M2M Control

M2M Control C 100

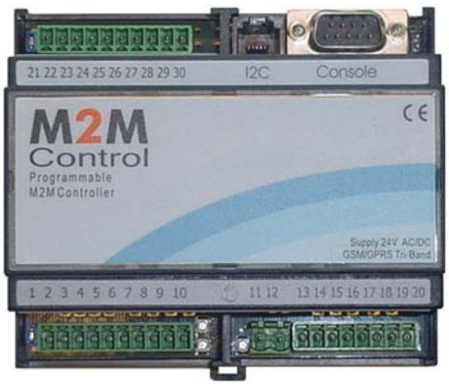
Remote Access & Control module

User and Programmer Manual

INDEX

1 About M2M

M2M means "machine-to-machine" or "man-to-machine" and is to do with the remote monitoring and control of machines.



We offer a total M2M system solution to the market under the brand M2M Control. We provide the means for connecting remote devices to Internet/intranet, enterprise IT systems and other proprietary systems - whether those devices themselves are moving or static.
By doing so, industrial processes can be automated. Widely dispersed machines or apparatus can be brought together into an integrated whole and a wide variety of new services can be created.

Using Wireless M2M communications, a machine can be installed virtually anywhere but still be connected to a support centre to signal performance or need for service. M2M data will improve the service quality and reduce operating costs. Many application areas can be improved using M2M. Assets can be tracked and guarded, refrigerated lorries can be monitored, security alarms can be made more effective, stock control improved. These are just a few examples.

Wireless M2M communications technology is here and will play an indispensable role in reducing costs across all business sectors.

“By 2007, there will be between 100 million and 200 million machine-to-machine connections worldwide that use wireless mobile networks or Internet.

For more details please visit:
www.m2mcontrol.de

2 About this manual

This manual introduces the free programmable generic
M2M Control C100 **Remote Access & Control module.**
On the CD in the back of this manual, you will find documentation and sample source files:
- BASIC sample programs.
- This manual published in Acrobat PDF format ver. 3.0.

Refer to the www.m2mcontrol.de web site for sample programs and any other information.

This manual explains: installation, programming and configuration of the *M2M Control C100* remote control module, as well as the BASIC language.

2.1 Programmer skills

The programmer / engineer have to poses the basic understanding of:
- BASIC programming language
- Communication techniques of GSM / GPRS, CDMA, MOBITEX or Satellite

2.2 Conventions

Edit	Bold Arial font indicates the Command line text that should be typed
Italics	Bold Courier Italics indicates the returned text as a response to a command
"…"	Names like file names, email addresses, web-sites, etc are within quotes.
<..>	Buttons.

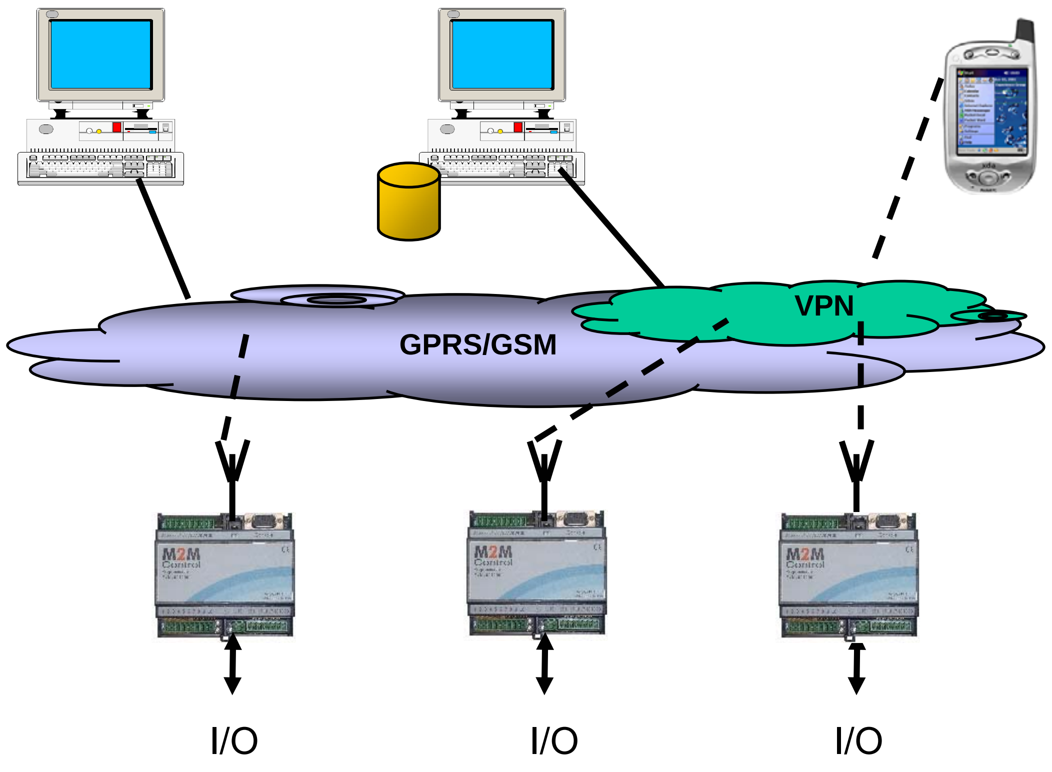
3 About the *M2M Control C100* Remote Access & Control module

The *M2M Control C100* is a generic remote access & control module for self-programming.
This module is available in four versions:
1. C_100_GSM based on GSM communication.
2. C_100_MOB based on MOBITEX communication.
3. C_100_CDMA based on CDMA (Sprint or Verizon) communication.
4. C_100_SAT based on SATELLITE communication using ORBCOM.

This manual explains the hardware and programming of the module.
The term “Remote control” and “Remote access” are used for applications needed for remote management and covers functions such as: control, monitoring, configuration, diagnostics, etc. by means of a communication link.

3.1 Basic network architecture

The *M2M Control C100* Remote Access & Control module is a network module for use in a GSM/GPRS or MOBITEX network. Both types are designed for easy monitoring and control. The latter is achieved by means of a simple BASIC interpreter and easy (remote) command line instructions. This makes the *M2M Control C100* easy to program by anyone with a little understanding of the BASIC programming language.



3.2 Easy programming

The *M2M Control C100* Remote Access & Control module has an embedded BASIC interpreter for easy programming. Simple BASIC instructions give the programmer instant control over Inputs and Outputs, sending SMS’s, e-mail or real-time data. This manual will explain the use of the BASIC programming language in detail in chapter 12, BASIC Tutorial. In chapter 10 you will find many programming examples.

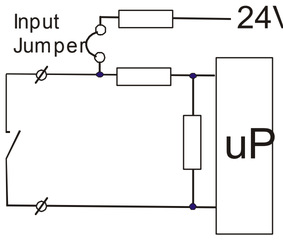
4 Technical specification

The *M2M Control C100* is a generic module with digital and analog Inputs as well as Outputs.

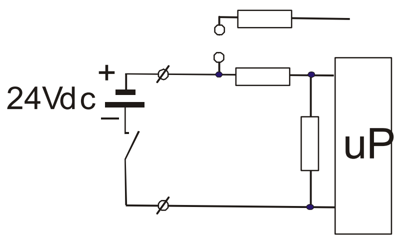
4.1 Digital Inputs

8 x Digital inputs (Group1-4 and Group5-8)

- With Input jumper installed: For “**Volt Free contacts**”, if the contact closes this results in inactive state of the input (0) current through contact is 10mA. The voltage on open contact is then 24Vdc.
- With Input jumper removed: Use of **external 24Vdc** input voltage, 0Volt results in an inactive state of the input, 24V ± 10% is active, and the current through contact is 0.2mA.

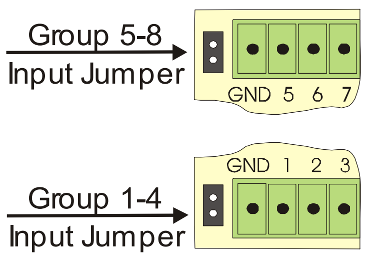


Volt Free contact



External input voltage

In general: HIGH input is ACTIVE



- Insert jumper for VOLTFREE CONTACT INPUT
- Remove jumper for 24VDC INPUT

<u>Group 1-4</u>
Each input has two software representatives called variables:
1 variable per input for the input <u>state</u>: DI1, DI2, DI3 and DI4. Value = 1 or 0 (1 = 24V ON INPUT, SWITCH CLOSED).

1 variable per input for the <u>change</u> of the input:
DE1, DE2, DE3 and DE4. Value = 1 for positive change, -1 for negative change

<u>Group 5-8</u>
Each input has two software representatives called variables:
1 variable per input for the input <u>state</u>: DI5, DI6, DI7 and DI8. Value = 1 or 0 (1 = 24V ON INPUT, SWITCH CLOSED).

1 variable per input as a counter to <u>count</u> input pulses:
DP5, DP6, DP7 and DP8. Value = 0 – 32767

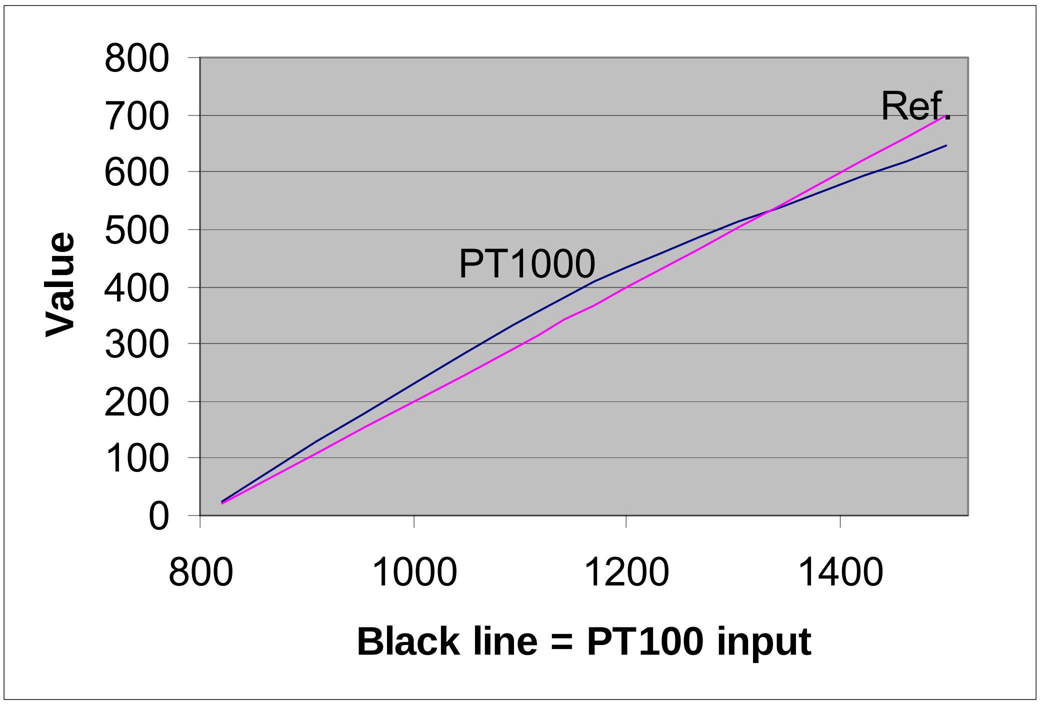
4.2 Digital Outputs

8 x Digital output
24Vdc / 0.2 Amp, depending on simultaneous activity.
Each Output (all 8) has one software representative called variable DOx:
8 variables for the state: DO1, DO2 . . . DO8. Value = 1 or 0 (1 = ON, 24 V OUTPUT).

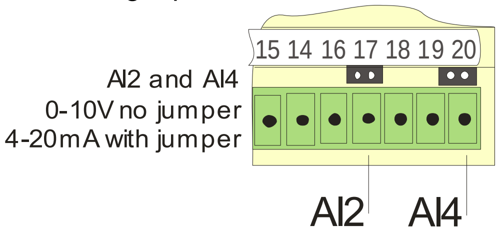
4.3 Analog Inputs

2 x Analog input PT1000 temperature sensor.
Each input has one software representative called variable AIx:
2 variables for the state: AI1 and AI3.
Variables AI1 and AI3 represent temperature with 0.1 decimal accuracy (100 = 10.0 ℃).
- PT1000 mode: (See: graph, purple line is a linear reference)

2	equals	800Ω =	-50º
215	eqeals	1000Ω =	0º
420	eqeals	1200Ω =	50º
580	eqeals	1400Ω =	100º



2 x Analog input 0-10Vdc or 4-20mA configurable by jumper



0 -10Vdc mode: Value = 0 equals 0V, 1000 equals 10V).
4 – 20mA mode: Value = 0 iequals 0mA, 1000 equals 20mA.

4.4 Analog Output

1 x Analog output (0-10Vdc)

The output has a software representative called variable AOx:
1 variable for the <u>state</u>: AO1. Value = 0 – 1000 (0= 0V, 1000 = 10V).

4.5 Serial port

1 x RS232e Serial communication port for console, use XON/XOFF handshake.

4.6 I/O extension port

1 x I^2C interface (for use of extension module, like data logger).

4.7 Power supply

24Vac/dc, 2VA during normal operation,, 4VA during transmission burst.
Typical current consumption <80mA (no transmission)

4.8 Communication

- GSM/GPRS modem with internal antenna (external antenna is optional)

or

- MOBITEX modem with external antenna.

4.9 Modem

GSM/GPRS:	TELIT GM862 PCS
MOBITEX:	CNI type 304ME
	Wavenet type BM3-xxx

4.10 Antenna

Coaxial connector for MOBITEX. This connector is optional for the GSM/GPRS version.

4.11 EEPROM variables

Cell endurance of the EEPROM variables is typically one million writes. Minimum 100.000.
Please calculate the life cycle of your module carefully!

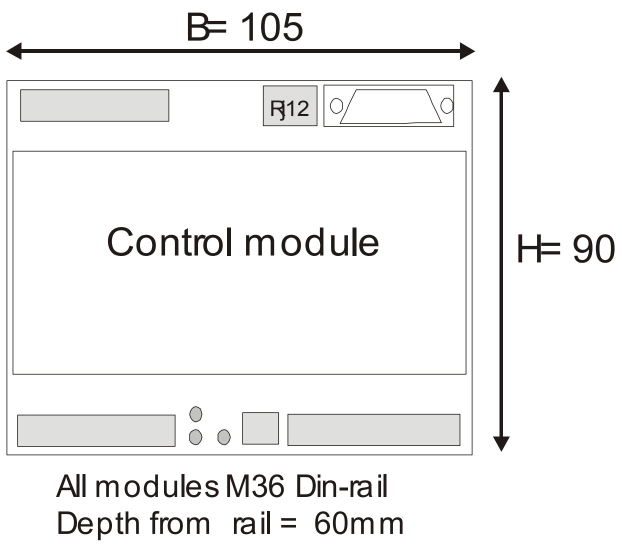
Sample: To store a value in the **EEA** once every 5 minutes.
This is 12 times per hour, 288 times a day, and 104832 per year. With this write frequency you arrive into the critical zone within one year.

Writing to EERPOM should NOT exceed ONCE per HOUR!

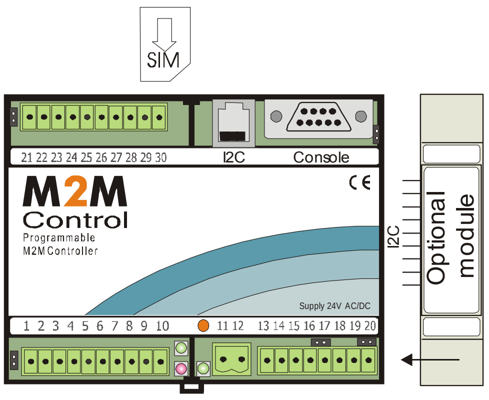
5 Hardware specification

The *M2M Control C100* is a generic module with digital and analog Inputs as well as Outputs.
The module form factor is based on the M36 DIN-rail enclosure standard.

5.1 Physical dimensions



5.2 Connectors/Plugs



Below order numbers for the plugs that fit the different connectors:

24Vac power: Phoenix 5 mm, 2pin
MSTBW 2,5/ 2-ST

DI/O 1-4 Phoenix 3,8 mm, 10pin
MCVW 1,5/10-ST-3,81

DI/O 5-8 Phoenix 3,8 mm, 10 pin
MCVW 1,5/10-ST-3,81

AO Phoenix 3,8 mm, 2pin
MCVW 1,5/2-ST-3,81

AI 1-2 Phoenix 3,8 mm, 6 pin
MCVW 1,5/6-ST-3,81

Console: 9 pin D Female

I^2C: RJ12 6 pin

5.3 Connections

DI/O 1-4

Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9	Pin10
GND	In1	In2	In3	In4	GND	Out1	Out2	Out3	Out4

DI/O 5-8

Pin21	Pin22	Pin23	Pin24	Pin25	Pin26	Pin27	Pin28	Pin29	Pin30
GND	In5	In6	In7	In8	GND	Out5	Out6	Out7	Out8

Attention:
- All Inputs are not floating, when input jumper is installed, 2k2K ohm pull-up to +24V.
- All Outputs are solid-state output

AO 1

Pin13	Pin14
GND	Out

Attention: Output is an op-amp output; you can drive a limited current (1-2mA) to high impedance input only.

AI 1-4

Pin15	Pin16	Pin17	Pin18	Pin19	Pin20
GND	In1	In2	GND	In3	In4

Attention: Inputs are not floating

Console

Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9
-	RXD	TXD	DTR	GND	DSR	RTS	CTS	-

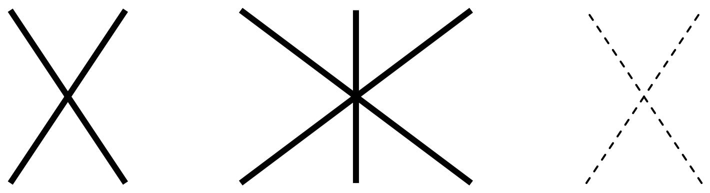
I2C

Pin1	Pin2	Pin3	Pin4	Pin5	Pin6
INT	SDA	+ 5V	+5V	GND	SCL

RS232 Cable

M2M Control C100 Module: 9 pin Female D connector

Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9
DCD	RXD	TXD	DTR	GND	DSR	RTS	CTS	RI



-	RXD	TXD	DTR	GND	DSR	RTS	CTS	-
Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9

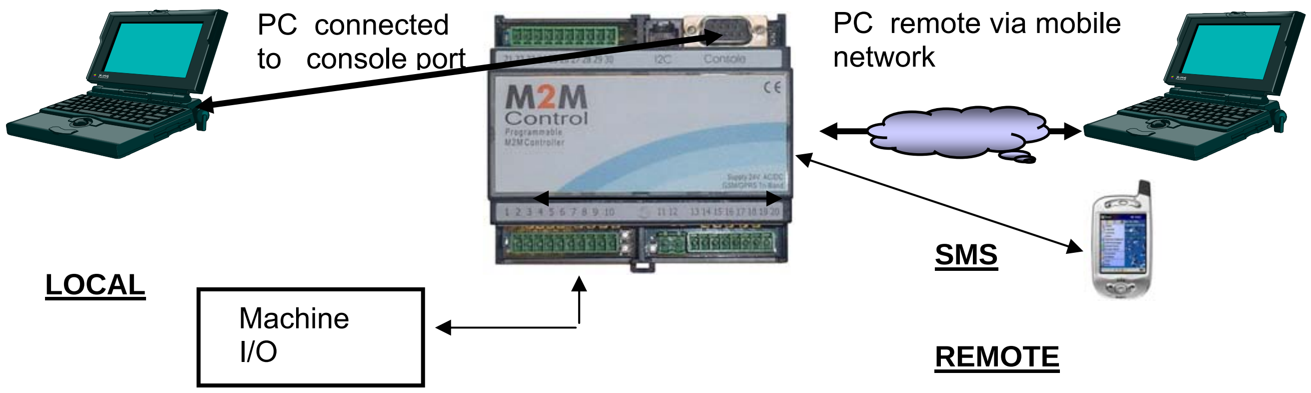
PC: 9 pin Female D connector

Attention: Only pin 2,3,4,5,6 are needed, other pins may be wired.
Beware that some apparatus's or software request DTR high, in that case you need a fully wired cable.

6 Functional description

6.1 Console modes

The *M2M Control C100* Remote Access & Control module can communicate with a PC or application through 2 different paths
a- Local via RS232 console port using a VT100 terminal application (like Hyper Terminal)
b- Remote via GSM (dialup or SMS), GPRS (Telnet LINK) or MOBITEX mobile data LINK.



6.2 Operating modes

The *M2M Control C100* Control module can run in two modes:

1. Command line mode
   This mode is for the operator and enables: testing, programming and configuration of the *M2M Control C100* Control module, either LOCAL or from REMOTE.

2. Execution mode
   This mode is a continuous program execution and enables: I/O manipulation under BASIC program control, automatic sending messages as well as handling of received messages.

6.3 Communication modes

The *M2M Control C100*_GSM / GPRS version (version 2.0) can communicate in 2 ways: Using GSM and GPRS.
This mode is configurable with SET TCP ON / OFF (see SET commands).

The *M2M Control C100*_MOBITEX version has different modem hardware inside.

Funtional difference:

FUNCTION:	GSM/ GPRS	MOBITEX	CDMA	SATCOM
E-mail	X	-	-	X
SMS	X	-	X	-
SEND	X	X	X	-

6.4 Semi modem

The *M2M Control C100* has a semi modem function for any message received by SMS, or MOBITEX that does not start with the valid PASSWORD. All data without PASSWORD will be passed to the Console port. The format is:
For MOBITEX: <MAN#>:<data>
For GSM# or TEL#: <data>
This function is handy for logging unauthorised data traffic or can be used for sending particular data to the receiver console port.

6.5 Power-up

The *M2M Control C100* Control module will start after power-up in the following Operating Mode:

1. Command line mode, if Console is connected to the *M2M Control C100*
2. Execution mode, if NO Console is connected to the *M2M Control C100*

ATTENTION: Removing the RS232 console connector will switch the module automatically into Execution mode!

6.6 Error handling

1. During Execution mode **without** a **Console** connected, the *M2M Control C100* Control module will continue to RUN at all times, therefore while encounting a program error the BASIC program will reset and start at the first program line.
2. During Execution mode **with** a **Console** connected, the *M2M Control C100* Control module will STOP to RUN and show the error text.

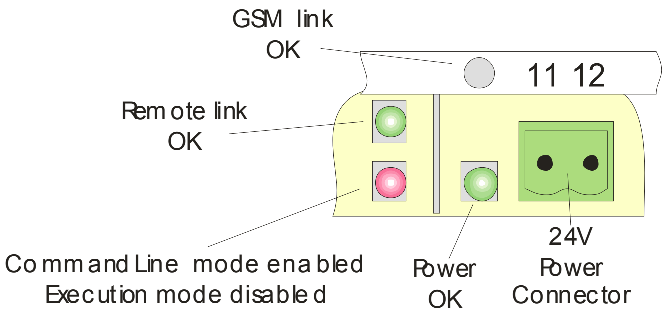
6.7 End of BASIC Program

During Execution mode **without** a **Console** connected, the *M2M Control C100* Control module will reset and start at the first program line after encounting:
- END instruction

or
- last BASIC program line without a RETURN or GOTO instruction.

6.8 LED Indicators

The 3 LED indicators adjacent to the power connector have the following function if ON:



The GSM Link LED

LED status	**Device Status**
Permanent OFF	Device OFF
Fast blinking (period 1s, 0.5s ON	Net search / Not registerd / Turning OFF
Slow blinging (period 3s, 0.3s ON	Registerd full service
Permanently ON	A call is active

7 Command line mode

In the command line mode the user can do the following:

- **LOAD** a basic program into the module, max. 4096 char. program text.
- **LIST** the internal basic program.
- **RUN** the internal basic program starting at the first line (switch to "Execution mode").
- **STOP** the internal basic program (switch to "Command line mode").
- **CONT**inue execution at the position the program was stopped.
- **HELP** shows initial instructions.
- **STEP** 1 line used for debugging (via console only).
- **RESET** go to first valid line.
- **<Enter>** repeat last command.
- **STATUS** shows general status.
- **RSSI** measure signal strength, this is shown with STATUS command (MOBITEX only)
- **CONFIG** shows configuration.
- **! and ? commands** to read and write I/O and variables, see chapter 7.2.
- **GET commands** to get configuration, see chapter 7.3.
- **SET commands** to set configuration, see chapter 7.4.1 - 7.4.2
- **GET xxxxx** Log commands, see *M2M Control* Data logger manual.
- **SEND <host><text>** sends text to host (MOBITEX only).
- **Verbose commands** to follow network messages, see chapter 7.5.
- **OFF** turns modem OFF (MOBITEX version only).
- **ON** turns modem ON (MOBITEX version only).

RUN changes the operating mode from "Command line mode" to "Execution mode"
STOP changes the operating mode from "Execution mode" to "Command line mode"
ATTENTION: Commands are NOT case sensitive load or Load is OK.

7.1 Command line prompts

> Normal prompt
- > Print prompt
>> Step prompt
L> Data from Logger

7.2 ! and ? commands

In the "Command line mode" the operator can type the commands !xx and ?xx to create a direct I/O change or read and write to BASIC variables.
This can be done from a PC connected to the RS232 Console port or from a Remote location; either way gives the same results.

7.2.1 Write to Outputs

BEWARE! ! ! Please know what outputs do. It can harm persons or machine parts to switch ON particular outputs without a thorough understanding of its consequences!

Type the following commands:

!DO1=1	Sets Digital output1 ON
!DO1=0	Sets Digital output1 OFF
!DO1=1,DO2=1,DO3=0	Sets Digital output1 ON, output2 ON, output3 OFF
!AO1=500	Sets Analog output to 5Vdc

7.2.2 Write to Variables

For debugging purpose it is very handy to have direct control over internal BASIC variables.
You can write any variable using the **!** – command.
You can write to the following variables: **A…Z, TM1…TM8, DO1…DO8, AO1, DE1…DE4, DP5…DP8, EEA…EEJ**

!A=10	Sets BASIC variable A to 10
!A=10, B=25	Sets BASIC variable A to 10 and B to 25

ATTENTION!
All command line commands starting with **!** are <u>**disabled**</u> during **execution** mode, unless you SET USERIO ON.

The **!** command can be used from the Console, Dail-up or Direct GPRS link.
Using **Mobitex** or **SMS** you need to add the password for example,
admin!DO1=1,DO2=0,DO3=1

7.2.3 Read Input or Output status

Type the following commands:

<u>**Digital I/O**</u>

?DI1	reads DI1, returns: ***?DI1=0*** *or* ***?DI1=1***
?DI1,DI4	reads DI1 and DI4, returns e.g.: ***?DI1=0,DI4=0***
?DI0	reads all inputs, returns: ***?DI1=0,DI2=0,DI3=0,…****etc.*
?DO0	reads all outputs, returns: ***?DO1=0,DO2=0,DO3=1…****etc.*

<u>**Analog I/O**</u>

?AI2	reads AI1, returns e.g. : ***?AI2=460***
?AI0	reads all inputs, returns e.g.: ***?AI1=246,AI2=1000***
?AO1	reads analog output value: ***?AO1=500***

Easy Way!
The command line **STATUS** command will give you an overview of all I/O status at once!

7.2.4 Read internal BASIC variables

For debugging purposes you can use the PRINT statement in your program or use the command line **?** – command to get the value of internal BASIC variables.
You can read to the following variables: **A…Z, TM1…TM8, DI1…DI8, DO1…DO8, AO1, DE1…DE4, DP5…DP8, DV5…DV8, EEA…EEJ**

?A	reads BASIC variable A and returns e.g.: ***?A=10***
?TM2	reads BASIC Timer2 status and returns e.g.: ***?TM2=-1***
?DE1	reads Input1 Event status and returns e.g.: ***?DE1=-1***
?DP6	reads Input6 Counter status and returns e.g.: ***?DP6=234***

7.2.5 ! and ? commands via SMS

The Write (!) and Read (?) commands can be sent to the module by SMS. In case of a Read request the module will return an SMS message. Please be aware that it can take a while to receive an SMS message. This is stongly depending on the network availability.
Using SMS you need to add the password, for example **admin!DO1=1,DO2=0,DO3=1**

7.3 GET commands

For reading date and time:

7.3.1 GET DATE

Fetch date

7.3.2 GET TIME

Fetch Time

7.4 SET commands

SET command is used for configuration of the module once. All SET command entries are stored in EEPROM and non-volatile.

7.4.1 SET DATE

Set date for example SET DATE MON 01-01-07

7.4.2 SET TIME

Fetch Time for example SET TIME 00:00:00

ATTENTION: Please realize that the C100 will loose date&time data after power down.

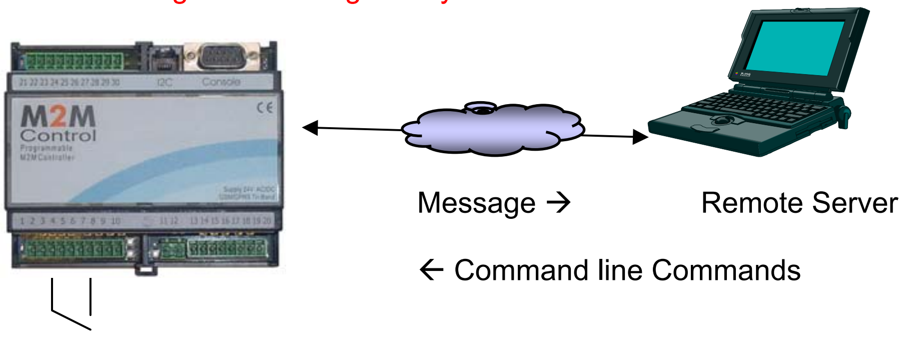
7.4.3 SET ID xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Module ID string will be used in e-mail. This ID is maximum 30 characters and available as static Basic variable ID.

7.4.4 SET EVENTS

SET EVENTS ON (MOBITEX and GPRS only)
SET EVENTS OFF (MOBITEX and GPRS only)

EVENTS ON is used for REAL-TIME monitoring with a remote application. This gives an instant update of the I/O status without the need of polling. It works with a real-time link, MOBITEX or GPRS only. In case **EVENTS** is set to **ON** the event is sent as an auto response like a read command with a leading #, e.g. #DI1=0.
ATTENTION!
Events are returned for Digital I/O changes only.



EVENT: Contact Closes, Message is send to Remote Server #DI3=1
EVENT: Contact Opens, Message is send to Remote Server #DI3=0

To monitor changes of **analog input values** use the **PRINT** or **SEND** command in your program to POLL the variable.

7.4.5 SET USERIO ON/OFF

USERIO On will make it possible to overrule an Output or Basic variable while executing a basic program. E.g. from local console or remote connection !DO1=1 will set DO1 to ON while the program is running.
ATTENTION: Please be aware that program execution can reverse this action.
For commands passed by MOBITEX or SMS, you need to add the PASSWORD like: Admin!DO1=1

7.4.6 SET PIN xxxx

A 4 digit PIN for SIM card is stored in EEPROM memory
ATTENTION: PIN is not shown in Config screen!

7.4.7 SET PASS xxxxxxxxxx

Enter password for Command Line protection (maximum 10 char.)
ATTENTION: Password is CASE SENSITIVE and not shown in Config screen!

7.4.8 SET TCP ON / OFF (Not for Mobitex version)

With this setting the GPRS link with a Host is activated or deactivated

7.4.9 SET APN xxxxxxxxxxxxxxxxxxxx (GPRS ONLY)

This is the Access Point Name for Internet access (maximum 20 char.)

7.4.10 SET TCPSUSER xxxxxxxxxxxxxxx (GPRS ONLY)

This is the user name for Internet access (maximum 15 char.)

7.4.11 SET TCPPASS xxxxxxxxxxxxxxx (GPRS ONLY)

This is the password for Internet access (maximum 15 char.)

7.4.12 SET MAILUSER xxxxxxxxxxxxxxx (GPRS ONLY)

This is the mail user name for Mail Server access (maximum 15 char.)

7.4.13 SET MAILPASS xxxxxxxxxxxxxxx (GPRS ONLY)

This is the password for Mail Server access (maximum 15 char.)

7.4.14 SET MAILSUBJECTxxxxxxxxxxxxxxx (GPRS ONLY)

Sample: Email from C100 device (maximum 30 char.)

7.4.15 SET HOST xxxxxxxx (GPRS ONLY)

With this command the remote host number is set.
- For MOBITEX HOST = MAN number (maximum 8 char.)
- For GPRS HOST = IP address of host or host name 123.234.345.567:9876 (maximum 30 char.)
- For GPRS HOST = URS of host or host name www….. (maximum 30 char.)
- For GPRS HOST = 0 No HOST connection used

7.4.16 SET PORT xxxxx (GPRS ONLY)

This is the IP port of the Host (maximum 65535.)
- For GPRS PORT = 0 No HOST connection used

7.4.17 SET SMTP xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (GPRS ONLY)

This is the Mail Server Name or IP address for sending an email (maximum 30 char.)

7.4.18 SET FROM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (GPRS ONLY)

This is the Sender email address for sending an email (maximum 30 char.)
The *M2M Control* 100 can not receive emails. Use any usefull address like mymodule@myplace.com.

7.4.19 SET MAIL1..5 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (GPRS ONLY)

With this command the email addresses are stored in register 1 to 5. (max 30 char.)
Use the – sign to clear number, e.g. SET MAIL2-
Example: SET MAIL1 m2m@M2M-Technologies.de, address is stored in first register
Use the **CONFIG** command to see all stored addresses on your console screen.

7.4.20 SET SMS1..10 xxxxxxxxxxxxxxx (max 15 char.) (For GSM only)

Use – sign to clear number
With this command the SMS number is stored in register 1 to 10.
Example: SET SMS1 +31623456789, number +31623456789 is stored in first register
SET SMS5 +31634567890, number +31634567890 is stored in fifth register
Clear number with: SET SMSn -
Use the **CONFIG** command to see all stored numbers on your console screen.

7.4.21 SET UPDATE ON (via local console only)

Used for updating the firmware of the module. See for new firmware releases and instructions www.m2m-control.eu

7.5 VERBOSE commands

For diagnostic purposes only:
The following commands are for the experienced engineer only and are worked from the Console port only.

If LINKVERBOSE is set to 1 or 2 the link events information is sent to the console.
Please note that there is a difference in GSM and MOBITEX.

<u>For MOBITEX version only</u>
LINKVERBOSE X
X = **0** verbose state is turned off
X = **1** link events only + state of link layer
X = **2**, is LINKVERBOSE 1 + commands
See appendix (chapter 13.2) for LINKBERBOSE state information

ATTENTION!
The following commands are to show the MOBITEX MASC protocol only!
The user needs a thorough understanding of the MASC protocol.

MASCVERBOSE X
X = **0** verbose state is turned off
X = **1** shows events and states in mask layer: See appendix (chapter 13.2) for MASC state information

RAMVERBOSE X
X = **0** verbose state is turned off
X = **1** shows status and traffic state: See appendix (chapter 13.2) for RAM state information

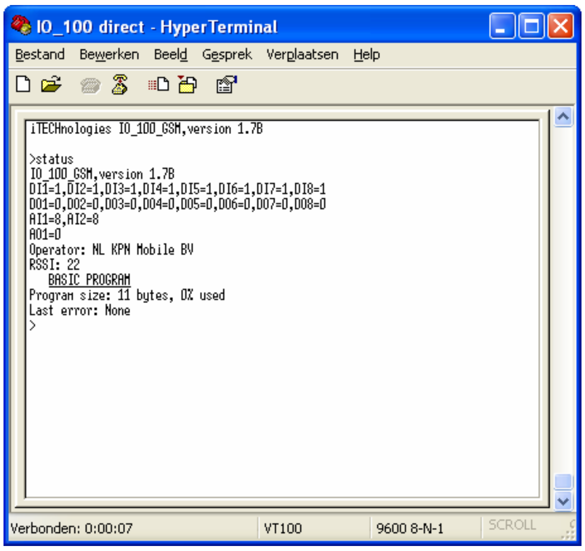
For GSM/GPRS version only
MODEMVERBOSE X
X = **0** verbose state is turned off
X = **1** shows modem states: See appendix (chapter 13.1) for modem state information

7.6 STATUS

The **STATUS** command is very usefull and shows the general status of the module.

First all Inputs and Outputs values
- DI1…DI8
- DO1…DO8
- AO1
- AI1 and AI2
- Mobile net operator

RSSI signal strength & quality
- RSSI: 0 is NO signal
- RSSI: 1…5 is 1 bar
- RSSI: 6…9 is 2 bars
- RSSI: 10…14 is 3 bars
- RSSI: 15…31 is 4 bars
- RSSI: 99 not detected signal strength
- Basic program size
- Last error: Basic program error


IO_100 direct - HyperTerminal
Bestand Bewerken Beeld Gesprek Verplaatsen Help
Verbonden: 0:00:07
VT100
9600 8-N-1
SCROLL

```
iTECHnologies IO_100_GSM,version 1.7B

>status
IO_100_GSM,version 1.7B
DI1=1,DI2=1,DI3=1,DI4=1,DI5=1,DI6=1,DI7=1,DI8=1
DO1=0,DO2=0,DO3=0,DO4=0,DO5=0,DO6=0,DO7=0,DO8=0
AI1=8,AI2=8
AO1=0
Operator: NL KPN Mobile BV
RSSI: 22
   BASIC PROGRAM
Program size: 11 bytes, 0% used
Last error: None
>
```

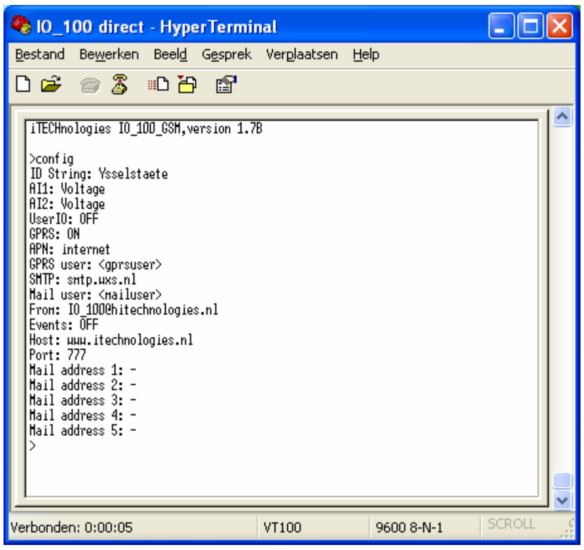
ATTENTION: for **MOBITEX**
To see the signal strength you need to execute the **RSSI** command first!

7.7 CONFIG

The **CONFIG** command is very usefull and shows the configuration of the module.

Configuration info:
USERIO, GRPS settings, Events, etc…

E-mail addresses or SMS phone book


IO_100 direct - HyperTerminal
Bestand Bewerken Beeld Gesprek Verplaatsen Help
Verbonden: 0:00:05
VT100
9600 8-N-1
SCROLL

```
iTECHnologies IO_100_GSM,version 1.7B

>config
ID String: Ysselstaete
AI1: Voltage
AI2: Voltage
UserIO: OFF
GPRS: ON
APN: internet
GPRS user: <gprsuser>
SMTP: smtp.wxs.nl
Mail user: <mailuser>
From: IO_100@hitechnologies.nl
Events: OFF
Host: www.itechnologies.nl
Port: 777
Mail address 1: -
Mail address 2: -
Mail address 3: -
Mail address 4: -
Mail address 5: -
>
```

7.8 Mobitex special

Because of the limited data size for MOBITEX messages, CONFIG, STATUS and LIST will arrive in blocks.

Block contents:
STX<identifier>:n<message part>ETB/EOT

All blocks start with "**Start of Text**" (0x02),
Followed by an identifier
- **C** for CONFIG
- **S** for STATUS
- **L** for LIST

Followed by "**:**" plus Block **NUMBERS** (starts with 0x01, till 0x255).
Followed by **MESSAGE** (max 60 bytes)
Followed by **End Transmission Block** (0x17), only the last block will close with **End Of Transmission** (0x04**)**.

HELP command via Mobitex is not supported.

7.9 SEND <host> <text> (from Console)

This command enables the MOBITEX user to do a quick communication test over the network.
If you send a command to a remote module you need to add the password, for example:
SEND 12345678 admin!DO1=1<Enter>
If you send a text string to a remote module, you do the following:
SEND 12345678 this is a text string <Enter>
This string ”this is a text string” will be directed to the Console port.

Similar to the PRINT-command, but sends the specified text through the *M2M Control* 100 host link to the attached server. You can check system variable CTSE (Clear To SEnd) to see whether there is a working host link (0=no link available, 1=link available). After sending the text, you can use system variable SERE (SEnd REsult) to see whether the message has been sent to the server host successfully. It is reset on each SEND command, and gets a value of 0 if the message was successfully sent. Any other value indicates an error. (For now, the only other value it will have is 1, but future versions of the *M2M Control C100* might give more specific error codes).

Execution mode
In the execution mode the processor will execute a BASIC program line by line, which is loaded in FLASH memory; The BASIC interpreter can perform the following instructions:

7.10 Instructions

• **PRINT …**	sends output to the console port, max 80 char.
• **IF ... THEN … ELSE**	conditional instruction
• **AND OR**	conditional instruction used in combination with IF…THEN
• **GOTO**	jump instruction
• **FOR … TO … NEXT**	loop instruction
• **GOSUB … RETURN**	jump to sub-routine, returns in next line
• **SEND …**	sends output to the remote receiver, max. 59 char.
• **SMS x, …**	sends output as SMS, max 80 char (NOT FOR MOBITEX or GPRS)
• **MAIL x, …**	sends output as E-Mail, max 80 char.
• **REM**	used to enter remarks
• **END**	terminate program execution, works with console connected only
• **LOG …**	sends output to optional data logger, max 50 char (like print …)

ATTENTION: Instructions are NOT case sensitive print or Print is OK

7.11 Remote use of ! and ? commands during execution

The user can influence the module in two ways:

1. Direct terminal connection via GSM Dial-in or GPRS host command session.
   This mode is like a direct console connection. ?-commands can be executed during execution mode, !-commands can be executed after program STOP only.

   ATTENTION: Please realize that while you SET an Output or Variable, that after the re-start of program execution the I/O or variables might change right back, you need to understand the application.

2. Send a SMS to the *M2M Control* module
   With a SMS message you can do everything that fits in ONE message.
   You need to put the valid PASSWORD as PREFIX, e.g. **admin!DO1=1**
   This command will set Digital Output1 to ON instantly with no need to STOP the BASIC program first.

   The SMS command: **admin?DO1,DI1,AI1** will result in receiving the following SMS: **?DO1=1,DI1=0,AI1=123**

   Long text outputs of STATUS and CONFIG are not supported by SMS, this would result in several SMS messages and is not very practical. Please use the Dial-up methode for STATUS and CONFIG.

   ATTENTION: Please realize that while you SET an Output or Variable, the program is still executed and might change your setting right back, you need to understand the application.

7.12 Predefined variables

The interpreter has the following pre-defined variables:

A … Z	as integer, -32768 ……+32767
EEA … EEJ	NON VOLATILE integer, -32768 ……+32767
ID	ID of maximum 30 characters READ ONLY, see SET ID for details
DI1 … DI8	for the 8 digital inputs STATUS, ATTENTION! DIx are READ ONLY variables, the values are 1 or 0
DE1 … DE4	for digital input EVENT, value = 1 for positive change, -1 for negative change
DP5 … DP8	for digital input COUNTER, value = 0 – 32767 (max 5 Bytes)
DV5 … DV8	for digital input COUNTER, value = 0000 – 7FFF (2 Bytes) compact hex data Read only and slave value of DPx
DO1 … DO8	for the 8 digital outputs STATUS, the values are 1 or 0
AI1 and **AI3**	2 analog inputs for temperature, value = 0 -1000 (0.1 degree C)
AI2 and **AI4**	2 analog inputs for Current or Voltage, value = 0 (0 - 20mA or 0 - 10V)
AO1	for the analog output, value = 0 -1000 (0 -10V)
TM1 … TM8	for 100 milliseconds timer, ATTENTION! if read, this variable returns 0, 1 or –1 0 = Timer EXPIRED, after once read this value becomes –1 1 = Timer is RUNNING -1 = Timer is IDLE Timer SET value is max. 32767 this equals 3276,7 seconds
SEC	contains the number of seconds since the last minute lapse (0-59)
MIN	contains the number of minutes since the last hour lapse (0-59)
HRS	contains the number of hours since midnight (0-23)
DAY	contains the day of the month (1-31)
WDAY	contains the day of the week (0-6, 0=Sunday)
MTH	contains the month (1-12)
YRS	contains the number of years since 2000 (0-255)
TRE	(Time REsult) is 0 when the *M2M Control C100* thinks it has a valid time.
CTSM	Clear To SMs must be 1 before using the SMS command
CTSE	Clear To SEnd to see whether there is a working host link (0=no link available, 1=link available)
CTM	Clear To Mail must be 1 before using the MAIL command
MRE	Mail REsult, equals 0=ok, 1 indicates error, the mail was sent
SMRE	SMs REsult, 0=ok, 1 indicates error, SMS was not sent
SERE	SEnd REsult, 0=ok, 1 indicates error, SEND message was not sent

See chapter 12 for more details on the use of variables.

7.13 Operands

The interpreter can manipulate variables with the following operands:

+ - * ^ / % = > < <> & |

See chapter 12.4 for more details on the use of operands.

7.14 Delimiters

The interpreter can handle the following delimiters:
; : ( ) ,
See chapter 12.5 for more details on the use of delimiters.

7.15 Program example

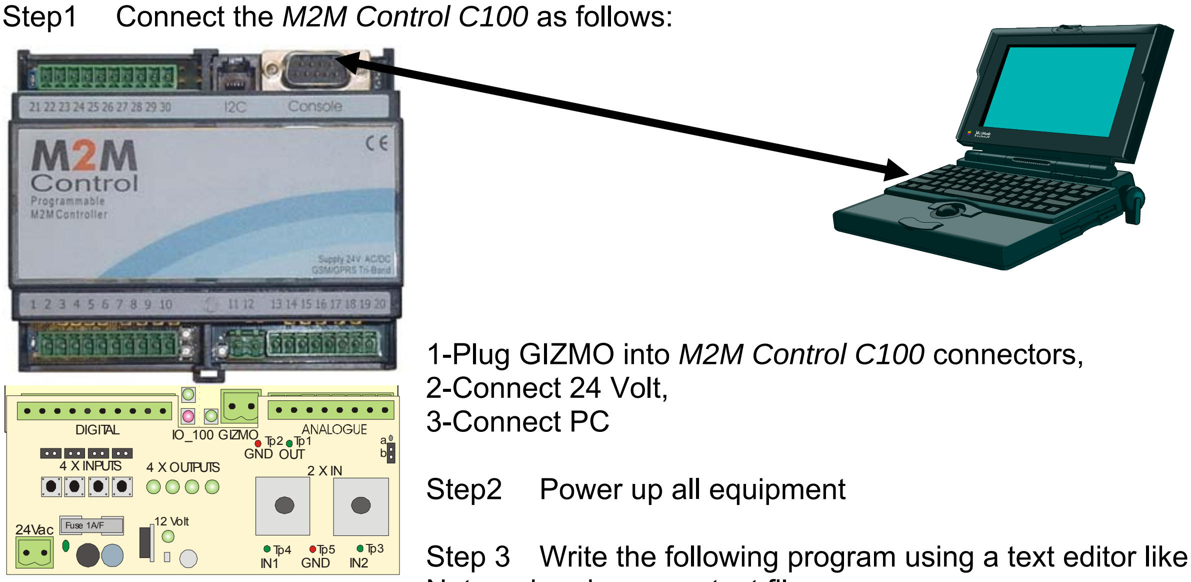
```
PRINT "This is an example of a BASIC program."
REM version 1.0
A=1:B=5:X=0
TM1=50
TM2=0
100 IF DI1 THEN GOSUB 200
IF DE1=-1 THEN SMS 1 “ALARM ON INPUT 1”
IF AI1<200 then DO2=1
IF TM1=0 THEN GOSUB 300
IF TM2<1 THEN DO2=0 ELSE DO2=1
GOTO 100

200 PRINT "X=";X,"A=";A,"B=";B
FOR T=1 TO 5
 X=X+T
 A=A*B
 PRINT "X=";X,"A=";A,"B=";B
NEXT
RETURN

300
 TM1=50
 TM2=10
RETURN
#
```

8 Let's start

8.1 1st step, first program

Let's start to load the first program.
Use any text editor to write a program, always save as plain ASCII text.
In case you use a word processor program like MS WORD, than save as .TXT text file.

Step1 Connect the *M2M Control C100* as follows:



1-Plug GIZMO into *M2M Control C100* connectors,
2-Connect 24 Volt,
3-Connect PC

Step2 Power up all equipment

Step 3 Write the following program using a text editor like Notepad and save as text file:

```
TM1=10
10 IF TM1=0 THEN GOSUB 100
GOTO 10
100 TM1=10
IF DO1=0 THEN DO1=1:DO2=0 ELSE DO1=0:DO2=1
RETURN
#
```

Step 4 Start HyperTerminal with the following configuration:
- Baudrate 9600
- 8 data bits, 1 stop bit, no parity, Hardware handshake XON/XOFF
- Set to VT100, Enable "Append Line feeds to incoming line ends" in ASCII settings
- Press on <Enter>; this should give a response with name, version and password

Step 5 Enter password: **admin** (factory default)

Step 6 Load the basic program into the module
Type **LOAD**
Select Transfer, Send Text file, Select File
After program load succeeds: the text "Program loaded" will show on your terminal
Possible errors: "program to big! Exceeds 4096 characters"

Step 6 List the internal basic program
Type **LIST**
Please look carefully if your entire program is in memory. (Please note: without #)

Step 7 If the program was loaded successfully then you are ready to Run the internal basic program
Type **RUN**

Step 8 Check how the outputs 1 and 2 alternate with a frequency of 1 second.

Step 9 Stop the internal basic program
Type **STOP**

Congratulations! you just managed to write and run your first program.

8.2 About programming in BASIC

For full details about BASIC programming refer to chapter 12 at the end of this manual.
For those who understand BASIC, let's take a closer look at this first program:

```
Line 1  TM1=10
Line 2  10 IF TM1=0 THEN GOSUB 100
Line 3  GOTO 10
Line 4  100 TM1=10
Line 5  IF DO1=0 THEN DO1=1:DO2=0 ELSE DO1=0:DO2=1
Line 6  RETURN
Line 7  #
```

`Line 1  TM1=10` — This initialises Timer1 for 10 x 100 mS. This equals 1 second.
The value of the variable "`TM1`" will become 0 as soon as it times out after 1 second.

`Line 2  10 IF TM1=0 THEN GOSUB 100`
`Line 3  GOTO 10` — These two lines form a loop and do check if the timer is finished. 10 and 100 are called labels. Within your program you can make conditions that if TRUE jump to a label.

`Line 4  100 TM1=10` — This is the start of a sub-routine; Timer is set to 1 second again.

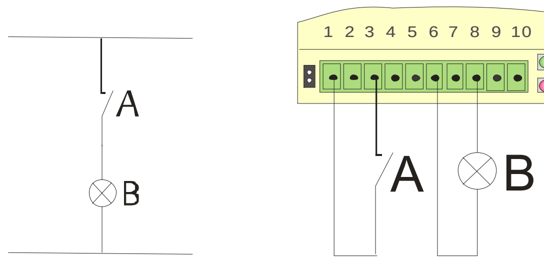
`Line 5  IF DO1=0 THEN DO1=1:DO2=0 ELSE DO1=0:DO2=1`
This is the actual function
It alternates Output 1 and 2

`Line 6  RETURN` — This is the end of a sub-routine

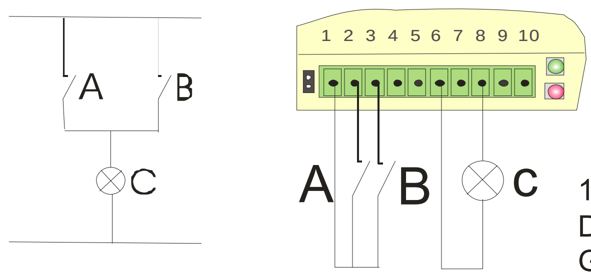
`Line 7  #` — Always put a # sign at the end.
This is used as an end of program sign by the loader.

9 Sample BASIC programs

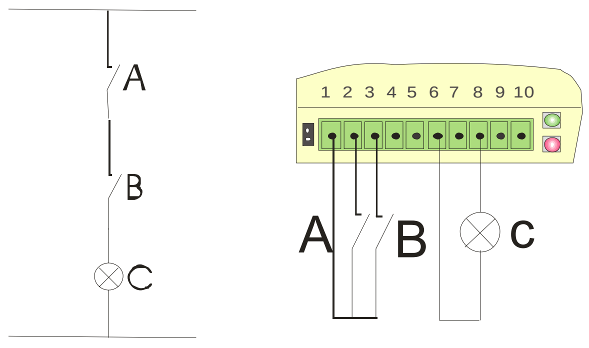
Below are some comparisons between a standard schematic and the software derivative.



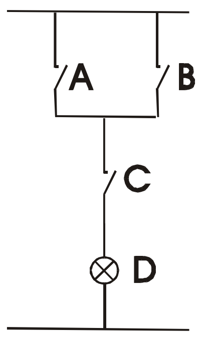
```
10 REM   A= DI2, B= DO2
DO2= DI2
GOTO 10
```

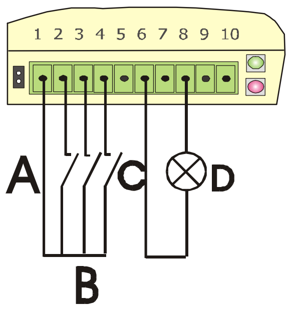


```
10 REM   A= DI1, B= DI2, C= DO2
DO2= DI1 OR DI2
GOTO 10
```

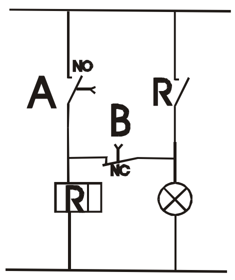


```
10 REM   A= DI1, B= DI2, C= DO2
IDO2= DI1 AND DI2
GOTO 10
```

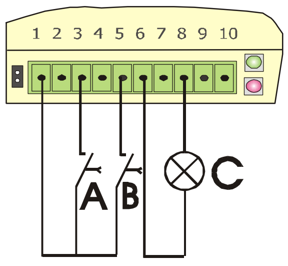




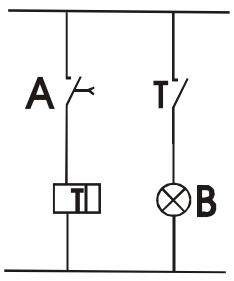
```
10 REM   A= DI1, B= DI2, C= DI3, D= DO2
DO2=  DI3 AND  (DI1 OR DI2)
GOTO 10
```



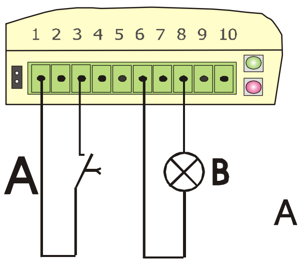
A =ON, B=OFF



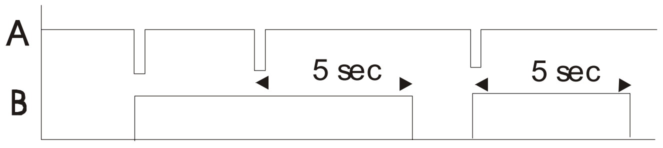
```
REM   A= DI2, B= DI4, C= D02
10 IF DI2 THEN DO2= 1: REM ON
IF DI4 THEN DO2= 0: REM OFF
GOTO 10
```

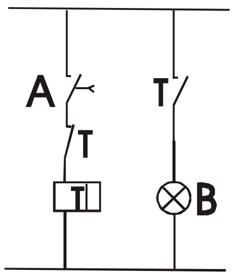


T=Timer,   A= Push button

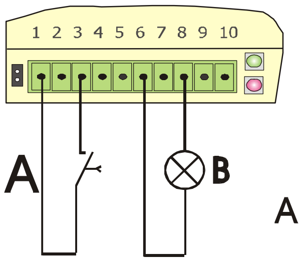


```
REM   A= DI2, B= DO2
REM   TM1= 50 is  5 sec
10 IF DI2 THEN TM1= 50
IF TM1= 0 THEN DO2= 1 ELSE DO2= 0
GOTO 10
```

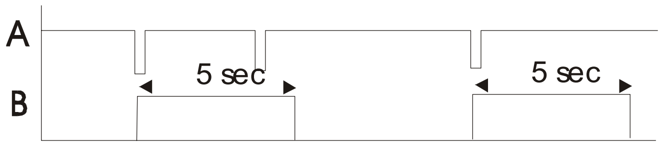




T=Timer,   A= Push button



```
REM   A= DI2, B= DO2
REM   TM1= 50 is  5 sec
10 IF DI2 AND TM1= -1 THEN TM1= 50
IF TM1= 0 THEN DO2= 1 ELSE DO2= 0
GOTO 10
```



First program

This program switches ON the LED’s 1 – 4 with a few seconds in between. At A=500 ALL LED’s are switched OFF and the process starts all over again.
The timing is produced by the loop of a hundred times returning to line 30 and incrementing A.

```
10 REM this is the first demonstration program
20 PRINT "Hello world, this is my first program"
30 A=A+1
PRINT A
IF A=100 THEN DO1=1
IF A=200 THEN DO2=1
IF A=300 THEN DO3=1
IF A=400 THEN DO4=1
IF A=500 THEN DO1=0:DO2=0:DO3=0:DO4=0:A=0
GOTO 30
#
```

Second program

This program consist of two small demonstrations of the use of the IF…THEN statement

```
10 PRINT "2nd program demo of IF...ELSE"
20 A=0:DO1=0
30 PRINT "DI1=";DI1,"A=";A
IF DI1=1 THEN A=10:DO1=1 ELSE A=0:DO1=0
GOTO 30
#
```

```
PRINT "2nd program demo of IF...AND..."
PRINT "and   IF...OR..."
20 DO1=0:DO4=0
30 IF DI1=1 AND DI2=1 THEN DO1=1 ELSE DO1=0
40 IF DI3=1 OR DI4=1 THEN DO4=1 ELSE DO4=0
GOTO 30
#
```

Third program
This program demonstrates the use of sub-routines.
With DI1 you trigger an output pulse of 5 seconds.
You need to return the switch to reset and activate the next pulse.
With DI2 ON you print out the output state.

```
10 PRINT "3rd program demo of sub-routines"
REM PULSE OUTPUT, START WITH DI1, PRINT VALUES WITH DI2
20 A=0:B=0
30 IF DI2=1 THEN PRINT "DI1=";DI1," DO1=";DO1," TM1=";TM1
REM ATTENTION!, Printed value of TM1 is 1 (ON) or -1 (OFF)
IF DI1=1 THEN GOSUB 100
GOSUB 200
GOTO 30

100 IF A=0 THEN TM1=50
REM TM1 = 50 equals 5 seconds
A=1:B=1
RETURN

200 IF TM1=1 THEN DO1=1
IF TM1<1 THEN DO1=0:IF DI1=0 THEN A=0:GOSUB 300
RETURN

300 IF B=1 THEN PRINT "Reset"
B=0
REM This is a second level sub-routine
RETURN
#
```

Fourth program

This program creates output pulses with low frequency on DO2 and with high frequency on DO1. You switch between the two frequencies with DI1.
This demonstration program comes in two parts.
4a - using a FOR…NEXT loop
4b - using timer TM1

```
REM This is a demonstration program
10 PRINT "Program 4th demo of FOR...NEXT"
20 A=0:B=0:DO1=0:DO2=0
IF DI1=1 THEN DO1=1:GOSUB 100 ELSE DO2=1:GOSUB 200
GOTO 20

100 FOR A=1 TO 10
PRINT "A=";A
IF A=5 THEN DO1=0
NEXT
RETURN

200 FOR B=1 TO 30
PRINT "B=";B
IF B=15 THEN DO2=0
NEXT
RETURN
#

After the # (end of program sign) you can type any text.
This program demonstrates the working of a FOR...NEXT loop.
```

```
PRINT "4b as program 4a with timer"
A=0
20 IF DI1=1 THEN A=2 ELSE A=5
GOSUB 100
GOTO 20
100 PRINT "A=";A
IF TM1<1 THEN TM1=A:GOSUB 200
RETURN

200 IF A=2 AND DO1=0 THEN DO1=1 ELSE DO1=0
IF A=5 AND DO2=0 THEN DO2=1 ELSE DO2=0
RETURN
#
```

Fifth program

This program demonstrates all kinds of calculation tricks.

```
10 PRINT "5th program demo of calculation"
20 A=1:B=2:C=3:D=4:E=5:F=20
PRINT "A=";A,"B=";B,"C=";C,"D=";D,"E=";E,"F=";F
A=B+C
PRINT "A=B+C   A=";A,"B=";B,"C=";C
A=E-C
PRINT "A=E-C   A=";A,"E=";E,"C=";C
A=B*C
PRINT "A=B*C   A=";A,"B=";B,"C=";C
A=F/C
PRINT "A=F/C   A=";A,"F=";F,"C=";C
A=F%C
PRINT "A=F%C   A=";A,"F=";F,"C=";C
A=B^D
PRINT "A=B^D   A=";A,"B=";B,"D=";D
A=(B+C)*(E+F)
PRINT "A=(B+C)*(E+F)  A=";A,"B=";B,"C=";C,"E=";E,"F=";F
#
```

Sixth program

This program demonstrates the STEP function.
Do not use RUN but use STEP.

```
10 PRINT "6th program demo STEP function"
20 PRINT "this is line 2"
PRINT "this is line 3"
PRINT "this is line 4"
PRINT "this is line 5"
PRINT "this is line 6"
PRINT "this is line 7"
PRINT "this is line 8"
PRINT "this is line 9"
PRINT "this is line 10"
GOSUB 100
GOTO 20

REM   SUB ROUTINES
100 PRINT "this is line 15"
GOSUB 200
RETURN
200 PRINT "this is line 18"
RETURN
#
```

Seventh program

This program demonstrates the use of a timer by calculating minutes, hours and days.

```
10 PRINT "7th program demo timer function"
20 REM set timer to 600 this equals 1 minute
30 TM1=600:M=0:U=0:D=0
40 IF TM1=0 THEN GOSUB 100
GOTO 40
REM SUB ROUTINES
100 TM1=600
M=M+1
IF M=60 THEN M=0:U=U+1
IF U=24 THEN U=0:D=D+1
IF D=7 THEN D=0
PRINT "M=";M," U=";U," D=";D
RETURN
#
```

Eighth program

This program demonstrates the use of the analog inputs.
Use the potentiometers on the GIZMO board to see the output value.

```
10 PRINT "8th program, demo of analog inputs"
20 print "AI1=";AI1,"AI2=";AI2
GOTO 20
#
```

Ninth program

This program demonstrates the use of the analog outputs.
Place Jumper a-b on GIZMO in position a.

```
10 PRINT "9th program, demo of analog output"
20 TM1=31:A=0
30 PRINT "AI2=";AI2,"AO1=";AO1
IF TM1<1 THEN GOSUB 100
GOTO 30

REM SUB ROUTINES
100 TM1=31
A=A+100
IF A>1000 THEN A=0
AO1=A
RETURN
#
```

Tenth program
This program demonstrates sending a **XML** formatted output.

```
REM this is a demonstration program for Row data and XML output
10 REM START OF LOOP
T=AI1/100+20: REM TEMP PROBE SAMPLE ONLY
L=AI2/10: REM 2-40mA SENSOR FULL SCALE = 100%
IF DE1=-1 THEN GOSUB 100
GOTO 10

100 SEND "<!-- File Name: XML_Demo.xml -->"
GOSUB 500
SEND "<?xml-stylesheet type='text/xsl' href='XML_Demo.xsl'?>"
GOSUB 500
SEND "<IO100><DI><DI1>";DI1;"</DI1><DI2>";DI2;"</DI2><DI3>";DI3;"</DI3>"
GOSUB 500
SEND "<DI4>";DI4;"</DI4></DI>"
GOSUB 500
SEND "<AI><AI1>";AI1;"</AI1><AI2>";AI2;"</AI2></AI><AO>";AO1;"</AO>"
GOSUB 500
SEND "<Temperature>";T;"</Temperature><Level>";L;"</Level></IO100>"
RETURN

500 REM WAIT FOR MODEM READY TO SEND NEXT DATA BLOCK
rem TM1=100
510 IF CTSE=1 THEN RETURN: REM TM1=0:RETURN
REM IF TM1=0 THEN GOTO 10: REM MODEM FAULT
GOTO 510
RETURN
#
```

<u>Example of the XML output file</u>

```
<!-- File Name: XML_Demo.xml --><?xml-stylesheet type=text/xsl
href=XML_Demo.xsl?>
<IO100>
  <DI>
    <DI1>0</DI1>
    <DI2>0</DI2>
    <DI3>1</DI3>
    <DI4>1</DI4>
  </DI>
  <AI>
    <AI1>1</AI1>
    <AI2>1</AI2>
  </AI>
  <AO>0</AO>
  <Temperature>20</Temperature>
  <Level>0</Level>
</IO100>
```

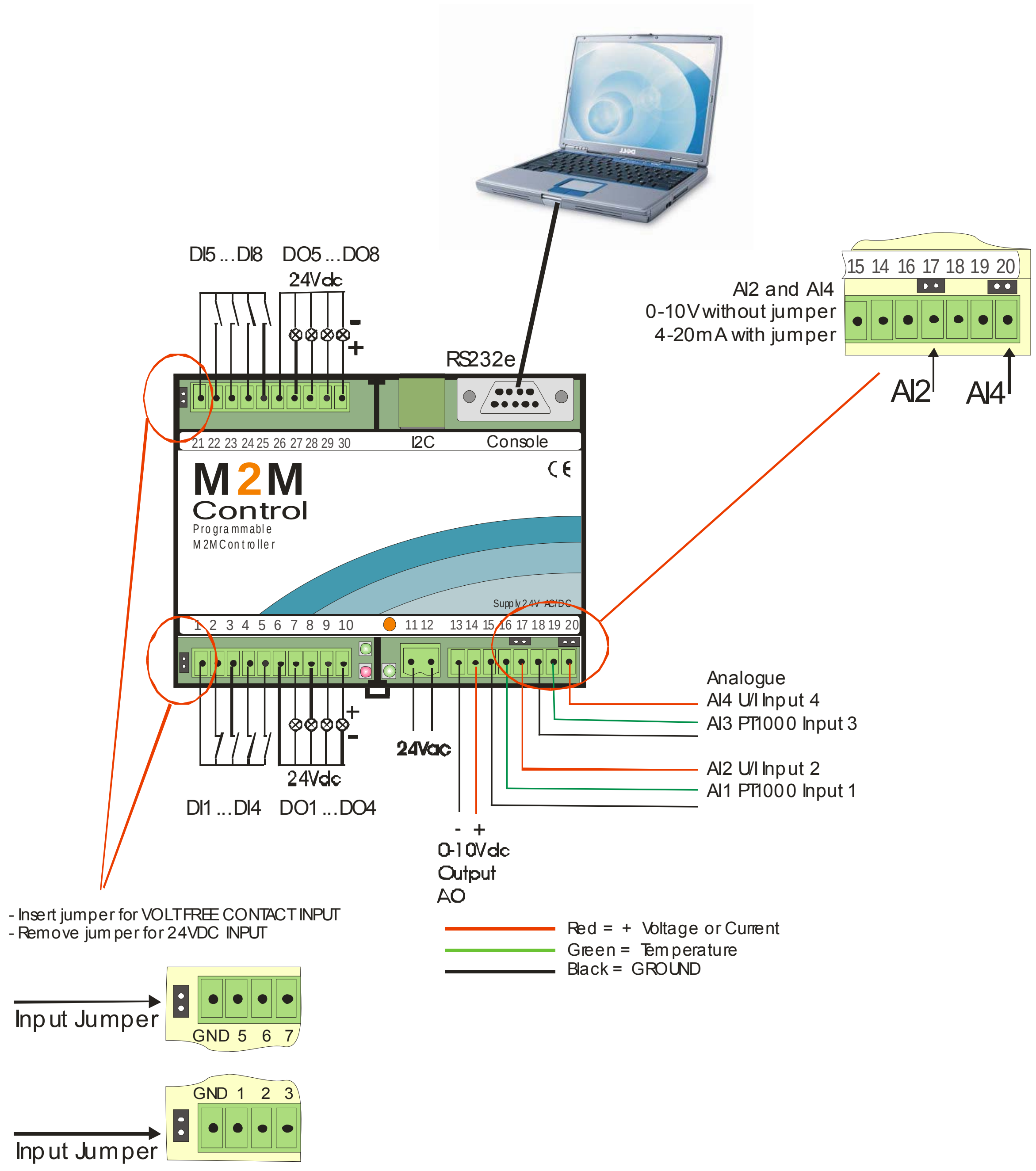
Eleventh program

This program demonstrates the use of the CTM and MRE variables.

```
print "CTM=";ctm, “Clear to mail”
print "MRE=";mre, “Mail result”

rem Wait until modem is ready to send mail
  10 if ctm=0 then goto 10

rem Send mail
  mail 1,"this is a test mail"

rem Wait until modem is finished sending mail
  20 if ctm=0 then goto 20

rem Print the mailed result
print "MRE=";mre


rem endless loop
  30 goto 30
```

10 Schematic diagram

10.1 *M2M Control C100* access & control module



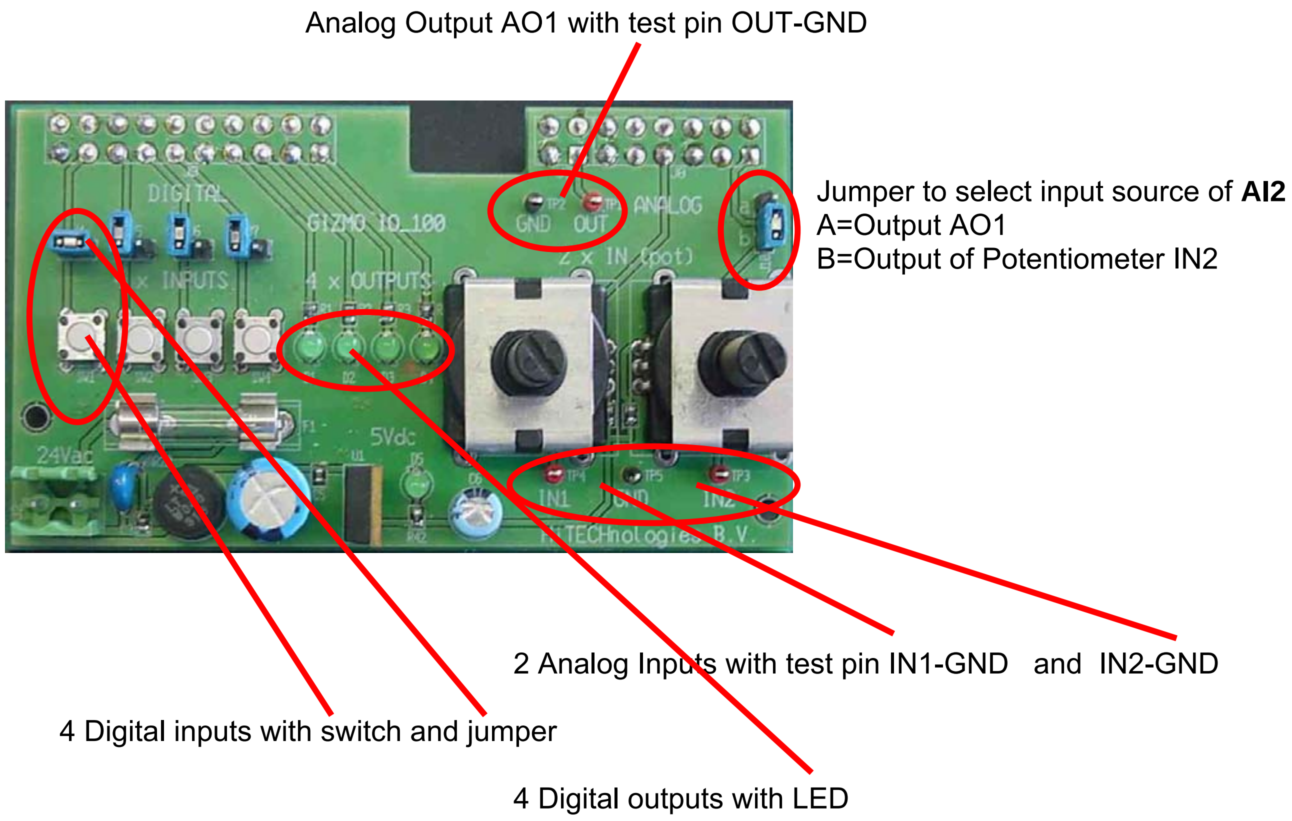
C 100

10.2 GIZMO

DIGITAL I/O

- 4 Inputs with Push button and Jumper
- 4 Outputs

ANALOG I/O
- 2 Inputs with potentiometer for 0-10Vdc
- 1 Output 0-10Vdc


Analog Output AO1 with test pin OUT-GND
Jumper to select input source of **AI2**
A=Output AO1
B=Output of Potentiometer IN2
2 Analog Inputs with test pin IN1-GND and IN2-GND
4 Digital inputs with switch and jumper
4 Digital outputs with LED
DIGITAL
GIZMO IO_100
4 x INPUTS
4 x OUTPUTS
GND
OUT
ANALOG
2 x IN (pot)
5Vdc
24Vac
IN1
GND
IN2
TECHnologies B.V.

11 BASIC Tutorial

11.1 PROGRAMMING IN BASIC

BASIC (Beginners All purpose Symbolic Instruction Code) was created in the 1960's as an easy to learn programming language for computers. Because of BASIC's simple format and because it was an interpreted language that gave the programmer instant feedback, it became the most popular programming language when microcomputers made their debut.
The "*M2M Control C100* Remote Access & Control Module" has an embedded BASIC Interpreter with a limited command set. The *M2M Control C100* does not support string manipulation. The BASIC interpreter has new non standard commands like the instruction to send a SMS message.

Perhaps you found your way here because you're new to programming and would like to start off by learning BASIC. This tutorial will cover important points in detail with sufficient explanation for programming the *M2M Control C100* only. This tutorial introduces the first principles of BASIC, but doesn't provide a thorough description of all language features. For more information on the full language and command set, we advice you to refer to standard teaching books.

A BASIC program is built up line by line out of combinations of the simplest software parts. Once you learn what these software parts are,,how they're used, and with some work and imagination it can take you almost anywhere. Programming is (simply put) the laying out of simple steps to solve a problem, and in a way that an *M2M Control C100* can understand. This is a little bit like teaching a person. These steps must be arranged in the correct order.

Just remember that most programming mistakes are due to programming the wrong sequence, for example you have an instruction to set an OUTPUT to ON and the next instruction sets it (unnoticed) back to OFF, or simply because of spelling mistakes for example: D01 instead of DO1.

The *M2M Control C100* has some simple built in diagnostic help functions, like the **STEP** command. This is a way to go line by line through your program.

11.2 TO START OFF, what is programming anyway?

How do you write a program? Basically,you write commands. These are special keywords (instructions) telling the processor inside the *M2M Control C100* to do something. For example, one of the simplest commands in BASIC is the PRINT statement. It goes like this:
PRINT "Hello!"
When you type this one line program using a text editor, you'll need to store this program with the **LOAD** command in non-volatile memory inside the *M2M Control C100* which can interpret this BASIC program and RUN it. Running a program means executing the commands line by line. In this case there is only one line and you'll see the word ***Hello!*** printed on the terminal screen. We call PRINT a "statement" or "command", PRINT sends out a text on the screen. This is called OUTPUT.

A program in its simplest form usually contains three kinds of activities:

1. INPUT; The program asks information, in our case INPUT is related to Digital and Analog signals;
2. CALCULATION and DECISION MAKING; The program transforms or manipulates the information;
3. OUTPUT; The program sets Digital or Analog OUTPUT signals to a particular value, PRINTs or SENDs an SMS message with the final result.

It is the programmer's job to determine exactly how to accomplish these steps.

11.3 VARIABLES - Doing something with information

In programming, you must assign each bit of data (or information) a unique name. This combination of a name or character is called a variable because the data can vary each time the program uses this variable. To calculate or manipulate Input- or Output-signals the I/O is linked to variables. Therefore the *M2M Control C100* has several predefined variables:

A … Z	as integer, within the range of: **-32768** to **+32767** After Power-Up, value = 0
EEA … EEJ	NON VOLATILE integer, **-32768 ……+32767**
ID	ID of maximum 30 characters READ ONLY, see SET ID for details
DI1 … DI8	for the 8 digital inputs, value is **1** or **0** **(ATTENTION!: read only)**
DE1 … DE4	for digital input EVENT, value = **1** for positive change, **-1** for negative change
DP5 … DP8	for digital input COUNTER, value = **0** to **32767**
DV5 … DV8	for digital input COUNTER, value = **00** to **FF** **Hex variable**
DO1 … DO8	for the 8 digital outputs, value is **1** or **0**
AI1 … AI4	for the 4 analog inputs, value is **0** to **1000** **(ATTENTION!: read only)**
AO1	for the analog output, value is **0** to **1000**
TM1 … TM8	for 100 milliseconds timer **ATTENTION!** if read, this variable returns **0**, **1** or **-1** Timer value can be set to **32767** max. this equals 3276,7 seconds.
SEC	contains the number of seconds since the last minute lapse (0-59)
MIN	contains the number of minutes since the last hour lapse (0-59)
HRS	contains the number of hours since midnight (0-23)
DAY	contains the day of the month (1-31)
WDAY	contains the day of the week (0-6, 0=Sunday)
MTH	contains the month (1-12)
YRS	contains the number of years since 2000 (0-255)

TRE	(Time REsult) is 0 when the *M2M Control C100* thinks it has a valid time. It is 1 at bootup and becomes 0 when the *M2M Control C100* has successfully fetched the time from a time server or when the time is manually configured.

These Date & Time variables can't be written to by the BASIC program. However, they can be set using direct SET commands.

CTSM	Clear To SMs must be 1 before using the SMS command
CTSE	Clear To SEnd to see whether there is a working host link (0=no link available, 1=link available)
CTM	Clear To Mail must be 1 before using the MAIL command
MRE	Mail REsult, equals 0=ok, 1 indicates error, the mail was sent
SMRE	SMs REsult, 0=ok, 1 indicates error, SMS was not sent
SERE	SEnd REsult, 0=ok, 1 indicates error, SEND message was not sent

The data of variables are stored in RAM, this is the volatile Random Access Memory and data is lost after power down. The BASIC program however is stored in FLASH (non-volatile) memory, a program remains in memory after power down.
Configuration and the variables EEA…EEJ are stored in EEPROM (non-volatile) memory, configuration settings and the variables EEA…EEJ remain in memory after power down.

11.4 OPERANDS – Doing some calculations

The *M2M Control C100* interpreter can manipulate variables with the following operands:
(assume variables with the following values: A=? or 2, B=5, C=3, D=10, E=1)

Sign	Name	How to use	Result
+	Plus (ADDITION)	A=B+C	8
-	Hyphen (SUBSTRACTION)	A=B-C	2
*	Asterisk (MULTIPLICATION)	A=B*C	15
^	Circumflex (To the power of)	A=B^C	125
/	Slash (DEVISION)	A=B/C	1
%	Percent (MODULUS)	A=B%C	2
=	Equals sign	A=B	5
>	Greater-than sign	A>B	False
<	Less-than sign	A<B	True
&	Ampersand (AND bit wise)	A&B	0
|	Vertical-line (OR bit wise)	A|B	7

11.5 DELIMITERS – separating data

The interpreter can handle the following delimiters. These are used to separate parts of variables or instructions like:

;	SEMICOLON	used in statement **X** for next variable, PRINT “A=”**;**A Result: ***->A=2***
,	COMMA	used in statement **X** as TAB between variables, PRINT A**,**B**,**C Result: ***->2 5 3***
( )	PARENTHESIS	used in calculations, A = **(**C+B**)*****(**D+E**)**:PRINT A Result: (A=8 x 11 =) ***->88***
“”	QUOTATION MARKS	used in statement **X** for TEXT to be printed, PRINT **“**A=**”**;A
:	COLON	used for following statements, DO1=1:DO2=1:DO3=0

X represents: PRINT, SEND, LOG, MAIL and SMS statements

11.6 AT FIRST – Doing something to start with

Now we have learned something about variables, calculations and delimiters, let’s write a program with some of these items:

```
A=5:B=10:C=30
C=(A+B)*C
PRINT “C=”;C
```

There is an easy way to load a short program, let’s try it!
Type: **LOAD**<enter>
Answer: ***Waiting for program or ESC.***

Type: **A=5:B=10:C=30** <enter>
Type: **C=(A+B)*C**<enter>
Type: **PRINT “C=”;C**<enter>

Type: #<enter>

Answer: ***Program Loaded. Type RUN to start.***

Let's check if you have made any typing errors before running it.

Type: **LIST**<enter>

Answer:

```
A=5:B=10:C=30
C=(A+B)*C
PRINT "C=";C
```

Type: **RUN**<enter>
Answer:

```
->C=450
```

That's all you need to do to load this program and RUN it.

The result is: ***->C=450*** shown on your terminal screen.
This is a quick and easy method, however not suitable for large programs.
ATTENTION!
Since the <Back space> key does not work you can not correct typing mistakes. This is because the *M2M Control C100* program loader expects a correct file transferred in one piece from a computer.

11.7 GOTO - Doing something more than once

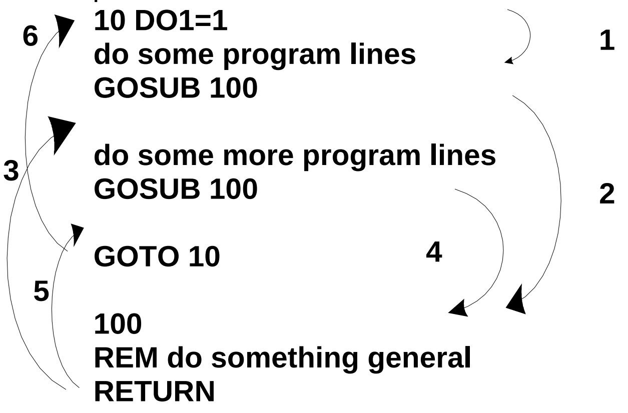
Assuming that the previous BASIC program does what we want, it still only does it once. Each time you want to use this little program you have to RUN it again. This is useless for a monitoring or control function. That's the reason why you will recieve the warning that the program has stopped. What we need is a way for your program to go back to the beginning and do it over, and over again to monitor Input variables and Output results. In BASIC the command for doing this is called GOTO.
Knowing that we have to GOTO some place is not enough. We also need to know where to go. The mechanism that BASIC uses to mark places that we can go to is called a branch label. You mark the place in your BASIC program where you want it to continue running with a branch label using a line number. In principle you can freely use any number, however it makes sense to follow an order, for example: 10, 20, 30, . . . or
100, 200, 300, . . . max 99999. Maximum number of labels = 10.

```
10 DO1=1
 ...
do some more program lines
 ...
GOTO 10
 #
```

ATTENTION!
For control and monitoring applications this mechanism is the most important. You should at all costs prevent your program from getting in an undefined loop. Your program sequence should <u>always</u> continue to guarantee its proper working at all times. It is the programmer's skill that will accomplish this.

11.8 GOSUB - Doing some jump to a program part, do something and return

In the case that you would like to use some general program parts from different parts of your program., you write a sub-routine. Use this from different parts and return to the next statement after the sub-routine called.
Example:



```
10 DO1=1
do some program lines
GOSUB 100

do some more program lines
GOSUB 100

GOTO 10

100
REM do something general
RETURN
```

Of course you can call a sub-routine from a sub-routine; in that case you can nest maximum 10 GOSUBs in GOSUBs.

11.9 IF...THEN…ELSE - Adding intelligence to your program

Your program usually needs to make some decisions at some points. Let's learn how to add some intelligence to your program. One way that this can be done is with the IF . . . THEN statement.
For example:

IF DI1=1 THEN GOSUB 100
This program line lets you make a branch to the sub-routine at line 100 if Input 1 is ON.

Comparing numbers - The = (equal) operator is only one of several that can be used to make decisions in an IF . . . THEN statement. We can use the IF . . . THEN statement and the =, <>, <, > operators to determine whether:
a = b a is equal to b
a <> b a is unequal to b
a < b a is less than b
a > b a is greater than b

As shown in the above example we use the IF . . . THEN statement in the same way for Input variables.
For example:

```
10 DO1=0
IF DI1=1 THEN DO1=1
GOTO 10
#
```

In this case the output DO1 is set to 0 each time the program passes line 10 DO1=0
This will result in flickering, OFF…ON…OFF…ON…etc.

The ELSE statement gives the answer:

```
10 DO1=0
20 IF DI1=1 THEN DO1=1 ELSE DO1=0
GOTO 20
#
```

The result of this program is a stable situation:
When Input 1 is ON then Output 1 is ON
When Input 1 is OFF then Output 1 is OFF
The first line (10) is used as an initialisation phase to guarantee a proper output state after power-up.

ATTENTION: IF A=1 THEN B=1 ELSE C=1:A=10
In this case the statement A=10 is part of the ELSE execution

11.10 IF...AND/OR…THEN…ELSE - Adding more intelligence to your program

To check on one condition is usually not enough, the AND or OR statement gives the answer:

```
10 DO1=0
20 IF DI1=1 AND DI2=1 THEN DO1=1 ELSE DO1=0
GOTO 20
#
```

Now we need to set 2 inputs to the ON position to set the output DO1 to ON.

11.11 More advance programming

For example:

```
10 DO1=0
20 IF DI1=1 THEN DO1=1 ELSE DO1=0
GOTO 20
#
```

Same without IF

```
10 DO1=DI1
GOTO 10
#
```

For example:

```
10 DO1=0
20 IF DI1=1 OR DI2=1 THEN DO1=1 ELSE DO1=0
GOTO 20
#
```

Same without IF

```
10 DO1=DI1 OR DI2
GOTO 10
#
```

For example:

```
10 DO1=0
20 IF DI1=1 OR DI2=1 THEN GOSUB 100
GOTO 20
100 IF DI3=1 THEN DO1=1 ELSE DO1=0
#
```

Same without IF

```
10 DO1=DI3 AND (DI1 OR DI2)
GOTO 10
#
```

11.12 FOR…TO…NEXT - Adding loops to your program

In a for loop, we give a variable a value that it should start with. Then we make it loop by incrementing this variable every time it runs through until it gets to a certain point. An example loop might look like this:

```
FOR I = 1 TO 10
Something to do 10 times
NEXT
```

In case we need to increase a variable with small equal steps, the FOR…NEXT loop is a handy way to do this. Assume we have a dimmer that uses 0-10 Vdc for 0 to 100% lighting level.
We can use the analog output to do the work. You would like to start the lamp smoothly after Input 1 is set to 1 for a short period (pulse), and dim back to off after Input 1 is pulsed to 1 again:

```
10 AO=0: A=0
20 IF DI1=1 AND A=0 THEN GOSUB 100
20 IF DI1=1 AND A=1 THEN GOSUB 200
GOTO 20

100 A=1
FOR  I = 1 TO 100
 AO=I*10
NEXT
RETURN

200 A=0
FOR  J = 1 TO 100
 AO=AO-10
NEXT
RETURN
#
```

This program expects a pulse on Input 1. At first it will change the analog output gradually from 0 to 10 volts. After the next pulse it will gradually decrease the output value from 10 to 0 volts. Of course you can have a FOR…NEXT in a FOR…NEXT, in that case you can nest maximum 10 FOR…NEXT's in a FOR…NEXT. An unwritten rule in BASIC is to use I and J in a FOR…NEXT.

ATTENTION: please note that the FOR…NEXT supports incrementing the loop variable only.

11.13 PRINT, SEND, LOG, MAIL and SMS - Informing the outside world

We talked about the PRINT statement. This is a very useful statement while testing or debugging your program. SEND does the same, not to the console port but to the internal modem.
In other words it sends a PRINT command to some remote connection.
SMS does the same, but sends it out as an SMS to a mobile GSM cell phone.

The format is the same for all three commands.
PRINT "These are the results: A=";A,"B=";B,"C=";C
SEND "These are the results: A=";A,"B=";B,"C=";C
LOG "These are the results: A=";A,"B=";B,"C=";C
MAIL 1,"These are the results: A=";A,"B=";B,"C=";C
SMS 1,"These are the results: A=";A,"B=";B,"C=";C

Finally shown as:

These are the results: ->A=2   B=5   C=3

If you wonder which GSM phone this message will be sent to, the telephone number is a part of the configuration parameters of the IO_100. see SET SMS….

11.14 REM - Adding remarks to your program

We often like to add some remarks to particular statements or program parts. The REM statement is very useful for this purpose.
Like this:

```
REM Dimmer program version 1.0
REM John Steed,  02-02-2003
10 AO=0: A=0
20 IF DI1=1 AND A=0 THEN GOSUB 100
…….etc.
```

ATTENTION!
Do not use to many REM statements in your program since the total available memory is limited to 4 K bytes.

If you would like to comment on your program extensively do this in your source file after the # sign.
Like this:

```
……..
FOR  I = 1 TO 100
AO=AO-10
NEXT I
RETURN
#
The following text explains the above program in detail.
At line 10 we set the analog output to 0 Volt.
The variable A is used as a switch do distinguish between UP en DOWN
Etc.. etc…
```

The # sign is used by the *M2M Control C100* program loader as an end of program indicator. All text after the # sign is ignored and not stored in the program memory.
ATTENTION: while loading the program it can show some parts of the text; this is due to buffers that store blocks of text. Do LIST and you will see that it's OK.

11.15 FINAL - doing it more complex

So far we explained a lot of different statements. A BASIC program will be a logical sequence of statements. It needs no explanation that we can do LOOPs in LOOPs, for example:

```
FOR  I = 1 TO 10
  FOR  J = 1 TO 10
   AO=AO-10
  NEXT
NEXT
```

You can nest maximum 10 LOOPs in LOOPs
Or we can do GOSUBs in GOSUBs, for example:

```
Do something…
GOSUB 100

100
Do something…
GOSUB 200
RETURN
```

You can nest maximum 10 GOSUBs in GOSUBs

ATTENTION!
Be aware not to use too many GOTO statements, it makes your program a mess and no one will understand the real workings.

11.16 COMMAND LINE MODE - doing something outside your program

Well, this was all about BASIC programming.
When you power-up your IO_100, by default it will start to execute the program in memory.
This is needed since the *M2M Control C100* is a monitoring and control module that has to do its job all the time. Just in case it becomes power less it should return to its normal operation mode as soon as the power returns.
However, when you have a Console connected to your *M2M Control C100* the program will NOT START and will wait for your commands.

To stop executing your program type: **STOP** <enter>
The *M2M Control C100* will go into the Command Line mode.
In this mode you have full control over all outputs.

ATTENTION!
Be sure that you know what outputs do before switching something ON!
Use the GIZMO demonstration board for learning.

Please read chapter 7 to understand the “Command line mode” commands in detail.

Since we talked about BASIC programming in this chapter we like to mention some handy "Command line mode" commands to use during programming.

• **LOAD**	a basic program into the non-volatile memory of the module, max. 4096 characters of program text.
• **LIST**	the internal basic program.
• **RUN**	the internal basic program starting at the first line.
• **STOP**	the internal basic program.
• **CONT**	continue execution at the position the program was stopped.
• **HELP**	shows initial instructions.
• **STEP**	step 1 line used for debugging, each executed line is printed on the terminal.
• **RESET**	start from first line.

We wish you happy programming and advice you to use the GIZMO board as an easy I/O interface with a fast start. Many demonstration programs are available to give you a better understanding of the *M2M Control C100* and the BASIC interpreter.
Good Luck!

- / -

12 APPENDIX

12.1 GSM Modem states

In order to see what the modem is doing the user can turn ON the verbose mode type:
MODEMVERBOSE 1
This will result in modem state numbers on the console port. Below a list of the state numbers:

Because of a review of the library handling the modem communication there has been a change in the states and substates reported by the M2M Control module when MODEMVERBOSE is set.

<u>Modem states</u>

0:	STATE_INIT	- The M2M Control is initializing the modem
1:	STATE_IDLE	- The modem is initialized and the module is able to send SMS's and to receive incomming calls
2:	STATE_DIAL	- Not used yet (for dialing out)
3:	STATE_DATA_OUT	- Not used yet (dail out connection established)
4:	STATE_ANSWER	- Anwering an incomming call
5:	STATE_DATA_IN	- Incomming call established
6:	STATE_DIAL_TCP	- Not used yet (for setting up TCP connection)
7:	STATE_TCP_OUT	- TCP out connection established
8:	STATE_DISCONNECT	- Remote disconnected, or problems trying to connect
9:	STATE_SPECIALS_START	- Not used as real state, only as reference
10:	STATE_OPERATOR	- Requesting the GMS operator
11:	STATE_RSSI	- Requesting the signal strength

<u>Modem substates</u>

0:	SUBSTATE_INIT_NO_MODEM	- No modem detected (default startup state)
1:	SUBSTATE_INIT_TELIT_MODEM	- Check if modem is ON
2:	SUBSTATE_INIT_TELIT_PULSE_WAIT	- Wait for modem to switch ON
3:	SUBSTATE_INIT_TELIT_SEARCH	- Search the modem
4:	SUBSTATE_INIT_MODEM_DETECTED	- A modem is detected
5:	SUBSTATE_INIT_ESCAPE1	- Delay for escape command
6:	SUBSTATE_INIT_ESCAPE2	- Send escape command
7:	SUBSTATE_INIT_STRING	- Send the initialization string
8:	SUBSTATE_INIT_PIN_GET	- Request PIN status
9:	SUBSTATE_INIT_EXPECT_PIN_GET	- Wait for PIN_GET response
10:	SUBSTATE_INIT_SIM_PIN	- SIM expects PIN
11:	SUBSTATE_INIT_EXPECT_PIN_RESULT	- PIN sent, wait for response
12:	SUBSTATE_INIT_SIM_ERROR	- Problems with SIM: not there, defect
13:	SUBSTATE_INIT_PIN_ERROR	- Error, probably wrong PIN
14:	SUBSTATE_INIT_SIM_PUK	- SIM needs PUK code (put SIM in cellphone to handle this, can not be done out of M2M Control)
15:	SUBSTATE_INIT_SIM_OK	- PIN accepted
16:	SUBSTATE_INIT_CHECK_SIM_READY	- Wait until the SIM is ready to communi-cate with (called repeatedly at powerup)
17:	SUBSTATE_INIT_CHECK_NETWORK_READY	- Check if SIM is registered to an operator.

18: SUBSTATE_INIT_SET_FORMAT	- Set SMS format to text
19: SUBSTATE_INIT_WAIT_FORMAT_OK	- Wait for SMS format accepted
20: SUBSTATE_INIT_SET_SMS_INDICATION	- Let modem indicate the reception of a new SMS
21: SUBSTATE_INIT_SMS_INDICATION_OK	- Wait for acceptence of SMS indication
22: SUBSTATE_INIT_TX_BUF_FULL	- Sending command to modem not possible
23: SUBSTATE_INIT_TIMER_EXPIRED	- Answer from modem took too long
24..38: Free space for future initialization states	
39: SUBSTATE_IDLE	- Modem ready to send and receive
40: SUBSTATE_SMS_WAIT_SMS_PROMPT	- Wait for SMS prompt
41: SUBSTATE_SMS_SEND	- Sending an SMS
42: SUBSTATE_SMS_WAIT_OK	- Wait for sending SMS OK
43: SUBSTATE_SMS_TX_BUF_FULL	- Sending SMS to modem not possible
44: SUBSTATE_SMS_TIMER_EXPIRED	- Answer from modem took too long (possible GPRS only SIM)
45: SUBSTATE_SMS_WAIT_BUFFER_EMPTY	- Wait until all data to modem is sent
46: SUBSTATE_SMS_ESCAPE1	- Delay for escape command
47: SUBSTATE_SMS_ESCAPE2	- Send escape command
48: SUBSTATE_SMS_ESCAPED	- Switched from data to text mode
49: SUBSTATE_SMS_RETURN_DATA	- Return from text to data mode
50: SUBSTATE_SMS_WAIT_RETURN_OK	- Wait for returning to data mode OK
51: SUBSTATE_SMS_EXPECT_SMS	- Expect SMS from modem
52: SUBSTATE_SMS_GET_DATE	- Retreive relevant data from SMS
53: SUBSTATE_SMS_DEL_SMS	- Delete the SMS from the SIM
54..69 Free space for future SMS states	
70: SUBSTATE_WAIT_CONNECT	- Wait for the CONNECT string to receive
71: SUBSTATE_WAIT_NO_CARRIER	- Wait for NO CARRIER text while disconnecting
72..79 Free space for dial-in/dial-out states	
80: SUBSTATE_OPERATOR	- Request the operator
81: SUBSTATE_EXPECT_OPERATOR	- Expect the operator
82: SUBSTATE_OPERATOR_TIMER_EXPIRED	- Answer from modem took too long
83: SUBSTATE_RSSI	- Request the signal strength
84: SUBSTATE_EXPECT_RSSI	- Expect the signal strength
85: SUBSTATE_RSSI_TIMER_EXPIRED	- Answer from modem took too long
86..99 Free space for future special states	
100: SUBSTATE_GPRS_SET_CONTEXT	- Set the GPRS context parameters
101: SUBSTATE_GPRS_SET_SMTP	- Set the SMTP(mail)server for sending emails
102: SUBSTATE_GPRS_SET_MAILUSER	- Set username for mailserver
103: SUBSTATE_GPRS_SET_MAILPASS	- Set password for mailserver
104: SUBSTATE_GPRS_SET_FROM	- Set sender of the mail
105: SUBSTATE_GPRS_SET_USER	- Set the username for if connecting to the GPRS network requires one
106: SUBSTATE_GPRS_SET_PASS	- Set the corresponding password for if connecting to the GPRS network requires one
107: SUBSTATE_GPRS_SET_SOCKET	- Set the remote socket to connect to

108: SUBSTATE_GPRS_OPEN_SOCKET	- Open the socket, that is, make a connection to the configured host on the configured port
109: SUBSTATE_GPRS_WAIT_CONNECT	- Wait for the connection to be established
110: SUBSTATE_GPRS_DELAY_CONNECT	- Delay a short time before reconnecting. This requires the modem

111..129 Free space for future GPRS initialisation states

130(-126): SUBSTATE_GPRS_SEND_EMAIL	- Send an email
131(-125): SUBSTATE_GPRS_WAIT_MAIL_PROMPT	- Wait for the mail prompt
132(-124): SUBSTATE_GPRS_WAIT_MAIL_OK	- Wait for OK after sending the email
133(-123): SUBSTATE_GPRS_MAIL_TIMER_EXPIRED	- Answer from modem took too long
134(-122): SUBSTATE_GPRS_MAIL_WAIT_BUFFER_EMPTY	- Wait for an empty serial buffer before closing the GPRS link
135(-121): SUBSTATE_GPRS_MAIL_DELAY	- Delay a required time before sending the connection close command
136(-120): SUBSTATE_GPRS_MAIL_CLOSE	- Close the connection
137(-119): SUBSTATE_GPRS_MAIL_CLOSED	- Connection closed for sending mail. This state indicates that after sending the mail, the GPRS connecion must be re-established

()The negative values are printed as well, because the first versions of the M2M Control suffered a signed/unsigned bug for printing the states.

SMS states

0:	HTMODEM_SMS_STATE_INIT	- The M2M Control is initializing the modem
1:	HTMODEM_SMS_STATE_IDLE	- Modem ready to send SMS's
2:	HTMODEM_SMS_STATE_WAIT_SEND	- SMS ready to be sent
3:	HTMODEM_SMS_STATE_WAIT_RECEIVE	- Request for reading SMS pending
4	HTMODEM_SMS_STATE_WAIT_POLL	- Not used yet (for polling incomming SMS's)
5:	HTMODEM_SMS_STATE_ESCAPE_SEND	- Escape from data to text mode for sending SMS
6:	HTMODEM_SMS_STATE_ESCAPE_POLL	- Not used yet (for escaping to poll incomming SMS's)
7:	HTMODEM_SMS_STATE_SEND	- Modem is sending an SMS
8:	HTMODEM_SMS_STATE_RECEIVE	- Read SMS from SIM
9:	HTMODEM_SMS_STATE_POLL	- Not used yet (For modem is polling incomming SMS's)
10:	HTMODEM_SMS_STATE_RETURN_DATA	- Return from text to data mode

Email states

0: HTMODEM_MAIL_STATE_INIT	- The M2M Control is initializing the modem
1: HTMODEM_MAIL_STATE_IDLE	- Modem ready to send emails
2: HTMODEM_MAIL_STATE_WAIT_SEND	- Email ready to be sent
3: HTMODEM_MAIL_STATE_ESCAPE_SEND	- Close GPRS link before sending email
4: HTMODEM_MAIL_STATE_DISCONNECT	- Close GPRS link
5: HTMODEM_MAIL_STATE_SEND	- Send the email comming from IDLE state
6: HTMODEM_MAIL_STATE_SEND_CLOSED	- Send the email comming from TCP_OUT state

Turn OFF the verbose mode type: **MODEMVERBOSE 0**

12.2 MOBITEX Modem states

Link Level States **LINKVERBOSE 1**

1	LLS_INITIAL,	- Initial state, wait for dsr
2	LLS_START,	- Start link, wait for link establised
3	LLS_IDLE,	- Idle link
4	LLS_SEND,	- Send, wait for ack
5	LLS_RACK	- Rack, wait for ack after sending rack

Link Level Events

1	LLE_STARTUP,	- Done in first control level call
2	LLE_TERMINATE,	- Close connection
3	LLE_SEND,	- If frame_outgoing is YES
4	LLE_SENS,	- Incoming SENS frame
5	LLE_SACK,	- Incoming SACK frame
6	LLE_ACK,	- Incoming ACK frame
7	LLE_NACK,	- Incoming NACK frame
8	LLE_RACK,	- Incoming RACK frame
9	LLE_I_FRAME,	- Incoming INFO frame
10	LLE_ACK_TIMEOUT,	- The ACKnowledgement timer timed out
11	LLE_SENS_TIMEOUT,	- The SENS timer timed out
12	LLE_DSR_TIMEOUT,	- The INIT timer timed out
13	LLE_DSR_DOWN,	- Dsr down
14	LLE_DSR_UP,	- Dsr up
15	LLE_NONE,	- NO event occured
16	LLE_UNKNOWN	- Unknown incoming frame

Masc level states **MASKVERBOSE 1**

0:	MLS_INITIAL	- Initial state
1:	MLS_STARTUP	- Starting up the link
2:	MLS_INIT_MODEM	- Initialising the modem
3:	MLS_RESET	- Resetting the link (happens on all initialisation and link problems)
4:	MLS_AWAIT_MAN	- Wait for the local Mobitex Access Number
5:	MLS_AWAIT_TYPE	- Wait for Masc device information
6:	MLS_IDLE	- The modem is initialized and the link is up
7:	MLS_SEND_MPAK	- Send a Mobitex PAcKet
8:	MLS_AWAIT_TRANSMIT	- Wait for transmitting is possible
9:	MLS_TERMINATE	- Terminate the current connection
10:	MLS_NO_COVERAGE	- No network coverage (check antenna location and position)
11:	MLS_TERM_PENDING1	- Terminating state on normal operation
12:	MLS_TERM_PENDING2	- Terminating state when MAN or TYPE is expected
13:	MLS_CANCEL_MPAK1	- Sending MPAK timed out
14:	MLS_CANCEL_MPAK2	- Cancel sending because out of coverage
15:	MLS_CANCEL_PENDING	- Detection of no coverage (start handle it)
16:	MLS_TERM_PENDING3	- Handle abnormal termination (timeout/ out of coverage)
17:	MLS_SEND_FRAME	- Frame sent
18:	MLS_SEND_FRAME2	- Frame being sent
19:	MLS_SEND_FRAME3	- Handle sending a frame when no coverage

Masc level events

0:	MLE_STARTUP	- Starting up the MASC layer
1:	MLE_SEND_MPAK	- Sending an MPAK (Mobitex PAcKet)
2:	MLE_SEND_FRAME	- Sending a (link)control frame
3:	MLE_TERMINATE	- Terminate the current connection
4:	MLE_LINK_UP	- Mobitex link is up
5:	MLE_LINK_DOWN	- Mobitex link is down
6:	MLE_FRAME_SENT	- A frame is send
7:	MLE_FRAME_FAILED	- Sending a frame failed
8:	MLE_B_FRAME	- B-frame received (including parameters for MASC protocol)
9:	MLE_FA_FRAME	- FA-frame received (including RSSI value)
10:	MLE_FF_FRAME	- FF-frame received (modem in contact with network)
11:	MLE_FG_FRAME	- FG-frame received (modem lost contact with network)
12:	MLE_FH_FRAME	- FH-frame received (MPAK sent to network)
13:	MLE_FK_FRAME	- FK-frame received (error message from modem received)
14:	MLE_FO_FRAME	- FO-frame received (network contact shut down)
15:	MLE_FP_FRAME	- FP- frame received (message containing the modem MAN)
16:	MLE_FQ_FRAME	- FQ-frame received (special message on initialisation)
17:	MLE_M_FRAME	- M-frame received (Mobitex PAcKet received)
18:	MLE_E_FRAME	- E-frame received (sent command can not be executed)
19:	MLE_N_FRAME	- N-frame received (returns an MPAK that could not be sent)
20:	MLE_R_FRAME	- R-frame received (erroneous MPAK returned)
21:	MLE_PO05_FRAME	- P-frame received (contains roaming parameters; not used)
22:	MLE_RESET_TIMEOUT	- Reset timer expired, retry to connect to network
23:	MLE_MPAK_TIMEOUT	- Trying to send an MPAK timed out
24:	MLE_OTHER_FRAME	- An undefined frame was received (for statistics only)
25:	MLE_TIME	- A time packet received
26:	MLE_FLEXLIST	- Flexlist received (not used)
27:	MLE_GROUPLIST	- Grouplist received (not used)

Ram statusses

0:	MNC_MASC_UP	- The MASC protocol layer is up
1:	MNC_MASC_DOWN	- The MASC protocol layer is down
2:	MNC_IN_COVERAGE	- Modem is in coverage
3:	MNC_OUT_OF_COVERAGE	- Modem is out of coverage
4:	MNC_MPAK_SENT	- An MPAK (Mobitex PacKet) is sent succesfully
5:	MNC_MPAK_FAILED	- Sending an MPAK failed
6:	MNC_MPAK_INACCURATE	- A message to be sent has an invalid format
7:	MNC_MPAK_RECEIVED	- An MPAK has been received
8:	MNC_MPAK_RETURNED	- A message to be sent has been returned
9:	MNC_TIME	- A time message received
10:	MNC_FLEXLIST	- A flexlist has been received (not used)
11:	MNC_GROUPLIST	- A grouplist has been received (not used)
12:	MNC_FRAME_RECEIVED	- An error message or unknow message has been received
13:	MNC_FRAME_SENT	- A frame has been sent succesfully
14:	MNC_FRAME_FAILED	- A frame could not be sent
15:	MNC_INFO_MODE_REPLY	- Power save mode reply (not used)
16:	MNC_INFO_BATT_REPLY	- Battery information reply (not used)
17:	MNC_INFO_RSSI_REPLY	- RSSI value reply
18:	MNC_INFO_BASE_REPLY	- Info with connected base information repy (not used)

Ram traffic states

0:	MPAK_OK	- A correct MPAK (Mobitex PacKet) has been received
1:	MPAK_FROM_MAIL	- An MPAK buffered by the network has been received
2:	MPAK_IN_MAIL	- An MPAK could not be delivered and is buffered by the network
3:	MPAK_NO_TRANSFER	- An MPAK could not be delivered because the emote side is not registered to the network

4:	MPAK_ILLEGAL	- An MPAK could not be delivered because sender and addressee are not in the same closed user group
5:	MPAK_CONGEST	- An MPAK could not be delivered because the line or radio channels are congested
6:	MPAK_ERROR	- An MPAK could not be delivered because there is a technical error on the network

Exception MPAK return code

1:	MPAK_RC_FROM_MAIL	- An MPAK buffered by the network has been received
2:	MPAK_RC_IN_MAIL	- An MPAK could not be delivered and is buffered by the network
3:	MPAK_RC_NO_TRANSFER	- An MPAK could not be delivered because the emote side is not registered to the network
4:	MPAK_RC_ILLEGAL	- An MPAK could not be delivered because sender and addressee are not in the same closed user group
5:	MPAK_RC_CONGEST	- An MPAK could not be delivered because the line or radio channels are congested
6:	MPAK_RC_ERROR	- An MPAK could not be delivered because there is a technical error on the network
7:	MPAK_RC_BUSY	- Not applicable (from datasheets)
8:	MPAK_RC_MPAK_FAILED	- Sending an MPAK failed
9:	MPAK_RC_MPAK_INACCURATE	- A message to be sent has an invalid format
10:	MPAK_RC_MPAK_RETURNED	- A message to be sent has been returned

12.3 Configuration HINTs

For use of GPRS
General settings:
- SET APN, like "Internet"
- SET TCPSUSER [1])
- SET TCPSPASS [1])
- TCP ON [2])

[1]) Use when specified by Telco for authentication

For use of E-MAIL function only
- General settings
- SET MAILUSER [3])
- SET MAILPASS [3])
- SET SMTP [4])
- SET HOST = 0 [5])
- SET PORT = 0 [5])

[2]) Only if you use SENT command (link with HOST)
[3]) Check with Telco or Mail Server owner for details
[4]) Be sure to use the correct name e.g.: In Germany not smtp.vodafone.de but authsmtp.vodafone.de
[5]) One or both should be 0

Please check the CTM (Clear To Mail) variable for 1 before sending an e-mail message.
In case your mail does not get through we advise to fill the FROM field with a ....@SMTP Server domain name. e.g.: M2M Control@vodafone.de
This is because some mail servers use either User/Password or check domain extension.

For use of GPRS LINK
- General settings
- SET HOST = IP address or URL of HOST
- SET PORT = Port address of port used on HOST

The GPRS link can only be used when suitable software is running on the specified host listening to the specified port. (M2M Technologies will release sample software (VB6 source) soon)

For use of E-MAIL function + GPRS LINK
- General settings
- SET MAILUSER
- SET MAILPASS
- SET SMTP
- SET HOST = IP address or URL of HOST
- SET PORT = Port address of port used on HOST

M2M Control **is a trade name of**

Infranet Technologies GmbH
Tempowerkring 2
21079 Hamburg - Germany
tel. +49 (0)40 696 47 – 260 fax +49 (0)40 696 47 – 259

E-Mail: info@m2mcontrol.de Web: www.m2mcontrol.de