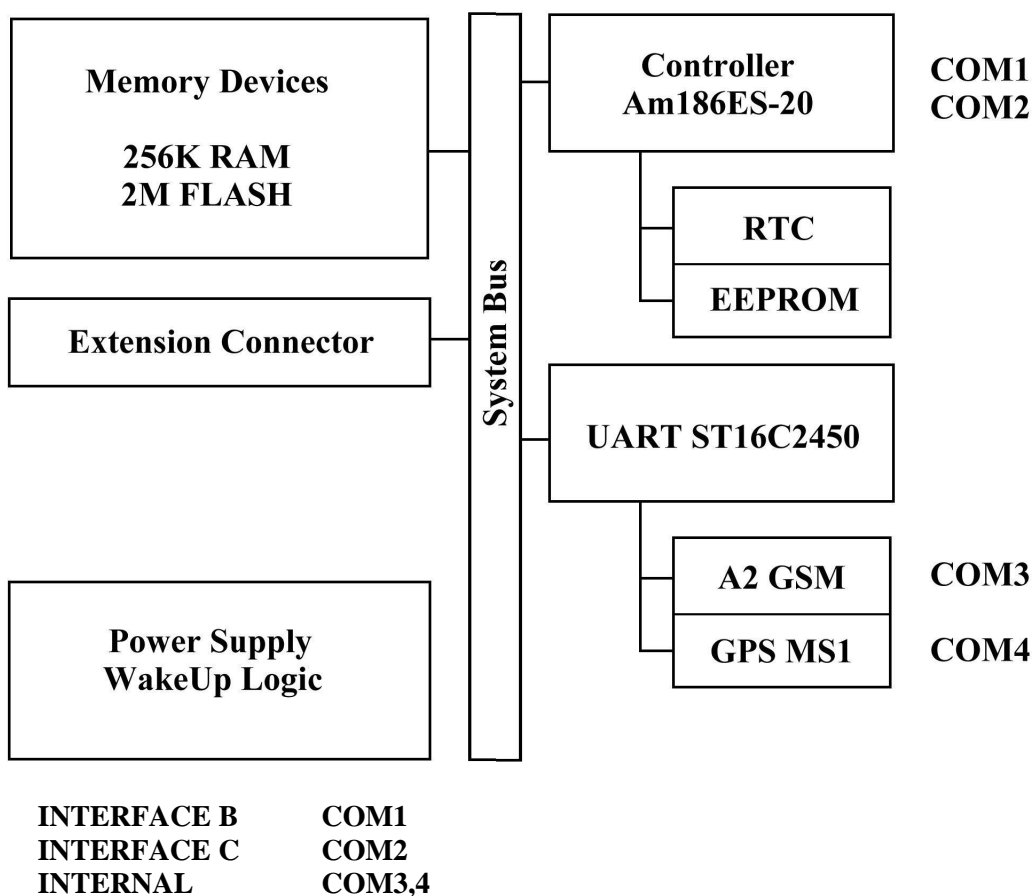
MON186 is the operating system for FALCOM A2-3 with the Am186ES controller.

MON186 is a basic monitor. It supports download of executable images to ROM or RAM, and rudimentary debugging. For developers just getting started, however, MON186 running on an FALCOM A2-3 modem provides a powerful tool to allow quick prototyping and benchmarking of simple algorithms, before a major investment is made in x86 development tools. Its minimal DOS emulator allows the developer to download and run small .EXE files which were developed and tested using standard compilers on a PC running DOS.

**NOTE! THIS DESCRIPTION APPLIES TO BOARDS OPERATING AT FACTORY DEFAULT SETTINGS. SEE "DETAILED STARTUP INFORMATION" BELOW IF THIS PROCEDURE DOES NOT WORK FOR YOU.**

## Summary schematics A2-3

For a quick overview please have a look on the scematic of the A2-3. Detailed information You will find in the chapter 6 „Technical Data“.



## Detailed startup information

Set up your PC's terminal program for 9600 Baud, at 8 bits per character, no parity and one stop bit. Set the terminal program's flow control to hardware flowcontrol. Connect the supplied serial cable from the PC to the FALCOM A2-3.

When the FALCOM A2-3 is reset, the 2 LEDs will go on. This first pattern will last for four seconds at which time MON186 will start a default modem application or display its sign on screen to the terminal and updating the LED display. At this point, you can press '?' followed by <ENTER> for MON186's help screen. When power is supplied, the initial LED pattern indicates that MON186 is waiting for an '@' character to be received from the terminal. If it receives an '@', it will automatically adjust to the baud rate of the '@', and display the MON186 welcome message and prompt. If it receives any character other than an '@' it will restart the terminal check and let the user try again to press an '@'.

If the user does not press an '@' during the initial LED pattern (nominally four seconds), MON186's next action depends on whether the user has installed a startup program in the ROM or not. If the user has used the 'W' command to store a DOS EXE program in the flash and the set "autorun" variable to mark it for running at startup time, then that DOS program will be executed. Otherwise, MON186 will display the welcome message and prompt, but must assume the baud rate. If the baud rate does not match that one of the terminal, the user will see nothing or garbled characters. (See the „Downloading EXE files" sections for information about installing user programs. )

At the factory, the baud rate is set to 9600 and the setting is 8N1. You can change this default by setting the COM "baudrate" variable on a common value.

The automatic baud rate detection is very useful in the following circumstances:

- If a user program is installed, but the user wishes to invoke the monitor instead.
- If the programmed baud rate does not match the terminal baud rate.
- If the programmed CPU speed does not match the actual CPU speed.  
(The bit clock is divided down from the CPU clock.)
- If the user doesn't want to wait 4 seconds for the monitor to boot.

The automatic baud rate detection is designed to detect baud rates from 1200 to 115200, but how well it works depends on the CPU type and speed. The algorithm may also fail at higher baud rates if you run the CPU at slower frequencies than the default 18.432 MHz.

MON186 supports downloading of Intel extended hex files into RAM or ROM. The hex file should contain type 2 extended address records, which specify the load address in the 1MB address range and the last record in the file should be a type 1 EOF record.

A file which is being downloaded to RAM for execution should be located between 410h and the start of the monitor data at the end of the RAM, and a file which is being downloaded to ROM for execution should be located between the start of the ROM and F0000h. The monitor 'I' command will show the size and location of the free RAM, and information about the size and location of the flash ROM.

It is impermissible for the file to have some sections download to RAM and others download to ROM, because MON186 relocates itself to some RAM locations while running. MON186 will report a range error on the download of such a file.

If you are downloading into ROM, you should first make sure the target download area is empty by using the 'X' command to erase the flash sectors. Unless you are storing multiple programs into flash, the easiest way to do this is to use 'XZ' to erase all the application sectors.

There is no specific command to download hex files. Simply start transferring with your terminal program in "ASCII" or "raw ASCII" mode. MON186 will echo the first record as it receives it, but when it parses it and determines that it is a hex file record, it will switch into a file transfer mode. The type 1 EOF record at the end of the file will switch back to command mode.

If an error is encountered during the download, an error message will be printed, and MON186 will stay in download mode until it receives an Escape character (1Bh), at which time it will print a more detailed error message and then return to command mode.

## Downloading EXE files

MON186 can download and run DOS executable files, enabling customers to use affordable, readily available, and familiar PC-based compilers and assemblers to develop initial test and benchmarking code. MON186 provides a minimal subset of DOS int 21h functionality, which is fully described in the section, "MON186 system services" chapter 3. Most compilers are capable of generating EXE files which work within this environment, as long as the user does not use library functions which require file-based I/O.

Unlike some prior versions, MON186 V3.36 does not support direct downloading of EXE files. Instead, it supports AMD LPD extensions to the Intel hex file format, and a supplied conversion program will convert EXE files into this extended hex file format. There are several reasons for this change:

- (1) Unlike hex files, exe files do not have error checking
- (2) Some terminal programs, e.g. HyperTerm which comes with Windows 95, will not transmit binary data unchanged.
- (3) The added overhead of transferring a hex file is mitigated by the fact that MON186 allows baud rates up to 115200.
- (4) The relocatable hex file can be stored to Flash (using the 'W' command) and later moved to RAM and executed (using the 'L' command).

To convert your EXE file into a HEX file, use the MAKEHEX utility supplied on this archive in the TOOL subdirectory. For example, to convert FOOBAR.EXE into FOOBAR.HEX, simply type MAKEHEX FOOBAR (assuming MAKEHEX.EXE is in your path).

Once you have converted your EXE file, simply download it to MON186 as described in the previous section. Once it is downloaded, you can set parameters for the program (if it expects a command line) with the 'N' command, and then start execution with the 'G' command.

Alternatively, use the 'W' command before you start downloading the file, to program it into flash. Since flash is non-volatile, the program can then be run multiple times, even after power has been cycled.

## Upgrading previous monitor versions

- (1) Use the 'XZ' command to erase all application flash sectors.
- (2) Download A2MON3xx.HEX, the upgrade file, to the board. It is not necessary to type any command to do this, the new MON186 automatically recognizes a file download when it sees the colon which starts the file.
- (3) Use the 'G' command to go to the new monitor, which is running out of

user flash ROM space. This will automatically go to the correct address.

- (4) Press '@' to establish communication with the new monitor. You are now running out of the application ROM based copy of the monitor.
- (5) Type 'Z' <enter> to initiate the upgrade. You will be asked if this is really what you want to do. Answer 'Y' to perform the upgrade, but do not do this if your power is not stable, or if little children are near the On/Off button. If the upgrade is aborted before it finishes, you may need to send your board back to factory to have the flash reprogrammed.
- (6) Your monitor is now upgraded, but you are still running out of the application ROM copy of the monitor. To run out of the new boot copy of the monitor, either switch on/off power, or type "G FFFF0" to go to the reset vector, then press '@' within 4 seconds to establish communication with the boot copy of the monitor.
- (7) You can now use the 'XZ' command to remove the application copy of the monitor, and then download any desired hex file to application ROM.

## MON186 commands

The first step in understanding how to use MON186 commands is to understand the command parameters. Different commands take different parameters, but these parameters are very commonly used:

BYTE -- 1 or 2 hexadecimal digits

WORD -- 1-4 hexadecimal digits

DECIMAL -- 1-9 decimal digits

ADDRESS -- An address may be entered in typical x86 segment:offset format, e.g. F800:0 to refer to the base of the monitor, or a LINEAR address may be entered as 5 hex digits, e.g. F8000. If the linear address approach is used, MON186 treats the first 4 digits as the segment, and the last digit as the offset. Most commands which do not alter memory also support SHORT addresses. A short address is where only the offset is specified (between 1 and 4 hex digits). The current value of the DS register is implicitly used for the segment.

Commands which alter memory require a full address.

RANGE -- An address range may be specified in two different ways, either as <address> <space> <address>, where the address of the start of the range and the address immediately after the end of the range are specified, or as <address> L <length>, where the address of the start of the range and the length of the range are explicitly specified. The following commands are identical, and dump 1024 bytes starting at 16K:

```
D 400:0 400:400
D 0:4000 400:400
D 04000 L 400
d040001400
```

As the last command shows, spaces only matter where the parser would have trouble distinguishing the end of one number from the start of the next one, and all commands may be entered in upper or lower case.

**LIST** -- A list is a collection of bytes. Each byte may be specified with one or two hex digits, with the bytes separated by spaces, and ASCII data may be specified in single or double quotes. The following command will place an ASCII string, complete with carriage return and two line feeds, at 16K:

```
04000 "This is a quoted string" 0D A,0A
```

Note that (other than the mandatory 5 digits for a linear address) numbers do not require leading zeros. Also note that commas are optional. They may be used instead of or in conjunction with spaces.

Angle brackets <> indicate required parameters.

Square brackets [] indicate optional parameters.

Vertical bar | indicates the user should choose one of the parameters

<Break> When MON186 receives an RS232 break (usually invoked by pressing Alt-B or Ctrl-Break on the terminal emulator) it will break into the debugger. This is useful in some cases when your application appears 'hung' -- you can find out where it is executing. Note, however, that <Break> can also be used to debug MON186 itself, and you should be careful how many times you press it without pressing "G" to continue program execution. Too many breaks will cause a stack overflow within MON186 itself.

**B** <address> Sets a breakpoint by saving the value at a location, and then inserting an int 3 instruction (CCh) at that location. Only one breakpoint is active at a time -- setting one removes previous breakpoints. Breakpoints may only be set in RAM, not in ROM. When the int 3 at the breakpoint is executed, the code at the breakpoint is automatically restored. At this point, you may set another breakpoint if you desire, and use the G or T commands to continue execution.

**C** <range> <address> Compares two memory ranges. Each differing byte will be displayed on a single line as:  
    <address in range> <byte in range> <comparison byte> <comparison address>

**D**[WA] [range] Dumps a memory range, in hexadecimal bytes/words and/or ASCII. If the range is not specified, it will dump 128 bytes starting where the most recent dump command finished.

**E** <address> [list] Enter memory. If the list (at least one byte) is specified, the entire list will be stored in memory at <address>. If no list is specified, the command will prompt for entry of a list of bytes at incrementing addresses. When all data has been entered, respond to the prompt with a single dot '.' on a line, or with the escape key.

**F** <range> <list> Fills a memory range with a list of bytes. The entire range is filled, and the list is replicated as many times as it takes to fill it. The size of the list does not need to fit evenly in the range: the last copy of the list is truncated to fit.

**G** [=address]

"Go", e.g. start execution. If an address is given, it will be stored in CS:IP before execution starts. The equal sign is permitted for compatibility with DOS DEBUG.

I[W[word]] The "Input" command by itself will show information about the system. 'I' followed by a word will input from a byte-wide port and display the results, and 'IW' followed by a word will input from a word-wide port and display the results.

J The J command causes the automatic baud rate detection to be invoked. Once you have entered this command, you may change the terminal's baud rate. On an Am186ES processor, you may even connect to the alternate serial port. Once you are set up properly, simply press "a" to reestablish connection with the monitor. Note that automatic baud rate detection may not be reliable at baudrates which are high relative to the CPU frequency and bus width. At a CPU frequency of 18.432MHz, the Am186ES parts can reliably detect 115200 kBaud.

L[G] [decimal] The "Load" command loads a previously stored EXE file from flash to RAM. If no parameters are given, a list of currently stored programs is displayed. If a decimal number is given, the corresponding program is copied from flash to RAM. Programs are loaded to flash using the W command, and may be made bootable with the "AutoRun" setting. The 'LG' command is equivalent to the 'L' command immediately followed by a 'G' command, e.g. load and run the program.

M <range> <address> Moves a block of memory from one address to another. Overlapping blocks are handled correctly. The following command sequence shows how the monitor can be executed out of RAM:

```
M F8000 L 7000 00400 -- moves monitor to base of RAM
G 00400 -- starts execution
I -- shows new monitor CS and free memory
```

N <arguments> In DOS DEBUG, this command names the COM or EXE file to load or save, and also gives command line arguments. MON186 has no knowledge of the file name, so only requires command line arguments (if needed by the program). We recommend you design your test program so that it does not rely on command line arguments, as it is easy to forget to use the 'N' command.

O[W] <word> <byte>|<word> Outputs the second parameter (byte or word) to the port given in the first parameter. Use 'OW' for word-wide outputs, 'O' for byte-wide outputs.

P[ABC] [VariableName DecimalValue|String Value] Sets or shows Permanent Environment Parameters. The monitor stores these values in a 32 kBit serial eeprom. Use 'PC' to clear all environment parameter at once. Use 'P VariableName' to clear a specific setting. For its own configuration MON186 uses the following variables:

BOOT = cpuspeed,autorun,feature

cpuspeed -- This defines the speed of the CPU to the monitor. This is required for correct default baud rate set up and to correct internal timer tick correctly, which is used by benchmark programs and also governs the speed of the LED patterns.



autorun -- When this is non-zero, it selects which EXE program to load from the flash and run at boot time. A value between 8000 - F000 starts directly a program downloaded to this address in the flash. A value greater than 0 starts a EXE program loaded to the flash with the 'W' command.

feature -- This defines a special string with following meaning. When the character 'L' is defined, the monitor will use the LEDs to show current status. When this is not set, the monitor will not change the LEDs. When the character 'B' is defined, the monitor enter itself after receiving a break on the serial port.

COM1 = baudrate[,mode,muxvalue,handshake][,buffer size]  
COM2 = baudrate[,mode,muxvalue,handshake][,buffer size]

baudrate -- Sets the default baudrate for the serial port ( 1200 - 115200 ). The detection of the baudrate at startup overwrites this setting und the monitor uses the detected value instead.

mode -- Sets the default line setting for the serial port ( 7E1,7O1,8N1,8E1,8O1 ).

muxvalue – Dummy setting on the A2-3. On the device A1-3 sets the default value for the multiplexer of the serial port. The important settings are 0 ( means interface DB9 ) and 3 ( means interface RJ45 ). Please note that the 2 com port must have a different muxvalue to prevent a loss of communication to the MON186.

handshake - Sets the used flow control of serial operation. That means with 'X' the monitor uses XonXoff software flow control and with 'H' the monitor uses RtsCts hardware handshake.

buffer size - Sets the size of the buffer for the serial port. The default value is 256 Byte and can be set from 256 .. 8192 Byte.

R [RegisterName | ("F" FlagName)]

The "Register" command with no parameters will display the current state of all registers and flags. 'R' can also be used to set the value of any register or flag bit:

To examine a register: R AX

This will print the current value of the AX register and prompt you for a new value.

To change a register without examining it: R AX 5000

This will change the value of AX to 5000h.

To examine the flags: R F

This will print the current flag values, and prompt you for a two letter code to change them. Flag names are the same as DOS debug uses. Don't worry if you get the flag name wrong, MON186 will show you the names it expects.

To change a flag without examining it: R F DN

This will set the direction flag, so the direction is now "down".

NOTE: As discussed previously, in most situations, spaces are optional. These commands could be entered as RAX, RAX5000, RF, and RFDN, respectively.

S <range> <list>

„Search“ a given range for a list of bytes. The starting address of each occurrence of the list within the range is displayed. There will be no display if the list is not found within the range.

T [=address] [word]

This command uses the x86 trap flag to trace execution. Unlike breakpoints, traces may be performed in ROM as well as RAM. An optional starting address may be used to set CS:IP before the trace starts, and an optional number of steps to trace may be entered as well. The default is 1 step.

W [file name]

The “Write“ command initiates a download of a relocatable hex file (generated by running the host program MAKEHEX on a DOS executable) to the flash. The file name is given so that the program can be identified later if multiple programs are stored in the flash. Programs are stored starting at the lowest address of the flash. Use the ‘L‘ command later to move a program into RAM for execution, or use the "AutoRun" setting to cause the monitor to load and run a program at boot time.

X <sector number> | Z "eXterminate"

This command will erase one of the sectors in the application area of the flash ROM, or, if ‘XZ‘ is given, will erase all of them. The ‘I‘ command can be used to retrieve information about the sectoring of the flash part. Use 0 to refer to the first sector, 1 to the next one, etc.

U [hh.mm.ss][ dd.mm.yyyy]

The “U“ command sets the current system time and date to the real time clock or shows the current value.

Z The “Z“ command upgrades the boot monitor. It may be issued under two circumstances, either from a monitor which is running at the upgrade location (normally F0000h, but depends on flash type), to upgrade the boot monitor in the same flash part, or from a monitor which is running at the boot monitor location (F8000h) to replace a dead monitor in a different flash part (on boards which support a CS switch from one flash to another).

## 2. Programming guide

We choose Visual C++ V1.52 as programming environment for the FALCOM A2-3. That package includes all necessary tools to build application for the FALCOM A2-3. The standard „C“ functions are contained in the standard libraries of Visual C++. The different programming environment for the hardware related parts on the FALCOM A2-3 included in a additional library. That library „LIBA1.LIB“ contains hardware related serial, date, time and environ functions and the syntax of those additional functions listed below. For an overview of the Visual C++ standard function please look in the online helps or try to refer to it in a programming training course.

### Serial support functions

The functions ComPutch(), ComGetch(), ComRead(), ComWrite(), ComString() can be used to communicate with those serial devices. The functions ComGetConfig(), ComSetConfig(), ComLine() should be used for reading the current state of the com port or changing the com port configuration.

Parameter definitions:

```
#define PORT_COM1          0
#define PORT_COM2          1
#define PORT_COM3          2
#define PORT_COM4          3

#define LINE_STS_MASK      0xFF00
#define LINE_ERROR         0x8000
#define LINE_RECV_BREAK   0x2000
#define LINE_TRNS_BLOCKED 0x1000
#define LINE_RECV_FRAME    0x0800
#define LINE_RECV_PARITY   0x0400
#define LINE_RECV_OVER     0x0200
#define LINE_RECV_READY    0x0100

#define LINE_SET           0x8000
#define LINE_CLEAR         0x0000
#define LINE_MASK          0x00FF
#define LINE_FLUSH         0x4000
#define LINE_BREAK         0x2000
#define LINE_UPDATE        0x1000
#define LINE_RESET         0x0800
#define LINE_DCD           0x0080
#define LINE_DSR           0x0020
#define LINE_CTS           0x0010
#define LINE_DTR           0x0008
#define LINE_RTS           0x0004
#define LINE_RI            0x0002
#define LINE_DEVICE        0x0001

#define MODE_BIT_MASK      0x0003
#define MODE_BIT_5         0x0000
#define MODE_BIT_6         0x0001
#define MODE_BIT_7         0x0002
#define MODE_BIT_8         0x0003
```

```

#define MODE_STOP_MASK      0x0004
#define MODE_STOP_1        0x0000
#define MODE_STOP_2        0x0004

#define MODE_PAR_MASK      0x0018
#define MODE_PAR_NONE      0x0000
#define MODE_PAR_ODD       0x0008
#define MODE_PAR_EVEN      0x0018

#define MODE_BAUD_MASK     0x00E0
#define MODE_BAUD_1200     0x0000
#define MODE_BAUD_2400     0x0020
#define MODE_BAUD_4800     0x0040
#define MODE_BAUD_9600     0x0060
#define MODE_BAUD_19200    0x0080
#define MODE_BAUD_38400    0x00A0
#define MODE_BAUD_57600    0x00C0
#define MODE_BAUD_115200   0x00E0

#define MODE_FLOW_MASK     0x0300
#define MODE_FLOW_H        0x0100
#define MODE_FLOW_X        0x0200

#define MODE_MUX_MASK      0x0C00
#define MODE_MUX_DB9       0x0000
#define MODE_MUX_GPS       0x0400
#define MODE_MUX_GSM       0x0800
#define MODE_MUX_WS        0x0C00

```

Get Parameter of com port:

**WORD ComGetConfig( BYTE com,lpWORD config,lpWORD time );**

Parameter	BYTE	com	ComPort
	lpWORD	config	ComConfig
	lpWORD	time	Timeout
Result	WORD	line	LineState

Set Parameter of com port:

**WORD ComSetConfig( BYTE com,WORD config,WORD time );**

Parameter	BYTE	com	ComPort
	lpWORD	config	ComConfig
	lpWORD	timeout	Timeout
Result	WORD	line	LineState

Get a character from com port:

**WORD ComGetch( BYTE com );**

Parameter	BYTE	com	ComPort
Result	WORD	line	LineState (HighByte) and InputData (LowByte)

Put a character to com port:

**WORD ComPutch( BYTE com, BYTE xch );**

Parameter	BYTE	com	ComPort
	BYTE	xch	OutputData
Result	WORD	line	LineState

Read data from com port:

**WORD ComRead( BYTE com, lpBYTE p, WORD num );**

Parameter	BYTE	com	ComPort
	lpBYTE	p	Buffer
	WORD	num	Count
Result	WORD	line	LineState

Write data to com port:

**WORD ComWrite( BYTE com, lpBYTE p, WORD num );**

Parameter	BYTE	com	ComPort
	lpBYTE	p	Buffer
	WORD	num	Count
Result	WORD	line	LineState

Put a string to com port:

**WORD ComString( BYTE com, lpBYTE p );**

Parameter	BYTE	com	ComPort
	lpBYTE	p	Buffer
Result	WORD	line	LineState

Set the state of the com port:

**WORD ComLine( BYTE com, WORD set );**

Parameter	BYTE	com	ComPort
	WORD	config	ComConfig
	WORD	set	LineState
Result	WORD	line	LineState

## Environment support functions

The functions SetEnviron(), GetEnviron() and EnvironString() can be used to communicate with a serial eeprom device. To handle different data types these functions use a type Parameter EnvType wich can be ENV\_CLEAR (delete a entry), ENV\_VALUE (integer data), ENV\_STRING (string arrays) and ENV\_DATA (binary arrays).

Parameter definitions:

```
enum {  
    ENV_CLEAR, ENV_STRING, ENV_DATA, ENV_VALUE  
} EnvType;
```

Write a environ entry:

**INT SetEnviron( WORD typ,lpBYTE entry,lpBYTE env,WORD len );**

Parameter	WORD	typ	EnvType
	lpBYTE	entry	EnvName
	lpBYTE	env	EnvData
	WORD	len	Number of data to write
Result	INT	error	Operation succeed

Read a environ entry:

**INT GetEnviron( WORD typ,lpBYTE entry,lpBYTE env,WORD len );**

Parameter	WORD	typ	EnvType
	lpBYTE	entry	EnvName
	lpBYTE	env	EnvData
	WORD	len	Maximum number of data to read
Result	INT	error	Operation succeed

Read or write a string environ entry:

**INT EnvironString( BOOL write,lpBYTE entry,lpBYTE env,WORD len );**

Parameter	BOOL	write	TRUE to write, FALSE to read
	lpBYTE	entry	EnvName
	lpBYTE	env	EnvData
	WORD	len	Maximum number of data to read
Result	INT	error	Operation succeed

## Time and Date support functions

The functions GetTime(), SetTime() can be used to communicate with the real time clock.

Parameter definitions:

```
typedef struct Time {
    BYTE Hundredths;
    BYTE Seconds;
    BYTE Minutes;
    BYTE Hour;
    BYTE Day;
    BYTE Month;
    WORD Year;
    BYTE DayOfWeek;
    DWORD TotalTime;
} RtcTime, __far *lpRtcTime;
```

Get current system time:

**DWORD GetTime( lpRtcTime t );**

Parameter	lpRtcTime	t	SystemTime
Result	DWORD	ticks	ticks of the day, value in hundredth seconds.

Set current system time:

**void SetTime( lpRtcTime t );**

Parameter	lpRtcTime t	SystemTime
Result	nothing	

File and AUX port functions

```
#define HANDLE_STDIN    0x00    /* file handles */
#define HANDLE_STDOUT  0x01
#define HANDLE_STDERR  0x02
#define HANDLE_AUX     0x03

#define AUX_IGNITION   0x0080   /* input values */
#define AUX_POWER      0x0040
#define AUX_RESET      0x0020
#define AUX_SIMCHNG    0x0010
#define AUX_LINE4      0x0008   /* in & output values */
#define AUX_LINE3      0x0004
#define AUX_LINE2      0x0002
#define AUX_LINE1      0x0001
```

Read and write to the AUX port:

**void SetAux( WORD io );**  
**WORD GetAux( void );**

File support functions:

**void DosWrite( WORD hnd,lpBYTE data,WORD len );**  
**void DosRead( WORD hnd,lpBYTE data,WORD len );**

There are some differences between the AUX and the file functions for the HANDLE\_AUX value. A file operation handles read or write patterns to a display on the LEDs of the device (LED line of the A1-3 or the 2 LED's on the A2-3). A SetAux() or GetAux() function reads or writes to an additional io port. (ignition line, power fail comparator, reset and simcard switch and general io lines).

### 3. MON186 system services

#### Serial support functions

The A2-3 handles serial ports COM1-COM4 for the connection with different serial io devices. The serial lines connected are shown in the following diagram:

COM1	Serial Interface on the DB15
COM2	Serial Interface on the RJ45
COM3	Serial Interface gsm modem
COM4	Serial Interface internal gps receiver or debug port

The functions ComPutch(), ComGetch(), ComRead(), ComWrite(), ComString() can be used to communicate with that serial devices. The functions ComGetConfig(), ComSetConfig(), ComLine() should be used for reading the current state of the com port or changing the com port configuration. The MON186 support the COM service 00h - Init com port, 01h - Get com port state, 02h - Get character from com port, 03h - Put character to com port, 04h - Get string from com port, 05h - Put string to com port and 06h - Init com port with a string configuration.

#### INT22 service 00h: Init com port

Parameter	AH = 00h	COM service 00h
	AL = ComPort	handle of com port
	CX = ComConfig	new configuration setting
	DX = Timeout	new timeout setting
Result	AX = LineState	current state of com port

#### INT22 service 01h: Get com port state

Parameter	AH = 01h	COM service 01h
	AL = ComPort	handle of com port
	CX = LineState	set line state of com port
Result	AX = LineState	current state of com port
	CX = ComConfig	configuration setting
	DX = Timeout	timeout setting

#### INT22 service 02h: Get character from com port

Parameter	AH = 02h	COM service 02h
	AL = ComPort	number of com port
Result	AX = LineState	current state of com port
	CL = InputData	char read from com port

#### INT22 service 03h: Put character to com port

Parameter	AH = 03h	COM service 03h
	AL = ComPort	handle of com port
	CL = OutputData	character writes to com port
Result	AX = LineState	current state of com port



### **INT22 service 04h: Get string from com port**

Parameter	AH = 04h AL = ComPort ES:BX = Buffer CX = Count	COM service 04h handle of com port string buffer size of maximum characters to read
Result	AX = LineState CX = ReadCount	current state of com port size of characters read from com port

### **INT22 service 05h: Put string to com port**

Parameter	AH = 05h AL = ComPort ES:BX = Buffer CX = Count	COM service 05h handle of com port string buffer size of characters to write
Result	AX = LineState CX = WriteCount	current state of com port size of characters written to com port

### **INT22 service 06h: Init com port with string configuration**

Parameter	AH = 00h AL = ComPort ES:BX = ComConfig	COM service 00h handle of com port configuration string com port
Result	AX = LineState	current state of com port

## **Environment support functions**

On the A2-3 a nonvolatile memory for storage of settings, Parameters, low volume data, etc is used. This device is a serial eeprom with a capacity of 4096 Byte and with a guaranteed write cycles of one million. The functions SetEnviron(),GetEnviron() and EnvironString() can be used to communicate with that device. To handle that different data types these functions use a type Parameter EnvType wich can be ENV\_CLEAR (delete a entry), ENV\_VALUE (integer data), ENV\_STRING (string arrays) and ENV\_DATA (binary arrays). The other Parameter are the name and the data of an environ entry. The third function is used for an easy handling of ascii strings. You should note, that while a write operation to the device a preview entry with the same name will be overwritten. The Mon186 supports the DOS service 2Eh - Set environment and 2Fh - Get environment to read and write data to the environ memory.

### **INT21 service 2Eh: Set environment data**

Parameter	AH = 2Eh AL = EnvType  DS:DX = EnvName ES:BX = EnvData CX = Count	DOS service 2Eh type ENV_CLEAR means delete entry type ENV_VALUE means write integer data type ENV_DATA means write binary data type ENV_STRING means write ascii data environ entry name (max 63 chars) environ entry data (max free space of device) maximum size of data
Result	CF = 0  AX = Size	operation successful  size of written data

CF = 1                    operation failed  
 AX = ErrorCode

**INT21 service 2Fh:    Get environment data**

Parameter	AH = 2Fh AL = EnvType  DS:DX = EnvName ES:BX = EnvData CX = Count	DOS service 2Fh type ENV_VALUE means read integer data type ENV_DATA means read binary data type ENV_STRING means read ascii data environ entry name (max 63 chars) environ entry data (max free space of device) size of data
Result	CF = 0 AX = Size	operation successful size of read data
	CF = 1 AX = ErrorCode	operation failed

**Time and Date support functions**

On the A2-3 a real time clock and calendar device is used. The functions SetTime()and GetTime() can be used to communicate with that device. That real time clock is a low power device with a common CR1220 lithium backup battery with a typical life time of 2 years. The MON186 supports the DOS service 2Ah - Set date, 2Bh - Get date, 2Ch - Set time, 2Dh - Get time to read and write data to the real time device.

**INT21 service 2Ah:    Set real time clock date**

Parameter	AH = 2Ah CX = Year DH = Month DL = Day    day ( 1 .. 31 )	DOS service 2Ah year ( 1980 .. 2079 ) month ( 1 .. 12 )
Result	CF = 0 AL = 0	operation successful
	CF = 1 AL = ErrorCode	operation failed

**INT21 service 2Bh:    Get real time clock date**

Parameter	AH = 2Bh	DOS service 2Bh
Result	CF = 0 AL = 0 CX = Year DH = Month DL = Day BL = Weekday	operation successful  year ( 1980 .. 2079 ) month ( 1 .. 12 ) day ( 1 .. 31 ) day of week ( 0 .. 6 )
	CF = 1 AL = ErrorCode	operation failed

### **INT21 service 2Ch: Set real time clock time**

Parameter	AH = 2Ch	DOS service 2Ch
Result	CF = 0	operation successful
	AL = 0	
	DL = Msec	hundreds of second ( 0 .. 99 )
	DH = Sec	seconds ( 0 .. 59 )
	CL = Minutes	minutes ( 0 .. 59 )
	CH = Hour	hour ( 0 .. 23 )
	CF = 1	operation failed
	AL = ErrorCode	

### **INT21 service 2Dh: Get real time clock time**

Parameter	AH = 2Dh	DOS service 2Dh
	DL = Msec	hundreds of second ( 0 .. 99 )
	DH = Sec	seconds ( 0 .. 59 )
	CL = Minutes	minutes ( 0 .. 59 )
	CH = Hour	hour ( 0 .. 23 )
Result	CF = 0	operation successful
	AL = 0	
	CF = 1	operation failed
	AL = ErrorCode	

## **Memory management functions**

For handling with bigger memory junks in the global heap the standard functions `_fmalloc()`, `_ffree()` and `_frealloc()` should be used. Those functions are implemented through the standard DOS service memory functions listed below. The MON186 supports the DOS service 48h - Memory allocation, 49h Free allocated memory and 4Ah – Memory reallocation for a proper memory management.

### **INT21 service 48h: Memory allocation**

Parameter	AH = 48h	DOS service 48h
	BX = Size	block size in paragraph
Result	CF = 0	operation successful
ucced	AX = Segment	segment präfix
	CF = 1	operation failed
	BX = Size	maximum block size in paragraph
	AX = ErrorCode	

### **INT21 service 49h: Free allocated memory**

Parameter	AH = 49h ES = Segment	DOS service 49h segment präfix
Result succeed	CF = 0	operation successful
	CF = 1 AX = ErrorCode	operation failed

### **INT21 service 4Ah: Memory reallocation**

Parameter	AH = 4Ah ES = Segment BX = Size	DOS service 4Ah segment präfix block size in paragraph
Result succeed	CF = 0	operation successful
	AX = Segment	segment präfix
	CF = 1 BX = Size AX = ErrorCode	operation failed maximum block size in paragraph

## Process management functions

The MON186 support the DOS service 4Ch and 00h - Exit process for the realization of a process termination.

### **INT20: Process termination**

Parameter	nothing	old DOS termination service
-----------	---------	-----------------------------

### **INT21 service 00h: Process termination**

Parameter	AH = 00h	DOS service 00h
-----------	----------	-----------------

### **INT21 service 4Ch: Process termination**

Parameter	AH = 4Ch AL = ReturnCode	DOS service 4Ch dos return value
-----------	-----------------------------	-------------------------------------

## Console character input and output functions

The higher level io functions in the standard library are `putch()`, `getch()`, `printf()`, `scanf()`, etc. . These functions use standard dos calls to read and write to the console. By implementing those low level console functions you are able to use standard functions for input and output purposes. The MON186 supports the DOS service 01h - Character input with echo, 02h - Character output, 06h - Character raw input, 07h,08h - Character raw input, 09h - String output, 0Ah - String input, 0Bh - Console input state and 0Ch - Flush buffer and console input function.

**INT21 service 01h: Character input with echo**

Parameter AH = 01h DOS service 01h

Result AL = Input input character

**INT21 service 02h: Character output**

Parameter AH = 02h DOS service 02h  
DL = Output output character

**INT21 service 06h: Character raw input**

Parameter AH = 06h DOS service 06h  
DL = FFh read character  
DL = Output output character

Result ZF = 0 character in buffer  
AL = Input input character

ZF = 1 buffer empty

**INT21 service 07h: Character raw input**

Parameter AH = 07h DOS service 07h

Result AL = Input input character

**INT21 service 08h: Character raw input**

Parameter AH = 08h DOS service 08h

Result AL = Input input character

**INT21 service 09h: String output**

Parameter AH = 09h DOS service 09h  
DS:DX = Buffer output buffer

**INT21 service 0Ah: String input**

Parameter AH = 0Ah DOS service 0Ah  
DS:DX = Buffer input buffer

**INT21 service 0Bh: Console input state**

Parameter AH = 0Bh DOS service 0Bh

Result AL = State state of input console

**INT21 service 0Ch: Flush buffer and console input function**

Parameter AH = 0Ch DOS service 0Ch  
AL = InputFunction input function ( 01h,06h,07h,08h,0Ah )

## File functions

The MON186 support the DOS service 3fh - Read from file and 40h - Write to file for a minial file support with the file handle console and aux port.

HANDLE_STDIN	Redirect to console
HANDLE_STDOUT	Redirect to console
HANDLE_STDERR	Redirect to console
HANDLE_AUX	LED port handle

### INT21 service 3Fh: Read from file

Parameter	AH = 3Fh	DOS service 3Fh
	BX = Handle	file handle
	DS:DX = Buffer	data buffer
	CX = Count	size of data
Result	CF = 0	operation successful
succed	AX = Size	size of read data
	CF = 1	operation failed
	AX = ErrorCode	

### INT21 service 40h: Aux output state

Parameter	AH = 40h	DOS service 40h
	BX = Handle	file handle
	DS:DX = Buffer	data buffer
	CX = Count	size of data
Result	CF = 0	operation succed
	AX = Size	size of read data
	CF = 1	operation failed
	AX = ErrorCode	

## Auxiliary io functions

The MON186 supports the DOS service 03h - Aux input and 04h - Character output for that special aux port. Those functions can be use to read or to set following values:

#define AUX_IGNITION	0x0080
#define AUX_POWER	0x0040
#define AUX_RESET	0x0020
#define AUX_SIMCHNG	0x0010
#define AUX_LINE4	0x0008
#define AUX_LINE3	0x0004
#define AUX_LINE2	0x0002
#define AUX_LINE1	0x0001

**INT21 service 03h: Aux input state**

Parameter	AH = 03h	DOS service 03h
-----------	----------	-----------------

Result	AL = Input	state of aux port
--------	------------	-------------------

**INT21 service 04h: Aux output state**

Parameter	AH = 03h	DOS service 03h
	DL = Output	outport to aux port

**Miscellaneous functions**

At last, the MON186 supports some kind of utility functions DOS service 25h – Set an interrupt handler, 35h - Get an interrupt handler and 30h – Get system information.

**INT21 service 25h: Set interrupt handler**

Parameter	AH = 25h	DOS service 25h
	AL = Number	interrupt number
	DS:DX = Handler	interrupt handler

**INT21 service 35h: Get interrupt handler**

Parameter	AH = 35h	DOS service 35h
	AL = Number	interrupt number

Result	ES:BX = Handler	interrupt handler
--------	-----------------	-------------------

**INT21 service 30h: Get system information**

Parameter	AH = 30h	DOS service 30h
-----------	----------	-----------------

Result	AL = Version	dos version
	AH = Revision	
	CX = Device	device code
	DX = System	system version

## 4. HARDWARE SUPPORT

### A1-3 hardware settings

```
IOB186ES      0xff00          /* AM186ES register base */
PORT_LED      0x0000
PORT_WDI      0x0100

GPIO0         DCD line DB9          /* AM186ES ioport values */
GPIO1         DTR line DB9
GPIO2         DSR line DB9
GPIO3         DCD line modem
GPIO4         DTR line modem
GPIO5         DSR line modem
GPIO10        control mux0 ( TxD/RxD0 )
GPIO11        - " -
GPIO14        control mux1 ( TxD/RxD1 )
GPIO15        - " -
GPIO16        /CS output led
GPIO17        /WDI reset watchdog
GPIO18        RTS line DB9
GPIO19        CTS line DB9
GPIO20        RTS line modem
GPIO21        CTS line modem
GPIO24        Enable RS485
GPIO25        /IODIR connect modem to DB9
GPIO26        /GSMON on/off modem
GPIO30        SDA ( I2C data )
GPIO31        SCL ( I2C clock )

INT 08h       TIMER0              /* Interrupt routing */
INT 12h       TIMER1
INT 13h       TIMER2
INT 14h       COM1
INT 11h       COM2
```

### A2-3 hardware settings

```
IOB186ES      0xff00          /* AM186ES register base */
PORT_WDI      0x0100
UART_CSA      0x0500          /* UART channel A base */
UART_CSB      0x0600          /* UART channel B base */

GPIO0         DCD line DB9          /* AM186ES ioport values */
GPIO1         DTR line DB9
GPIO2         /CS uart channel a
GPIO3         /CS uart channel b
GPIO4         DSR line DB9
GPIO5         RI line DB9
GPIO10        tone signal output
GPIO11        io signal 1
GPIO12        io signal 2
```



```

GPIO13      io signal 3
GPIO14      io signal 4
GPIO15      flash address A20
GPIO16      flash address A20
GPIO17      /WDI reset watchdog
GPIO18      RTS line DB9
GPIO19      CTS line DB9
GPIO20      led green
GPIO21      led orange
GPIO24      Enable RS485
GPIO25      CaspOn device a2
GPIO26      SoftOn device a2
GPIO29      Update device a2
GPIO30      SDA ( I2C data )
GPIO31      SCL ( I2C clock )

CHA RTS .. RI  DB9 signals          /* UART channel A io values */

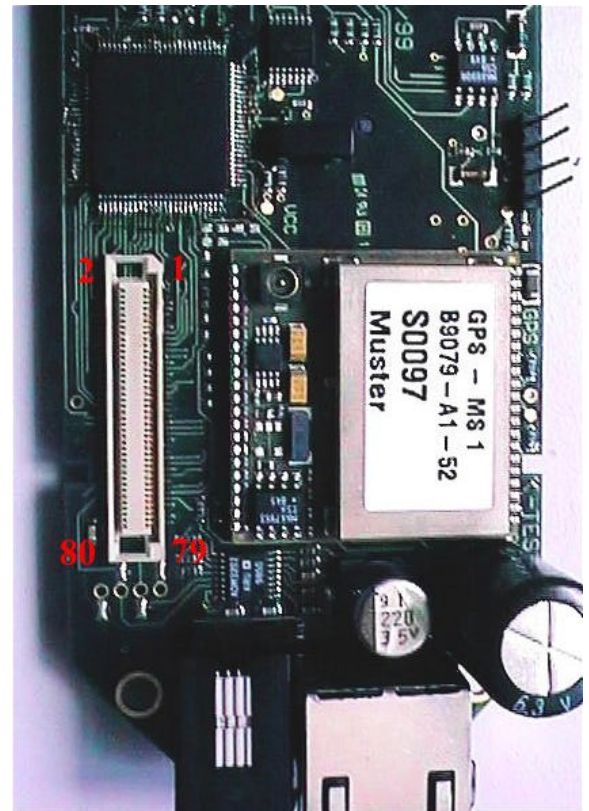
CHB RTS      Reset A2              /* UART channel B io values */
CHB DTR      Mute
CHB DSR      SIM button
CHB CTS      Reset button
CHB CD       Power fail
CHB RI       Ignition

INT 08h      TIMER0                /* Interrupt routing */
INT 12h      TIMER1
INT 13h      TIMER2
INT 0Ch      RTC
INT 14h      COM1
INT 11h      COM2
INT 0Dh      COM3
INT 0Fh      COM4

```

Table1: J2 bus expander for memory or other IO extensions ( Hirose FX8-80S-SV )

1	GND	2	GND
3	A1	4	D0
5	A2	6	D1
7	A3	8	D2
9	A4	10	D3
11	A5	12	D4
13	A6	14	D5
15	A7	16	D6
17	A8	18	D7
19	A9	20	D8
21	A10	22	D9
23	A11	24	D10
25	A12	26	D11
27	A13	28	D12
29	A14	30	D13
31	A15	32	D14
33	A16	34	D15
35	A17	36	/RES
37	A18	38	/UCS
39	A19	40	/LCS
41	A20	42	/WLB
43	A21	44	/WHB
45	/WR	46	HOLD
47	/RD	48	HLDA
49	18.432MHz	50	NMI
51	9.216MHz	52	WDI
53	ARDY	54	
55	SRDY	56	
57	SDA	58	
59	SCL	60	
61	MIC+	62	
63	MIC-	64	
65	SPK+	66	MUTE
67	SPK-	68	IGN
69	VBB	70	DBG TxD
71	VIN3V	72	DBG RxD
73	VCC5V	74	VCC5V
75	VCC3V	76	VCC3V
77	VCC3V	78	VCC3V
79	GND	80	GND



A description of the additional circuits are found on the manufactures links:

AM186ES [www.amd.com](http://www.amd.com)  
 ST16C2450 [www.exar.com](http://www.exar.com)  
 PCF8593 [www.philips.com](http://www.philips.com)  
 24LC32 [www.microchip.com](http://www.microchip.com)

## 5. Debug interface

For the A2-3 a development kit is available. That package includes a Visual C++ programming book and a training course and the Paradigm DEBUG/RT debug tool. By using this package you are ready to work with a powerful source debugging environment. In that chapter you will find the first steps to work with that tool. For installing that package please follow the next steps:

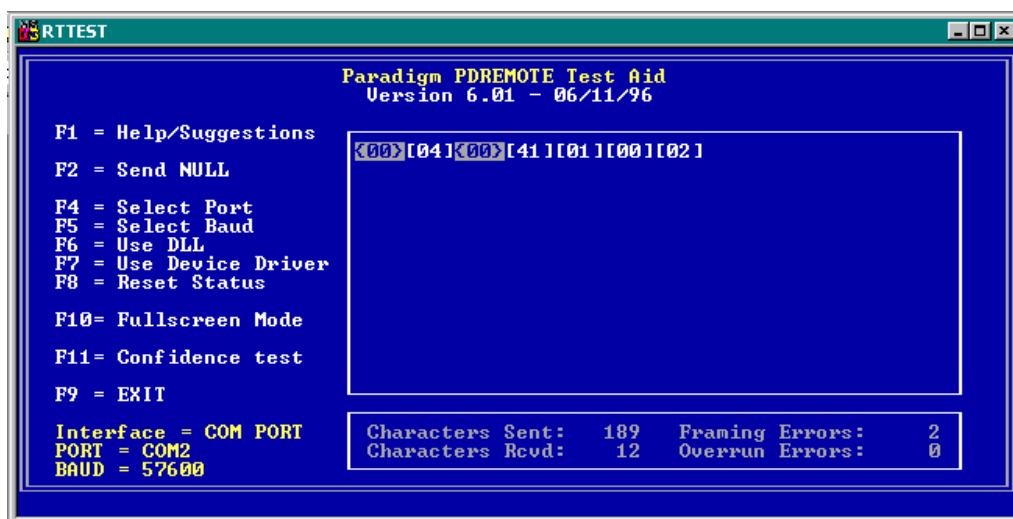
1. Install the Paradigm Locate and DEBUG/RT on your computer. To install that package please have a look in the documentation for that tool.
2. Install the PDREMOTE/ROM on the A2-3. The porting of that target hardware is done in the „A2KIT186.ZIP“ on the additional floppy disk included in that development kit. Unzip that archiv in your project tree. The file „PDREM.HEX“ should be downloaded on the A2-3 using the following commands:

Welcome to AMD186 Monitor ( ? <Enter> for help )

```
mon186: xz ; erase all flash locations
Erasing flash sector(s) ... 8000 9000 E000
mon186:02000002E0001C ; download „PDREM.HEX“
Begin file download ... Press ESC to abort
.....
Device programmed successfully
mon186: p BOOT "18432000,E000,1" ; set autostart entry
mon186: p
BOOT=18432000,E000,1
mon186:@ ; reboot the A2-3
```

Finally you can reboot the A2-3 or jump the PDREMOTE/ROM with the command „G E0000“.

3. Test the communication between host computer and the A2-3 target with the „RTTEST“ tool. Please choose the right com port setting depending on your system and the nominal baudrate of 57600 baud. By pressing two times the F2 key you will see following screen.



The white characters are the response from the PDREMOTE/ROM on the A2-3. With the F11 key you can run a cyclic confidence test in order to test the communication between the device and the host computer. If that test will hang or report some errors you should recompile the PDREMOTE/ROM with a smaller baud rate setting.

The next final step is to start the Paradigm DEBUG/RT. The setting for the communication parameter is defined in the „PDRT186.INI“. Relating the settings of the communication ports, tested with the test tool before, you should change the parameter in the „PDRT186.INI“. As an example of the configuration of the file "PDRT186.INI" see the next lines:

```
; This file is used by Paradigm DEBUG for initialization purposes.  
; Refer to the Paradigm DEBUG manual for a complete list of commands  
; that can be placed in .INI files.  
;
```

```
DEVICE = COM2      ; Communications device : COMn (n=1-4) or CUSTOM  
SPEED = 57600     ; COMn baud rate : 9600, 19200, 38400, 57600, or 115200  
TIMEOUT = 18      ; serial timeout (in DOS ticks, 18 per second)  
FLAGS =           ; Default command line options
```

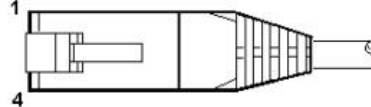
After that initial setting the DEBUG/RT will start properly and communicate with the target system. The PDREMOTE/ROM is capable to drive DEBUG/RT interrupt controlled. Before you start your debug session you should enable that in the setting „Debug controls“ and „Enable dynamic mode“. For the first test in the „A2KIT186.ZIP“ the example project „TIME“ is included in the „SAMPLE“ folder. Based on that example you should have a good starting-point to build and test your own applications with that A2-3 development kit.

## 6. Technical data

- \* **Dimensions:** 115mm x 54mm x 33mm ( B x W x H )
- \* **Weight:** 200g
- \* **Power supply:** 10,8...31,2 V DC  
265 mA at 12V ( max. )  
85 mA at 12V ( idle )  
0,5 mA at 12V ( shutdown )
- \* **Temperature limits:** -25°C bis +70°C ( Storage )  
-20°C bis +55°C ( Operating )
- **Hardware settings:** CpuSpeed 18.432 MHz  
Memory 1Mb Flash, 256Kb RAM  
Serial Devices GSM modem A2, GPS receiver GPSMS1  
IO Device PCF8593 (Real Time Clock, timer or alarm modi)  
24LC32 (32Kbit serial E2PROM)

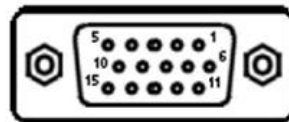
- \* **Interface A:** RJ11 power supply, Cable reference

pin 4 brown 10,8 .. 31,2V  
pin 3 green Ignition  
pin 2 yellow Mute  
pin 1 white GND



- \* **Interface B:** RS232 / V24 and 4 IO ports, 15 pin D-Sub

pin 1 TXD      pin 15 10,8 .. 31,2V (optional 5V)  
pin 2 CTS      pin 11 IO1  
pin 3 DSR      pin 12 IO2  
pin 4 DCD      pin 13 IO3  
pin 5 RI        pin 14 IO4  
pin 6 RXD  
pin 7 DTR  
pin 8 RTS  
pin 9 GND  
pin 10 GND



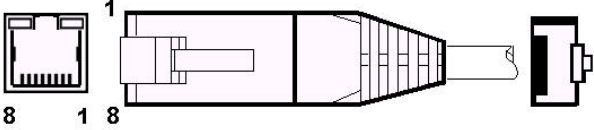
### Electrical Parameter general io ports

Iout max = 200 mA      VinH >= 4,5V  
Vout <= 31,2V      VinL <= 1,2V ( or left open )  
Rin = 470KΩ

### Cable reference for connector 9 pin D-Sub ( modem cable )

<b>DB15</b>	pin 4	to	<b>DB9</b>	pin 1	DCD
	pin 1			pin 2	TXD
	pin 6			pin 3	RXD
	pin 7			pin 4	DTR
	pin 9,10			pin 5	GND
	pin 3			pin 6	DSR
	pin 8			pin 7	RTS
	pin 2			pin 8	CTS
	pin 5			pin 9	RI

- \* **Interface C:** RJ 45 8 pin shielded ( Audio,RS232 )
  - pin 1 10,8 .. 31,2V (optional 5V)
  - pin 2 RXD
  - pin 3 TXD
  - pin 4 GND
  - pin 5 SPK+
  - pin 6 SPK-
  - pin 7 MIC+
  - pin 8 MIC-



  - LED yellow Power
  - LED green Registration
  
- \* **Interface D:** Antenna 50Ω FME female GSM, long cable
  
- \* **Interface E:** Antenna 50Ω FME female GPS, short cable (option)
  - Antenna description: GPS antenna with LNA (low noise amplifier)
  - Frequency range: 1575,42 ± 1,023 MHz
  - LNA gain: ≥ 25 dB
  - Power requirements: 5V ± 0,5V max. 50 mA
  
- \* **SIM interface:** SIM card holder for small SIM cards
  
- \* **Digital interface:** V.24 ( D-Sub 9pin )
  
- \* **Data protocol:** asynchron, transparent and non transparent GSM 07.01, 07.02, 04.21
  - 2400 bps V22 bis
  - 2400 bps V26 ter
  - 4800 bps V32
  - 9600 bps V32
  - 9600 bps V34
  - 2400 bps V110
  - 4800 bps V110
  - 9600 bps V110
  
- \* **Short Message Service:** GSM 03.40, 07.05
  - SMS mobile orginated
  - SMS mobile terminated
  - CMS
  - CBS
  
- \* **Audio interface:**
  - Electret microfon
  - Loudspeaker 150Ω
  - Ground

## 7. General hints

### **THIS CELLULAR MODEM COMPLIES WITH ALL APPLICABLE RF SAFETY STANDARDS.**

This cellular modem meets the standards and recommendations for the protection of public exposure to RF electromagnetic energy established by governmental bodies and other qualified organizations, such as the following :

Directives of the European Community, Directorate General V in Matters of Radio Frequency Electromagnetic Energy

The GSM module FALCOM A2 is licensed with its IMEI number for working in GSM networks. It meets the EC recommendations,

91/263/EWG CTR5 und CTR 9  
ETS 300342-1

confirmed by the CE sign.

You find the actual version of this manual, of the FALCOM A2 user manual and updates at internet homepage " [www.falcom.de](http://www.falcom.de)".

This information serves only for product specification and is in no way legally binding. Leipoldt OHG cannot be held responsible for any damages whatsoever, except in case of gross negligence on our part. We reserve the right to change or modify this product without notice. These operating instructions are protected by copyright. Reproduction is unlawful.

---

For further information please contact:

Funkanlagen Leipoldt OHG  
Gewerbering 6  
98704 Langewiesen  
Tel: (+49)03677/8042-0  
Fax: (+49)03677/8042-215

Internet: <http://www.falcom.de>  
Email: [info@falcom.de](mailto:info@falcom.de)

Revision	Date	Author	Comments
1.00	12.07.1999	R.Georgi	Creation
1.01	31.08.1999	R.Georgi	Additional hardware information
1.02	20.09.1999	R.Georgi	Scematic A2-3

Funkanlagen Leipoldt OHG  
Gewerbering 6  
98704 Langwiesen  
Germany

Tel.: (+49) 03677/ 8042-0  
Fax: (+49) 03677/8042-215

Internet  
DOWNLOAD: [www.falcom.de](http://www.falcom.de)  
EMAIL: [info @ falcom.de](mailto:info@falcom.de)