MIKROELEKTRONIK GMBH

Intelligente Sensoren & Mess-Systeme

M2-iLAN Laser-Scanner

M2D Laser Scanner
with integrated Ethernet-Interface
TCP/UDP

User Manual

MEL Mikroelektronik GmbH
Breslauer Str. 2
D-85386 Eching
Tel. +49 89 / 327 150-0
Fax +49 89 / 319 20 23

www.MELsensor.de

March 2010

Contents

Firmware-Version

Hardware-Revision	4.0	with M12 connectors
FPGA Firmware-Revision	4.6	
Controller Firmware Revision	2.0.59	

This manual describes the installation of the scanner hardware and the use of MEL demo and calibration software. The manual has been changed to the new version of hardware.

With the transit to the revision 4 hardware, the electronic system has been equipped with M12 connectors. The scanner head is connected to the network with the standard Ethernet M12 connector. The power and control connector is an M12 connector with 8 pins.

📖 *Tipp:* *default values should be read from the scanner.*
after a reset, the scanner restores all values to default.

to read out the default values:
1. send reset command 0x1E
2. request info telegram 0x21

default values could be different from given values in this manual.
relevant is the actual interface description available from MEL.
when you read out default values from the scanner, these may be the almost recent values.

📖 *Laser Safety*

Please refer to the Chapter "Laser Safety and Maintenance" to find recommendations and guidelines *for use of controls, adjustments of performance and recommended practice to avoid any hazardous radiation exposure (page 43). Use of controls or adjustments or performance of procedures other than those specified herein may result in hazardous radiation exposure.*

System description

With the new M2-iLAN „Ethernet"-Scanner the electronic system is integrated in the scanner head. The system has no extra controller box, so it is highly compact, tight and rugged. The scanner head has two M12 connectors for *Ethernet* and *power supply & controls*. Scanner heads designed for welding applications additionally have *water & air cooling fittings*.

M2W-iL-models are rated up to 500°C (950° F) environment temperature with water cooling, up to 150°C (240°F) with air cooling, dry and clean atmosphere, non corrosive.

Product label

The product label specifies the type of scanner head, the range and standoff distance and the serial number with production date (YYYY.MM.XXX) (XXX= Serial number).

Scanner head and Electronic System

The scanner head has a CCD camera, a line laser projector and an electronic system for digitizing the camera signals. The electronic system sends the camera signal to the electronic system. The camera is controlled by the automatic control through the electronic system or in external mode by the application software. Adjustment for shutter time, video gain and laser intensity are made through either automatic or under "manual" control by the application software.

Connections

Ethernet Hardware

Standard Ethernet components never have shown problems in the data transmission of the Ethernet scanners. Network connection Rx+, Rx-, Tx+, Tx- is available on a 4-pin M12-connector D-coded according to standards. The scanner should be connected with shielded Ethernet cables CAT 5e or better. Total cable length for Ethernet cabling is 200m according to Standard IEE 802.3

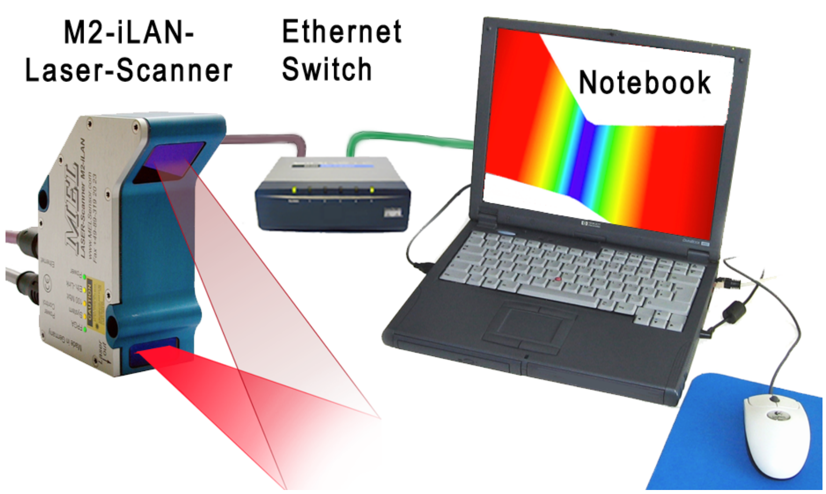
We also use Ethernet scanners in the company network without any effect on the daily office work running through the same network. Use Ethernet switches, hubs are not recommended. When several scanners are monitored by one PC, the graphic card and CPU must be strong enough to collect all the data provided by the scanners. When you doubt, that the Scanner does not deliver all images, you should check the image counter (page 15: register 0x11, status messages.) The image counter number allows sorting images, if they arrive in wrong order. This may happen accidentally when transmitting profile data over the internet.

Ethernet cabling

The Ethernet network connection is made with M12 connectors, 4 pin D coded. The control interface is connected through an 8 pin M12 connector, the pin details are given on page 6.

Ethernet cables should be double shielded. The cables should be according specification CAT 5e or better.

When the Scanner is connected to the network over a Ethernet switch, normal (1:1 wired) patch cable may be used. When you want to make a direct connection from the Scanner to the PC use a cross-wired cable.



Picture at the right side:
Connecting a scanner with switch and PC.

Multi-Scanner Arrays

Up to 32 Scanners could be integrated in a network. Normally, you should use a separate network card for the connections to the scanner(s). The use of a separate network card is optional and depends from your network settings, network traffic on the dedicated PC and system load. You may use a Ethernet switch, a Ethernet cable to each scanner and power supply 10 … 28 V DC.

How many Scanners can work in one network segment?

The Scanner sends 290 Profile points per Scan with 93.5 Hz :

- TCP-IP Header
- M2DF-LAN-Header
- Profile data
- Status information
- Reserved Bytes for additional data
- additional Information as Encoder-Data, FiFO
- CRC

Each Profile creates a 2048 Byte Block = 93.4 x 2.048 kBytes / s = 191.28 kByte / s.

The network channel has 100 Mbit = 12.500 kBytes / sec
This will allow theoretically maximum = 12.500 / 191 = 65.4 Scanners.

The numbers of collisions in a network rises, when the network load exceeds 50%. Therefore as a realistic figure, we assume 50 % of maximum transmission rate.
This leads to the number of **32 Scanners** for maximum usage in one network segment**.**

Segmentation of data blocks

Limitations of the network cards like setting of the MTU (Maximum Transfer Unit), and accordingly load in the network causes that the 2048 Bytes will not be transferred in one packet, but segmented into two packets of 1460 k and 588 k. The user must not think about this effect, though the TCP implementation of the operating systems cares about sorting and reconciling the TCP packets. More likely, the software engineer designing an application must know this and wait long enough to make sure that all packets have been received, before using the data.

Factory default IP address

At the end of the manufacturing test the scanners are set to:
IP address 192.168.123.245
Subnet mask 255.255.255.0.

Before you integrate these units in your network, you should set the network address to a working address which will operate in your network properly. For more technical details and background information see the network tutorial. The set up procedure is given on page 24.

Scanner heads

The scanner head power supply is connected with a 8 pin M12 connector. The control cable has power supply, sync signals, digital inputs for encoders and control of firmware upload and RS-232 diagnose interface. The pin definition is given on the next page.

The Ethernet connection is made with a 4 pin M12 connector according to standard. The Ethernet cable connectors are male.
The connector for the control cable is male. The *controls & power cable* has a female connector and at the other end is either with open leads for clamps or with a male connector.

Connectors

Ethernet cable

	Pin-Nr.	Signal	Colors ***	Remarks
Ethernet M12 round 4-pin D-coded female	1	Tx+	Green+white	Transmit data Ethernet +
	2	Rx+	Red+white	Receive data Ethernet +
	3	Tx-	Green	Transmit data Ethernet -
	4	Rx-	Red	Receive data Ethernet -
	Shield			Connect to case!

Control cable

	Pin-Nr.	Signal	Colors***	Remarks
M12 round 8-pin A-coded male	1	+ 24 V DC	White	Supply for Electronic box
	2	Digital input 1	Brown	Encoder and command input 1*
	3	GND	Green	Ground
	4	Digital input 2	Yellow	Encoder and command input 2 **
	5	Sync out	Grey	Sync signal output (Master)
	6	Sync in	Orange	Sync signal input (Slave); Hardware-Trigger
	7	TxD	Blue	Rs-232 Diagnosis out
	8	RxD	Red	RS-232 Programming data input
	Shield			Connect to pin 3 (ground)

* when connecting the sync output to the command input 1, at start of the sensor (power on), the sensor is brought to the *programming* mode. In this programming mode the firmware can be uploaded with the RS-232 connection or over the Ethernet network.

** when the sync output is connected to the command input 2, at start of the sensor, the sensor will activate the RiP mode and communicate only at the default IP address.
Please note: when both command inputs were connected to sync out, the sensor will go to the programming mode.

📖 The activation of the *Programming mode* or *RiP mode* is made only when the sensor is switched on and had no power before for a few 10 seconds.

*** colors are given as an example.

📖 *Tipp: the Scanner could be connected with <u>one single</u>*, highly flexible, cable for robotic applications. The cable used is a Ethernet cable specially made for the use with robots. The power supply is taken over two unused wires in the network cable, following the idea of the IEEE 802.3af standard. Details are given in the TE-M2-iLAN-connection-E.pdf. Cables following this definition are available from MEL on request.*

** single cable attachment: connection with a single cable specified for use with robots.*
*** PoE: Power over Ethernet, standard IEEE 802.3af.*

Status LED's

Status LED's	Meaning	color***	OK when
Power	Power OK	Green	LED is lit
Eth-Link	Ethernet Link in Function	Yellow	Blinking
100 Mbit	Ethernet Link Activity	Orange	Blinking
System	Hardware Self test OK	Orange	Blinking
FPGA	FPGA Self test OK	Orange	LED is lit

RJ-45 connector

Pin	Signal	Connector A		Pin	"Crossed" B	Signal
1	Transmit data +	Green +White	Colors are given as an example.	1	Red + White	Receive data +
2	Transmit data -	Green		2	Red	Receive data -
3	Receive data +	Red + White		3	Green +White	Transmit data +
4	Not used -	Blue	Cables could have different colors.	4	Blue	Not used -
5	Not used +	Blue + White		5	Blue + White	Not used +
6	Receive data -	Red		6	Green	Transmit data -
7	Not used +	Brown+ White		7	Brown+ White	Not used +
8	Not used -	Brown		8	Brown	Not used -

📖 *Tipp: the M12-Ethernet cable has male connectors at both ends. The M12-control cable has either a female connector at the electronic box and a male connector or open wires at the other end.*

Special accessories

Nr	Description	MEL Part name	MEL order number
1	M2-RS232-Programming Adapter Box Adapter to RS-232 interface, *for Firmware-Upload, power switch and default-IP- Address*. Requires + 24V power supply!	M2-RS232-ProgBox	i.550.006
2	Power supply + 24V, for M2-RS232-Box (Pos. 1)	Power supply 24V/250 mA	i.550.007
3	Connecting cable M12-8 pin; 2.0 m; free leads	Ctrl-Cbl-W-M12-o	i.550.008
4	Connecting cable M12-8 pin female socket – 8 pin, 0.4 m	SKK 215	i.550.009
5	Connection cable M12-4 pin, D-coded connector – 4 pin D-coded connectors, Length 0.4 m; PTFE	SKK 216	i.550.010
6	Adapter M12 – RJ45 straight, with thread M 16x1.5	Eth-Adapter-G-M12	i.550.011
7	Adapter M12 – RJ45 at angle 90°, with thread M 16x1.5	Eth-Adapter-W-M12	i.550.012
8	Connector M12-4pin-D coded; straight, auto-clamps	Eth-Con-D-G-M12	i.550.016
9	Connector M12-4pin-D coded; 90° angle, screw on clamps	Eth-Con-D-W-M12	i.550.018
10	Connecting cable M12 male – 4 pin; RJ-45; 1 m	Eth-Cbl-M-M12-RJ45	i.550.026
11	Connecting cable M12 female – 8 pin at angle 90° - free leads	Ctrl-Cbl-8pWW-o	i.550.027
12	Adapter for connecting two RJ45 network cables	Adaptor-2N-RJ45	i.550.028
13	Network cable RJ-45, 5m 1:1 (patch)	Network cable RJ45	i.550.029
14	Cross wired network cable RJ-45, 5 m	Crosslink cable RJ45	i.550.030
15	Connector M12 - 4 pin D-coded with *screw on clamps*	Con-M12 4 pin-m-scr	i.550.031
16	Cable receptacle 8 pin female with *screw on clamps*	Con-M12 8 pin-w-scr	i.550.032
17	Water / air fittings (spare parts)	WaAir-Stub	i.550.033
18	Ethernet cable; CAT5e, 4-pin, AWG24 flexible; PUR chloride free blue, shielded, straight connector, M12, D-coded – RJ45, 2m	Eth-Cbl-M12-RJ45-2m	i.550.041
19	Ethernet cable; CAT5e, 4-pin, AWG24 flexible; PUR chloride free blue, shielded, straight connector, M12, D-coded – RJ45, 5m	Eth-Cbl-M12-RJ45-5m	i.550.042
20	Ethernet cable; CAT5e, 4-pin, AWG24 flexible; PUR chloride free blue, shielded, straight connector, M12, D-coded – RJ45 – open leads, length = 15m	Eth-Cbl-M12-o-15m	i.550.040
21	Connecting cable M12-4pin crossed Ethernet cable with RJ-45 *Cable length customer specific*	EthRJ45-X-Cbl-M12-custom	i.550.043

↑ M2-RS232-ProgBox

M2-W-iLAN-80/40/55 →
Scanner head with air / water cooling option

Technical Data

Profiles update rate	max. 93.5 Hz	
Profile rate, Trigger-Mode	approx. 60 Hz	
Profile rate, external synchronized	max. 93.5 Hz	
Stability of time base	100×10^{-6}	
Min. / Max. Environment temperature	min. 0°C (+ 32°F)	max. 40°C (+ 95°F)
Min. / Max. Storage temperature	min. –30°C (+ 2°F)	max. 75°C (+ 167°F)

Integrated electronic system

Supply current	120 mA at 24 V; Laser on
Range of supply voltage	+ 10 ... 30 V DC
Digital inputs	Low = 0 ... 2 V High = 5 ... 30 V
Weight	300 g
Dimensions	102 mm x 74 mm x 27,5 mm
Mounting	holes for 4 x M5 screw
connections	12 pin round female for Scanner head M12 - 8 Pin male supply and controls Synchronization and external Trigger M12 4 Pin female = Ethernet D coded RJ-45-Ethernet internal for service
Protection class	IP 65

Vibration: the electronic unit has been specially protected against shock and vibration.

Encoder signal connection
Encoder-Data is received at the digital inputs at the control connector. The encoder information is read and transmitted embedded in the profile information over TCP Ethernet connection. The encoder data is always in sync to the profile relating to this encoder data.

RS-232 interface for diagnosis and firmware update

Baud rate: Monitoring	115. 200 Baud
Baud rate: Firmware-Update	57.600 Baud

The RS 232 interface does not carry profile data or scanner control information.

Software: the current release of the MEL „EthernerScanner.exe“ Demo- and configuration software is available as download from MEL FTP-Server: ftp://melsensor.de. Call MEL hotline for further details.

Communication between Electronic system and PC

The micro controller in the electronic system communicates over Ethernet TCP/IP with the PC. The Scanner head delivers image data to the micro controller and the FPGA in the electronic system.

Task of the Micro controller

- Read out of the Scanner head
- Send Scann profile data over TCP/IP-Protocol
- Receive commands for the Scanner head with TCP/IP Protocol
- Send commands to the Scanner head
- Change and store the TCP/IP-Address on request
- Send Status information
- Run integrated Web Browser

Task of the PC-Software

- Receive profile data from Scanner
- Decode Pixel image data format (see „Data format“)
- Process and display scan profile data
- Send commands to the Scanner head

Communication Components

TCP Clients and Server are components of the development environment in the software. In the readily compiled software (*.exe File) these components exist in the form of “ports”. These ports manage data traffic in and out to the individual elements of the application software.

The Scanner communicates with the *outer world* using *port address*, *IP address* and a valid *subnet mask*. A Gateway is ignored, the scanner opens a peer to peer connection. Therefore the set up of these communication components needs to be complete to allow proper function of the electronic components and data communication.

The Scanner stores data in FiFo memory, before data is sent out in a *package*. The transmission protocol is TCP/IP*.

* UDP firmware version is available on request!

Adjustment of Scanners IP-Address

The Scanners electronic system has a default address as well as a working IP address. The default IP address is activated, when you set the RiP-Pin. This is done, when you connect the Sync out pin to the D1 pin before starting power. The default IP address is set firmly, and can not be changed by the user.

Before delivery, the scanners were normally set to the **working-IP = 192.168.123.245**. When another IP address has been set, either a note in the shipment papers or a label on the electronic box of the scanner should show the IP address.

When you want to set a new working IP address, use a Web browser and follow the procedure on page 24.

🕮 Please note: after transferring the new working IP address to the scanner electronic system, the electronic system *automatically* restarts to make the change effective.

The Web Browser will then display a message to make you aware of the change.
When you receive this message connect to the scanner with the new IP address.

Resolve IP Address conflicts

Using the default IP address is meant for communication with one single scanner, for adjustment of a new working IP address or other service purpose. Using the default IP address for normal operation is *not* recommended.

DHCP is not supported. The Scanner must have a fixed IP address, as well as the network card to which the scanner shall communicate. Scanner and network card must be in the same network segment. This means, the network address should vary only in the last three digits. Computers in the network must use unique and different network addresses. When units in the network use the same address, a conflict exists. The communication in the network could severely have problems, when this happens.

Tipp: label the units when installing different units in a network
take notes and make a plan of the network addresses used in your network.

when a conflict exists, physically disconnect the network cable form different units to find out the source of conflict. ... only one unit at a time...

example:

Address Scanner 1: 192.168.123.222;
Address Scanner 2: 192.168.123.223;
Address Scanner 3: 192.168.123.224;
Address Scanner 4: 192.168.123.225;
Address Scanner 5: 192.168.123.226;
Address Scanner 6: 192.168.123.227;
Address Scanner 7: 192.168.123.224; // incorrect IP-Address = conflict!
Address Scanner 8: 192.168.123.229;
Address Scanner 9: 192.168.124.228; // different network segment
Address Network card: 192.168.123.199
The subnet mask for all Scanners and the network card = 255.255.255.0

Do not get mixed: a subnet mask 255.255.0.0 may also work. The subnet mask defines the maximum number of units in your network. When no more than 254 scanners were present in your network, the subnet mask 255.255.255.0 will be sufficient.

In our example Scanner 7 has been set erroneously to the IP address 192.169.123.224. this address is already taken from another Scanner – so a conflict exists. The correct IP-address would have been 192.168.123.228.
Scanner 9 is in another subnet (another network segment). For Scanner 9, the network card should be set to IP address = 192.168.124.10.

Tipp: how to find out the IP address of the scanner?
1. Use any terminal software, connect the RS-232 diagnostic port to Com-port 1 of your PC
2. Read out the information prompt send out at start up of the scanner.
3. For details see page 22.

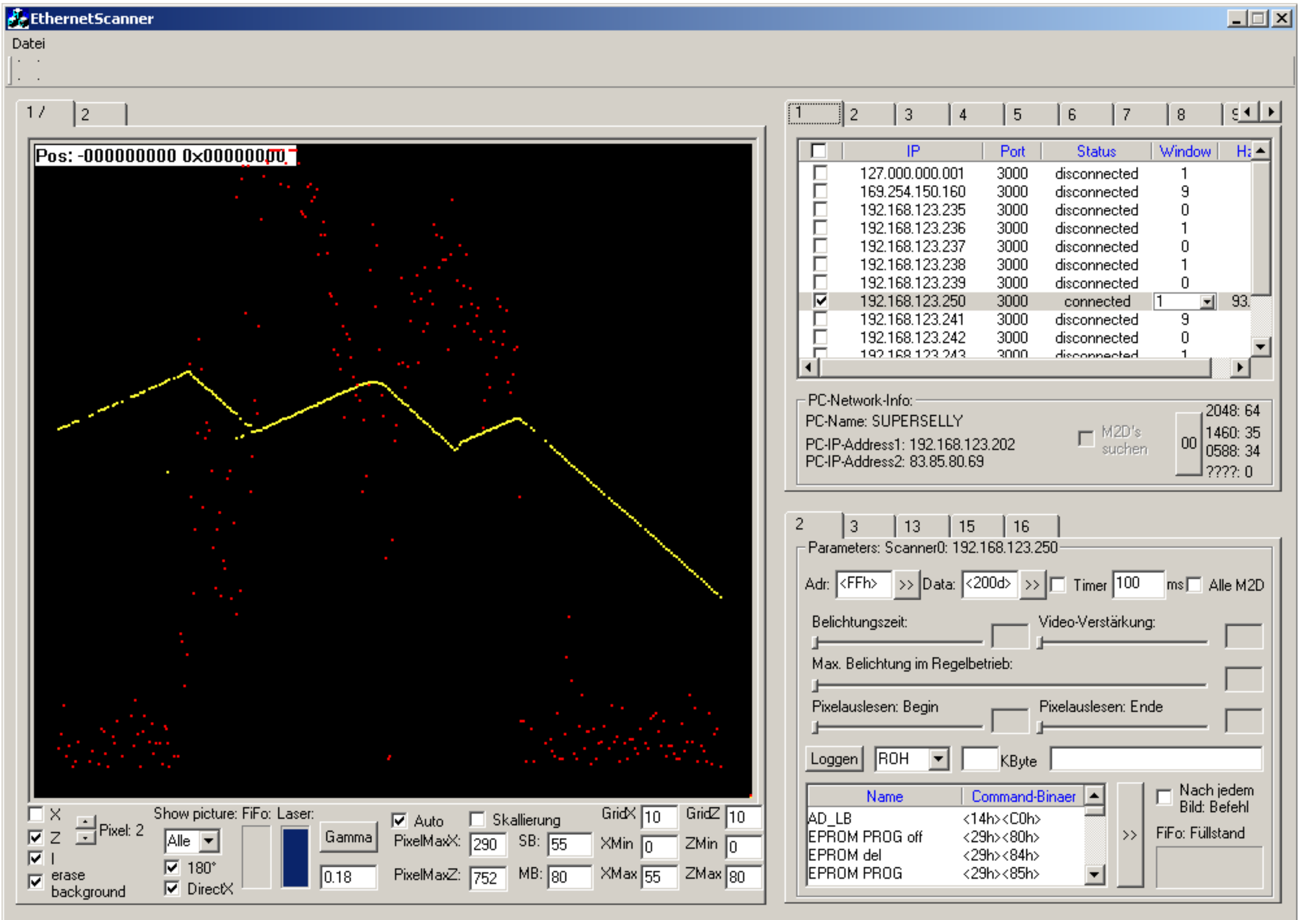
	IP-Address		**factory default**
o	MAC-Address	8 Byte	00-08-DC-00-00-00
o	Serial number	3 Byte	MM-YY-ZZZ
o	Scanner-IP	4 Byte	169.254.150.160:3000 = default Address
o	Subnet-Mask	4 Byte	255.255.0.0
o	Gateway IP	4 Byte	169.254.150.1
o	TCP Port	2 Byte, integer	last Segment of the TCP/IP-Address (for example 3000)

Back ground: the Scanner and the network card in the PC make a “peer to peer” connection. Once the communication between these two peers has been initiated, no other network device will be allowed to communicate with the active scanner. This means, the scanner can communicate only with one PC at a time. Besides the communication over TCP at any time, from any other PC a http request (Web browser) could be made.

MEL Configuration Software „EthernetScanner-2008“

After the Start of the Scanner (switch on power), the configuration of the scanner is loaded from the internal EPROM to the micro controllers memory. The MEL demo software then shows the following information:

- TCP/IP Address, Port and MAC Address
- Intensity, shutter time, Video gain
- Scanner Status, Firmware-Version, Scanner head temperature



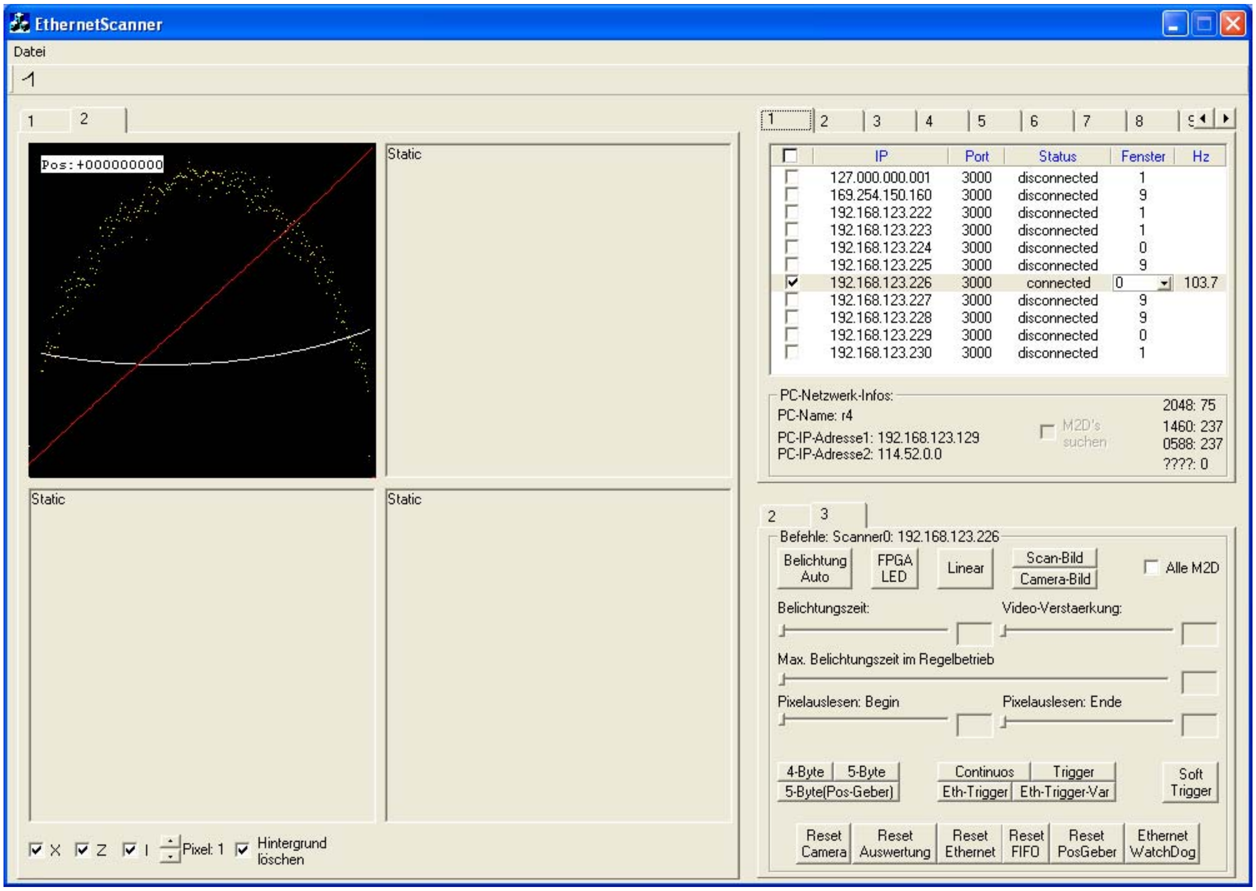
The main window of Ethernet-Demo-Software has the elements profile display and tabs for different function groups. The profile display at the left side can be switched to display one bigger screen or four small screens simultaneously. The colors of the profiles can be adjusted by defining RGB values in the settings.ini file. Function groups are selected with the Tabs 1 … 16. The important Tabs are discussed in the following text. The Tab 1 defines the IP address and assignment of the display window. The window count of the four windows starts at 0 and counts up to 3. The *big window* is number 9.

Temporary change of the IP Address

When you want to set the IP address only for a demonstration double click one of the entries in the Tab 1 and overwrite the IP address for a temporary change. The change will not be stored permanently. When you restart the application, the previous setting will be read in from the ini file.

To make the IP address change *permanent* edit the EthernetScanner.ini file.

Tipp: *use any Text editor to edit the EthernetScanner.ini File, but be aware: when you destroy the ini File, the application could have a problem.*



To change the assignment of the window number, click the column ***Fenster*** in **Tab 1** and select the window number from the pull down list. The column ***Hz*** displays the current scan profile frequency.

Tab 4 displays system parameters:

- Operating hours counter, On counter
- Temperature of the Scanner head
- Firmware-Version, MAC-Address, working und default IP-Address.

Tabs 6 ... 16 are reserved for diagnosis.

Temperature monitoring

The temperature of the scanner head is shown in °Celsius on *Tab 4*. The current temperature is also stored in the register 2 and can be read out from the PC. (see data format description, register 0x11 + status register 2, Byte 12, page 15 and following pages).

Pixelauslesen : Limit of scanners viewing area

The function slider "*Pixelauslesen begin*" and "*Pixelauslesen end*" allows to crop the scanners viewing area. This feature is helpful for scanning extremely difficult targets. It improves scan profiles by cutting off unwanted reflections. Depending on the setting of the demarcation limits, the internal profile acquisition algorithm will ignore reflections below or above the levels of demarcation, which could <u>greatly</u> enhance the online scan profile in extreme situations. The effect can be monitored directly on screen.

Tipp: *mathematical post processing of the profile will <u>not</u> be able to produce a similar effect.*
The effect of erasing reflections and unwanted profile noise is due to removing these data before creating the profile.
This is a function of the FPGA processing inside the scanner head.

Scanner controls

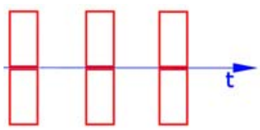
Control commands

No commands must be sent to the scanner, unless special operation is required. When no special command is sent, the scanner is in automatic mode, which is best for most applications.
When the scanner is powered on, it sends data to the connected PC in a continuous stream. You must not care to set the scanner to this operation mode, this will be made automatic, presumed once the application software has been configured correctly. Windows TCP socket cares, that all packets will arrive at the PC, no user interaction is required.

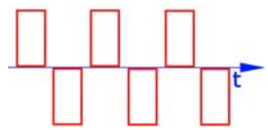
When creating a *new* communication with the scanner after a long waiting period, we recommend to send a FiFO reset command *just once* in the beginning. This is only relevant for your own application software, to avoid reading out *old data* from the FiFO memory at start. When using this command, you should trash the first image after the reset and *use only the second* one. When reading data continuously, FiFo reset is *not* recommended.

🕮 *Tipp: examples for the use of register addresses and commands were given in the appendix.*

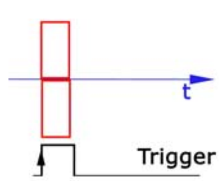
Trigger and Synchronisation operating modes



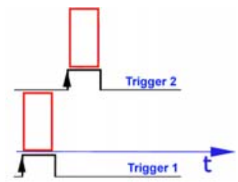
Simultaneous Sync



alternating Sync



simultaneous Trigger



alternating Trigger

External Trigger (Hardware-Trigger)

At Pin 6 = Sync-in, the leading edge of the trigger pulse makes the trigger event. The signal should not oscillate. The Sync-input is an optic coupler with current limit. The voltage range is TTL … 24 V (max. 30 V DC). When the trigger has been made, no additional trigger signal will be accepted during the active measurement. The maximum profile rate is approximately 60 Hz.

🕮 *Tipp:* *how to use trigger:*
1. *set register 0x14, Bit 3 to "1"*
2. *do trigger with a call of register 0x1D,see page 15*

🕮 *Please note:* *in trigger mode, synchronization with other scanners is not available.*

Synchronization

The sync-input is used for synchronization of 1 master with other slave scanners. One scanner is defined to be the master; the other scanners will be slaves. The sync output of the master scanner is wired to the slave scanners sync input, ground is connected accordingly.
🕮 *Please note:* *in sync mode trigger is not available*.

Ethernet-Trigger

The Ethernet-Trigger mode is set in register 0x23 according to the definition on page 17. In this operation mode, the scanner creates permanently profiles, *but holds up data send out*. Only after Ethernet trigger event received, the scanner sends out the last profile acquired *before*. Depending on the speed of the PC, profiles could be missing.

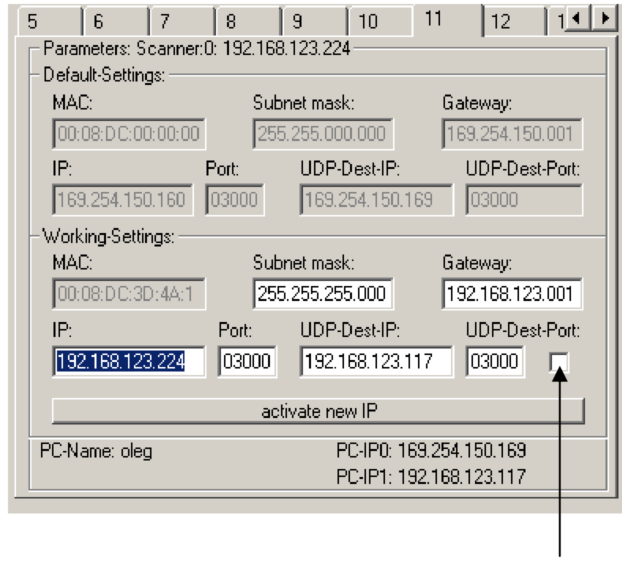
Digital inputs

Digital inputs 1 and 2 may be used to connect an A/B encoder to the scanner head.

🕮 *Tipp:* *the encoder count can be read out from the registers 13 ... 16.*
the information is transferred embedded into the scan profile data.
there is no need for special synchronization of the encoder data with the profile data.

UDP transmission mode

How to set UDP transmission mode



Open MEL EthernetScanner-2008 UDP demo software.

Set Tab 11:
Enter the values:

Scanner IP	= the working TCP IP (left)
UDP destination IP	= enter here the target IP and port where the UDP packets shall be sent.

TCP port can be 3000 or any other port number.
UDP port can be 3000 or any other port number.

Please make sure the ports are accessible in your target system.

When your are done with the entries, click the checkbox next right to UDP destination port (see the arrow) to activate the UDP mode and then click the command bar button below “activate new IP”.
The system will reload and connect to the designated values and mode.

Data format and interface description

Frame Format of TCP data transfer from scanner head to electronic unit

Total block size = =2048 Byte. Packet size: 1460 and 588, or 2048 Bytes. The packet size is set by the hardware and the communication channel of the attached network without any influence of the application software. The TCP buffer may be set to standard value. The user must not care about re-assembling and reconciling the TCP packets. This is done by the network card with the help of the operating system.

Header and payload data

Address	Parameter	Type	Byte	Meaning	Factory default	
00...05	MAC	unsigned char	[6]	Default MAC-Address	00:08:DC:00:00:00	Header
06...09	Reserve	unsigned char	[4]	Reserved		
10...13	Lga	unsigned char	[4]	Default GateWay	169.254.150.1	
14...17	Lsm	unsigned char	[4]	Default Subnet Mask	255.255.0.0	
18...21	Lip	unsigned char	[4]	Default IP-Address	169.254.150.160	
22...23	Tcp	unsigned char	[2]	Default TCP-Port	3000	
24...25	Reserve	unsigned char	[2]	Reserved		
26...31	Mac	unsigned char	[6]	Working MAC-Address	00:08:DC:xx:xx:xx *	
32...35	Reserve	unsigned char	[4]	Reserved	Customer specific	
36...39	Lga	unsigned char	[4]	Working GateWay	Customer specific	
40...43	Lsm	unsigned char	[4]	Working Subnet Mask	Customer specific	
44...47	Lip	unsigned char	[4]	Working IP-Address	Customer specific	
48...49	Tcp	unsigned char	[2]	Working TCP-Port	Customer specific	
50...51	Reserve	unsigned char	[2]	Reserved		
52...59	null_8	unsigned char	[8]	Synchronisations-Raster	8 Null bytes	
60...60	Version	unsigned char	[1]	Protocol-Version number	Default = 3 0x10 = status 0x11 = fault	
61...61	Status	unsigned char	[1]	Scanner Status		
62...62	pic_nr	unsigned char	[1]	Image number		
63...63	status2	unsigned char	[1]	Scanner Status		
64...64	Reserve	unsigned char	[1]	Reserved		
65...65	Reserve	unsigned char	[1]	Reserved		
66...	Scan	unsigned char	[**]	Scan Data	Length and contents depends from protocol version used	Scan
				8 x 0x00	8 Bytes	additional data
				Protocol number	1 Byte	
				Encoder Data	4 Bytes	
				Length of following Bytes	1 Byte	
				Function register	n Bytes	
				Length of following Bytes	1 Byte	
				Status register	m Bytes	
…2040	Fill		[...]	Fill bytes		
2041.2042			[2]	Pixel number horizontal	Typ. = 290	
2043.2044			[2]	Pixel number vertical	Typ. = 752	
2045..47	FiFO			FiFO-fill status **	3 Bytes	

* Serial number ** depending from transmission protocol (from Scanner head to electronic box).

Encoder Data, FiFO fill status

Encoder-Data is transferred with each *Scan-Profile - do not mix up with the special transfer protocol issued by the special command 0x21, which calls the info telegram. The following chapter describes this info telegram. On page 18, the description of the scan data block is given.*
At the end of the image profile information, a block of 8 Bytes 0x00, followed by the version of protocol (for example “3”) and then 4 bytes encoder data were transferred.
The last 3 Bytes before the end of the transmission block is the FiFO fill status data.

📖 *Tipp: the protocol-Version is given in the Header, Byte 60.*

0x21: info telegram: scanner status information

The command 0x21 reads out the status information in *one* complete packet from the scanner. When you have sent the command 0x21 to the scanner and watch out for a packet with the protocol version 0x10. This protocol revision number is in the byte 60 of the header. When you found 0x10 in Byte 60, you have identified the info telegram packet. Protocol version 0x11 is the message that something with profile scan data status is wrong.

At the end of the packet, additionally 31 bytes of function register status is sent, then the FiFO fill status information. The table on the next page shows the details.
When the command 0x21 is received by the scanner, a packet with 2048 Bytes is sent from the scanner to the PC. The packet contains all information of the registers 0 … 63, but no scan profile data. From Byte 130 on, the firmware version is sent as a string. The end of the string is 0x00.

Info Telegram (the scanners answer to the command 0x21)

	Byte Nr.	Register Nr.	Function	Length|type	Ending	Firmware	
Header	00 ... 51		Header	52 Byte		...1.10.0	1.11.0+ ...
Header	52 … 59		Synchronization	8 x 0x00		...1.10.0	1.11.0+ ...
Header	60 … 60		Protocol version	1 Byte		...1.10.0	1.11.0+ ...
Header	61 … 61		Scanner Status 1	1 Byte		...1.10.0	1.11.0+ ...
Header	62 … 62		Image number	1 Byte		...1.10.0	1.11.0+ ...
Header	63 … 63		Scanner Status 2	1 Byte		...1.10.0	1.11.0+ ...
Header	64 … 65		Reserve	2 Byte		...1.10.0	1.11.0+ ...
	66 … 97	0 ... 31	Status-Register	32 Byte	-	...1.10.0	1.11.0+ ...
	98 … 129	32 ...63	Eprom Data	32 Byte	-	...1.10.0	1.11.0+ ...
	130 … x	-	Firmware-Version	* String	0x00, 0xFF	...1.10.0	1.11.0+ ...
	X+1 ... x+32	0 ... 30	Functions-Register	31 Byte	0xFF	...1.10.0	1.11.0+ ...
	X+33 ... x+35	123...125	FiFO fill status	3 Byte	0xFF		1.11.0+ ...
	... 2047		Fill bytes	0xFF			1.11.0+ ...

* the length of the firmware string is defined by the ending 0x00. the length of the firmware string varies with each version! Between the register data and the FiFO-Bytes 0xFF may appear! The position of the FiFO-Bytes is after the function registers. Always a block of 2048 Bytes is sent, at the end of the block “old” data (trash) may exist.

command: 0x21, read out register-dump –Scanners action upon 0x21:

- 64 EEPROM Registers are read out
- the protocol version is set to 0x10
- the sync raster is written 8-Null Bytes
- from Byte 130 of the data section the firmware-Version number is sent
- no Scan-Data were transferred

Data format of the Packet valid from Firmware Version 1.11.0 and higher.

Data format: Register addresses, Commands or Data

MSB = „0“ = command or Register number MSB = „1“ = Register content
To change a value, first send the register number and then the value.
The register number remains active, unless a different register number is sent.

Tipp: *double registers were made active only when the higher register is transferred.*

Order of bits

Register Lo-Byte	Data Lo-Byte	Register Hi-byte	Data Hi-byte
76543210	76543210	76543210	76543210
0xxxxxxx	1xxxxxxx	0xxxxxxx	1xxxxxxx

Tipp: *if not otherwise stated: for registers Bit 7 is not used and always 0.*
for data, Bit 7 is always high.

Scanner controls

Register HEX	Register DEZ	Bit	Function register (marked light blue)	Remarks	
0x0	0	6..0	Shutter time Low	Shutter time 0 = max 1022 = min 1023 = Laser off	Manual control
0x1	1	2..0	Shutter time High		
0x2	2	6..0	Max shutter time Low	Limit for max. shutter time an. 0 = max 1022 = min; for alt. Sync P 130	Automatic control
0x3	3	2..0	Max shutter time High		
0x4	4	6..0	Begin pixel readout	The range is 0 … 127. 1 Bit corresponds to 8 Pixel condition: (Begin < End) Max is automatically limited	
0x5	5	6..0	End Pixel readout		
0x6	6	6..0	Video gain Low	0 = +6 dB 1023 = +40 dB (default = 400 ... dB) Value is transferred only after sending high	
0x7	7	2..0	Video gain high		
0x8	8	6..0	Intensity threshold	1 – 127; Intensity threshold for the internal laser control Default = 15	
0x9	9	6..0	Laser value	1 – 127 value for Laser control (default = 95)	
0xA	10	6..0	Peak width Limit	0 = 0ff; other = Peak width Limit, max. = 127 Pixel	
0xB	11	0	FPGA OK LED	0 = on 1 = off	
0xE	14	*	Reset Position counter	The counter is set to 0	
0xF	15	0	Synchronization	0=simultaneous measurement 1=alternating measurement	
0x10	16	0	set Scan profile, image complete	0=Scan profile data, 1=complete image (see page 22)	
0x11	17	5..0	Sensor adjustment Status messages (marked yellow)	0 = Sensor temperature 1 = Register contents 2 = Version number electronic system 3 = Version number camera 4,5,6,7,8 = operating hours counter in 0,25 Seconds-Ticks 9,10,11 = on counter 12 = inputs 1 + 2 + Bit for Sensor with mirror 13 = Bit 2 … 9 of Laser control 14 = Version number example: 3.5xxxx xxxx is the version number 31..15 = NC 63..32 = 32 Byte Eprom-Data (see below)	
0x12	18	2..0	Set protocol version	0 = Version1 / 1 = Version2 / 2 = Version3 / 3 = Version 4	
0x13	19	*	Reset camera chip	Image counter and camera were reset	
0x14	20	2..0 3 5 ..4	Do not use! Trigger mode | continuous Definition field mode	 0 = continuous 1 = single shot with trigger 00 = both fields| 01=only 1st field | 10 = only 2nd field	
0x15	21	0	Shutter control mode	0=automatic, 1=extern / manual (Register 0 & 1)	
0x16	22	0	Linearization	0=off, 1=on; Start value 1	
0x17	23	0…1	Control of M20D-XF 1.000 Hz	0 = 300 Hz; 1 = 500 Hz; 2 = 700 Hz; 3 = 1.000 Hz	
0x18	24		Special register for ISA Hardware	Read out Status register *do not use with Ethernet-Hardware!*	
0x19	25		Not used		
0x1B	27	6.. 0	Threshold of Profile recognition	Default = 10 Max = 127 = 50% of maximum description: p. 22	
0x1C	28	*	Reset FIFO	Clear FIFO is erased	
0x1D	29	*	Single shot in Trigger mode	Register 0x14 must be set to make this function available	
0x1E	30	*	Reset Sensor	All register values were reset	
0x1F	31	*	Reset Ethernet module	Ethernet module restarts	
0x20	32	*	Watch-Dog Test	Watch-Dog resets Ethernet Module	
0x21	33	*	Information telegram from Ethernet module	32 Register + 32 Eprom values are sent. The transmission protocol version is 0x10	
0x22	34	*	Save new network settings	See Manual page 14 and following.	
0x23	35	0	Ethernet Trigger mode	0= deactivate 1= activate when active, read out picture with accessing to register 0x1D	
0x24	36		Set HDR mode Dual shutter operation	0 = deactivate 1 = activate + 5 Byte Data Command: 0x24 | Active: 0x81 | Image: 0x80+xx | 1.shutter: 1.st field, 2.nd | 2.shutter: 1.st, 2.nd Detailed description on page 22.	
0x25	37		Not used	Do not use!	
0x26	38		MEL internal register	Do not use!	
0x27	39		Not used	Do not use!	
0x28	40		Not used	Do not use!	
0x29	41		Eprom command Register	MEL internal register – do not use! 0 = deactivated	

				1 = read Eprom 2 = write Eprom data into Ram (1Byte = 7 Bit) 3 = read Eprom data (1 Byte = 8 Bit) 4 = erase Eprom data 5 = program Eprom
0x2A	42	6..0	Eprom Data	512 k must be written. Data is sliced to 7 Bit, which are reconciled in the sensor to 8 Bit
0x7F	127		Dummy Register	No operation

* <u>each</u> access to the register triggers the function. It is not required, and it makes no sense to write any value to the register. Simply calling the register address makes the necessary function work.

The register 0x11 selects the content of the status register. The status register has the parameters temperature of sensor head, register contents, version of electronic system and camera, operating hours counter, on counter, digital inputs, version of EPROM Firmware and 32 Bytes EPROM data, serial umber, scan range geometry. This method is valid for Ethernet Scanners, ISA board and i-Control.

<u>Special register 0x18 (24)</u>

The special register 0x18 allows to query sensor data direct. When a value smaller than 127 is written to the register, the read out of the FiFO is stopped and the register is put to the data bus. The number of the register is the content of the special register. Data will remain on the bus as long as the special register is written again with 127. Then normal image transfer is possible again.

Register	Bit	Function	Meaning
0 - 30	6..0 7	Control register NC	Shows the content of all control registers. Except Registers which act as „impulse switch“ like „Reset Sensor“ - unused Registers and Bits give back a “0”
31	7..0	Status Register	Status messages are selected with Register 17
123	7..0	FIFO number of Bytes 1	Bit 7..0
124	7..0	FIFO number of Bytes 2	Bit 15..8
125	2..0 7..3	FIFO number of Bytes 3 NC	Bit 18..16
126	*	Save values for Register 123 - 125	

Data format

Sync signal and Status info, encoder information (protocol version 3)

Byte Nr	Value	Bit Nr.	Meaning	
52 .. 59	0	8 times 0x00 for Synchronization		
60	Version number	7...0	Version des Scan-Data format	
61	Status Byte 1	0	0=not linear, 1=linear	
		6...1	Content of Register 17	
		7	Always 0	
62	Image number	Continuously counting from 0..253		
63	Status Byte 2	**The content of status register_2 is selected by Register 0x11** all values made of more than 1 Byte, always the Bit 7 is 0. Per Byte only 7 Bit were transferred.		
64, 65	Reserved			
66 ...X	Image data	Data format Vers.1: 4 x 283 = 1132 Bytes * Data format Vers. 2+3: 5 x (Reg.34 ;35 (Rev.4=291)) = 1455 Bytes		
X+1..X+9	0	8 times 0x00 for Synchronization		
	Version number	7...0	Version of Data format (3)	
	Position encoder Reg.1	6..0	Position encoder Bit 6..0	Two's complement
	Position encoder Reg.2	6..0	Position encoder Bit 13..7	
	Position encoder Reg.3	6..0	Position encoder Bit 20..14	
	Position encoder Reg.4	5..0 6	Position encoder Bit 26..21 Direction of Position encoder	
… 2040	Fill	[...]	Fill bytes	
2041.2042	Pixel number horizontal	2 Bytes		Typ. = 290
2043.2044	Pixel number vertical	2 Bytes		Typ. = 752
2045..47	FiFO	3 Bytes	FiFO-fill status	

The Data packet has *always* the length of 2048 Bytes. * iLAN Scanner supports ***<u>only data format 3</u>***.

Meaning of the status byte 2 depending from register 0x11

Value = [0]: Sensor temperature

Reg. value	Temperature	Value (Hex)	Value (Bin)	
0	+126 C°	7E	1111 1110	In 1 degree steps from -55 to +126 C°. Bit 7 is the sign!
	+ 85 C°	55	1101 0101	
	+ 25 C°	19	1001 1001	
	+ 0 C°	00	0000 0000	
	- 1 C°	FF	0000 0001	
	- 25 C°	E7	0001 1001	
	- 55 C°	C9	0011 0111	

Value= [1]: Register contents

Reg. value	Bit Nr.	Meaning
1	0	0= not linear, 1= linear
	1	0= Register contents as after Reset, 1=after write to Register
	2	0= Scan data, 1= complete image
	3	Laser 0 = on 1 = off
	4	Measurement control 0 = continuous 1 = single shot with Trigger
	5	Laser control 0 = automatic 1 = extern (Register 0 & 1)
	6	NC
	7	0

Value = [2, 3]: Version number electronic system and camera

Reg. value	Bit Nr.	Meaning
2	7 ... 0	Version electronic system example: 36 corresponds to Revision 3.6
3	7 ... 0	camera

value = [4, 5, 6, 7, 8] : operating hours counters (Bit 7 always 0) 250msec per count

Reg. value	Bit Nr.	Meaning
4	6..0	Operating hours counter Bit 6..0
5	6..0	Operating hours counter Bit 13..7
6	6..0	Operating hours counter Bit 20..14
7	6..0	Operating hours counter Bit 27..21
8	3..0	Operating hours counter Bit 31..28

Value = [9, 10, 11]: on counter

Reg. value	Bit Nr.	Meaning
9	6..0	On counter Bit 6..0
10	6..0	On counter Bit 13..7
11	2..0	On counter Bit 16..14

Bit 7 is always 0. The maximum counter value is = 131.072

Value = [12]: Digital inputs 1 , 2 + Bit for Sensors with a mirror

Reg. value	Bit Nr.	Meaning
12	0	Digital input 1
	1	Digital input 2
	2	Bit for Sensor with a mirror ; „1“ = with mirror
	7...3	NC always 0

Value = [13]: Laser control Bit 2...9

Reg. value	Bit Nr.	Meaning
13	7 ... 0	Bit 7 ... 0 of the register 13 represent the Bits 9 ... 2 of the laser control value in automatic mode. The lower bits were not sent back by the sensor, these can not be read back. For adjustment, the bits 2 … 9 are sufficient for set up. For set up the sensor firmly to a special value, we recommend the following procedure: 1. set the Scanner in automatic mode on the target surface 2. read out Bit 2 ... 9 from status register 3. set Scanner in manual controlled (external) mode 4. write back the previously read out values Bits 0 and 1 are set to „0“

Value = [14]: Version number

Reg. value	Bit Nr.	Meaning
14	7 ... 0	The revision of the firmware is determined from status registers 2 and 14. Register 14 shows the 3rd digit of the version number. Example : revision = 3.6.2 Status Register 2 = 36 // Version x 10 = 3.6; Integer Status Register 14 = 2 // natural number = 1, 2, 3 … Integer

Value = [15 ...31]: not assigned

Reg. value	Bit Nr.	Meaning
15	7 ... 0	always 0

Value = [32 ... 63] 32 Byte Eprom Data Register

In the Register 0x11 (marked yellow on page 15) is defined which register values were read.
Bit 7 of the register is not used and always 0.

Reg. value	Bit Nr.	Meaning
32	6..0 = LB	Pixel number of camera horizontal
33	6..0 = HB	
34	6..0 = LB	Pixel number of camera Vertical
35	6..0 = HB	
36	6..0 = LB	Serial number (see below)
37	6..0 = LMB	
38	6..0 = HMB	
39	6..0 = HB	
40	6..0 = LB	Begin of range *
41	6..0 = HB	
42	6..0 = LB	range *
43	6..0 = HB	
44	6..0 = LB	Scan width at start of range *
45	6..0 = HB	
46	6..0 = LB	Scan width at end of range *
47	6..0 = HB	
48	6..0 = LB	Maximum value for measurement range linear
49	6..0 = HB	
50	6..0 = LB	Maximum value for scan range linear
51	6..0 = HB	
52	6..0 = LB	Minimum value for measurement range not linear
53	6..0 = HB	
54	6..0 = LB	Minimum value for scan range not linear
55	6..0 = HB	
56	6..0 = LB	Maximum value for Measurement range not linear
57	6..0 = HB	
58	6..0 = LB	Maximum value for Scan range not linear
59	6..0 = HB	
60	0	0 = Half frame camera, 1 = full frame camera
	1	0 = normal, 1 = camera image mirrored
	2	0 = normal, 1 = camera rotated by 90 degrees
	3	0 = dimensions in 0,1mm 1 = dimensions in 1mm*
	6..4	NC
61		Not used
62		
63	6..0	EProm Data Version = 1

* dimensions in 0,1 or 1 mm increments

Serial Numbers representation
MSB are not used

Bit-Nummer

HB	HMB	LMBB	LBB
27..21	20..14	13 ..7	6 ... 0

The special Register 0x18 puts the scanner into the operation mode without FiFO. This operation mode makes only sense for ISA electronic cards. Therefore this register is not discussed further in this manual. The register address 0x18 should not be used in the application software!

Register 0x1B: Profile peak recognition threshold

This register controls the algorithm responsible fort he profile detection in the FPGA. A change in setting should be made only, when the result could be verified immediately. When no improvement is achieved, you should immediately set back default value = see page 17.

HDR-shutter control, Register 0x24 (36)

A command sequence can make the camera toggle laser intensity from one frame to the next. This feature is meant for situations, where the object has highly shining and in the same time mat diffuse reflecting surfaces. Bright and very dark regions on the object may be pictured with two settings each giving optimum picture from parts, combining both will give a more complete picture. (HDR = high dynamic range).
The following code example shows the syntax of the command.

<24h> = command: The highest Bit is <u>not</u> set!
All following Bytes the highest Bit is set!

commands: <24h><80h> = deactivate
<24h><81h> = activate

FieldSwitchMode: <80h> = each field is captured with different shutter

FrameSwitchMode: <81h> = each 2^{nd} field is captured with different shutter

B1 = LaserIntensity 1; B2 = LaserIntensity 2
The scanner toggles between the two values at each field

<u>example:</u>
<24h><81h><80h><87h><D4h><80h><E5h> → B1=980 B2=101

syntax:
<24h><81h> = command "activate"
<80h> = definition "FieldSwitchMode"
<87h><D4h> = <Hi-Byte1><Lo-Byte1>
LaserIntensity -Hi-Byte: (X >> 0x07) | 0x80
LaserIntensity -Lo-Byte: (X & 0x7F) | 0x80
... This sends the value *980*
<80h><E5h> = <Hi-Byte2><Lo-Byte2>
LaserIntensity -Hi-Byte: (X >> 0x07) | 0x80
LaserIntensity -Lo-Byte: (X & 0x7F) | 0x80
... this sends the value *101*

For switching off, two Bytes are ok! „<24h><80h>“

Tipp: for activating of the function, the command must be sent in complete length:
(<24h><81h><80h><87h><D4h><80h><E5h>) – never fragment the command sequence!

* In Photography the term HDR is used for a technique, when two pictures of the same object are shot with different shutter time values. The first picture with high shutter time will show the high light situations, the second picture with longer shutter time will reproduce the shadowed parts of the object in a better way. The combination of both pictures in software will give a picture representing as well high light and shadows in the best way.
MEL now offers a HDR option for the M2-iLAN scanners.

M2DF/LAN-Structure of Image Data

Linearized and not linearized								
Each field has the number of pixels = lines / 2 of the CCD. For each line 5 Bytes with X and Z, and related Intensity value is given. X and Z are given with 14 Bit, Intensity I is given with 8 Bit.								
Byte Nr.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	0	**X6**	**X5**	**X4**	**X3**	**X2**	**X1**	**X0**
2	0	**X13**	**X12**	**X11**	**X10**	**X9**	**X8**	**X7**
3	0	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
4	0	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**	**Z7**
5	**I7**	**I6**	**I5**	**I4**	**I3**	**I2**	**I1**	**I0**
	<u>not linearized</u>				<u>linearized</u>			
distance Z Scan range X Intensity I	0..number of pixels horizontal 0..number of pixels vertical/2 1..254				0…max.16383 according to linearization table 0…max.16383 according to linearization table 1..254			
Byte value can never be bigger than 254. Bytes = FF(Hex) are invalid. This may appear when FIFO is empty or busy.								

Read out complete image

The complete camera image is read out. The sensor sends out raw data with 8 bit resolution. When the register 0x10 is set to "1", the sensor writes 2 fields complete into the FiFo. After each field, 8 Byte zero's and status Bytes are transferred. The read out of a new image starts with a hardware or software trigger event. Intensity value of the pixels is 0 … 254.
The number of Pixels horizontally and vertically of each profile can be read out of the registers 32 … 35 or am at the end of the Ethernet Packet in the Bytes 2041 … 2044.

Tipp:	Set Trigger mode		0x14: Bit 3 = 1	
	Activate image mode		0x10: Wert = 1	
	Empty FIFO	*	0x1C	
	Trigger	*	0x1D	* call of the register address triggers the function!

Synchronization of Sensors

„Synchronization" means simultaneous or alternating read out of the sensors. For synchronization, the Pins Sync out of the Master-Scanner and sync in of the Slave-Scanner unit must be connected, as well as ground pins. The connections are made in the power cable at the 8-pin connector.

Pin 2 of the master sensor is connected to pin 15 of the slave sensor. Both image counters work now synchronous. Phase error is less than 10 µsec. When the register 0x0F is set to „1" both sensors work in alternating mode. Shutter time in register „2" and „3" must be set to the value > 512, in order to avoid overlap of the two sensors.

Trigger Mode

Trigger mode must be activated in the register 0x14, Bit 3 with a "1". Hardware trigger is made with a leading edge of the sync input, pin 15. Software trigger is made by accessing the register 0x1D. Software trigger and hardware trigger both have the same function. In both modes, the sensor reacts on the leading (rising) 0/1-edge of the trigger impulse. The capture process starts 65 µsec after trigger. The image counter is reset, two fields are sent out.

Calculation of shutter time

The camera creates approximately 93.5 profiles per second. This corresponds to 10.696 msec period. The maximum shutter time is shorter than 10.696 msec, because of the minimum erase time interval.

Maximum shutter time	**= 10.696 msec * 256 / 290**	**= 9.442 msec**
Minimum shutter time	**= 9.442 msec / 16.384**	**= 0.5763 µsec.**

Number of pixels, encoder data

The actual camera model sends with 5 Byte data format selected and protocol version 3 the number of 290 pixels per profile.

RS-232 Monitoring

On the RS-232 interface the scanner sends status information.

Data format:	8 bit, no parity, 1 Stop bit 115.200 Baud
Requirements:	RS-232 cable connected at the D-sub-25 pin connector, 24 V power supply PC with terminal software application waiting for data

When switching power on, the electronic system sends out a prompt over the RS-232 interface.

Tipp: When the IP address is unknown, at first try to read the RS-232 prompt during the start-up.

RS-232 Prompt

- Status, Serial number, Version of Hardware, Version of Software
- Version of Firmware, Version of Eproms
- MAC-address, Gateway Address
- Default IP-Address, working IP-Address, Subnet mask

Example of a RS-232 Read out:

```
*******************************************************************************
M2-EthernetScanner-AllInOne v.2.0.55 090306 TCP/UDP 15:51:52
WebServer: aktiv
P9.4: 0 WDC5: 0x01
H:290 W:752
Seriennummer: 04 08 456
Register-EEPROM-Data:
80 03 2D 7F 5D 09 52 03
00 1C 02 00 04 00 01 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
70 05 22 02 08 77 18 00
12 04 58 04 2C 02 10 03
7F 1F 7F 1F 00 00 04 00
3C 17 47 04 02 7F 7F 01
M2D-Kopf: 127
M2D-Auswertung: 45-1
Working-Data
MAC: 00:08:DC:18:77:08
IP: 192.168.123.245:3000
SubNetz: 255.255.255.0
GateWay: 192.168.123.1
Broadcast-Packet
initW3100A
InitNetConfig
Socket-Web aktiv...
...
TCP-Connecting...
EthernetLinkOK
...
ClientIP:192.168.123.129:1378 FiFo:0 0 0
TCP-Rx: 1
0:0x21
InfoPacket:
H:290 W:752
97 03 2D 7F 0C 0A 52 03
00 1C 02 00 04 00 01 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
70 05 22 02 08 77 18 00
12 04 58 04 2C 02 10 03
7F 1F 7F 1F 00 00 04 00
3C 17 47 04 02 7F 7F 01
Version: v.2.0.55 090306 TCP/UDP 15:51:52
00 00 00 00 00 5E 10 03
23 5F 00 00 00 00 00 00
00 00 00 00 00 00 00 00
18 00 00 05 00 00 00 FF
FiFo: 07 FF FE
BildNr:0!=2
BildNr:106!=90
BildNr:91!=72
ClientIP:192.168.123.129:1378 FiFo:0 0 0
Freq: 198Hz
TCP-M2: 6
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
Freq: 93Hz
TCP-M2: 6
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
ClientIP:192.168.123.129:1378 FiFo:0 0 0
Freq: 93Hz
TCP-M2: 6
ClientIP:192.168.123.129:1378 FiFo:0 0 0
```

The scanner continues sending “connection status messages” like in the last lines every second. These messages may show that the scanner is “alive”.

Web-Server

Firmware Revision 2.0 and higher offers a built-in Web-Server. Over the Internet-Explorer or any other Web-Browser, you may access your Scanner.

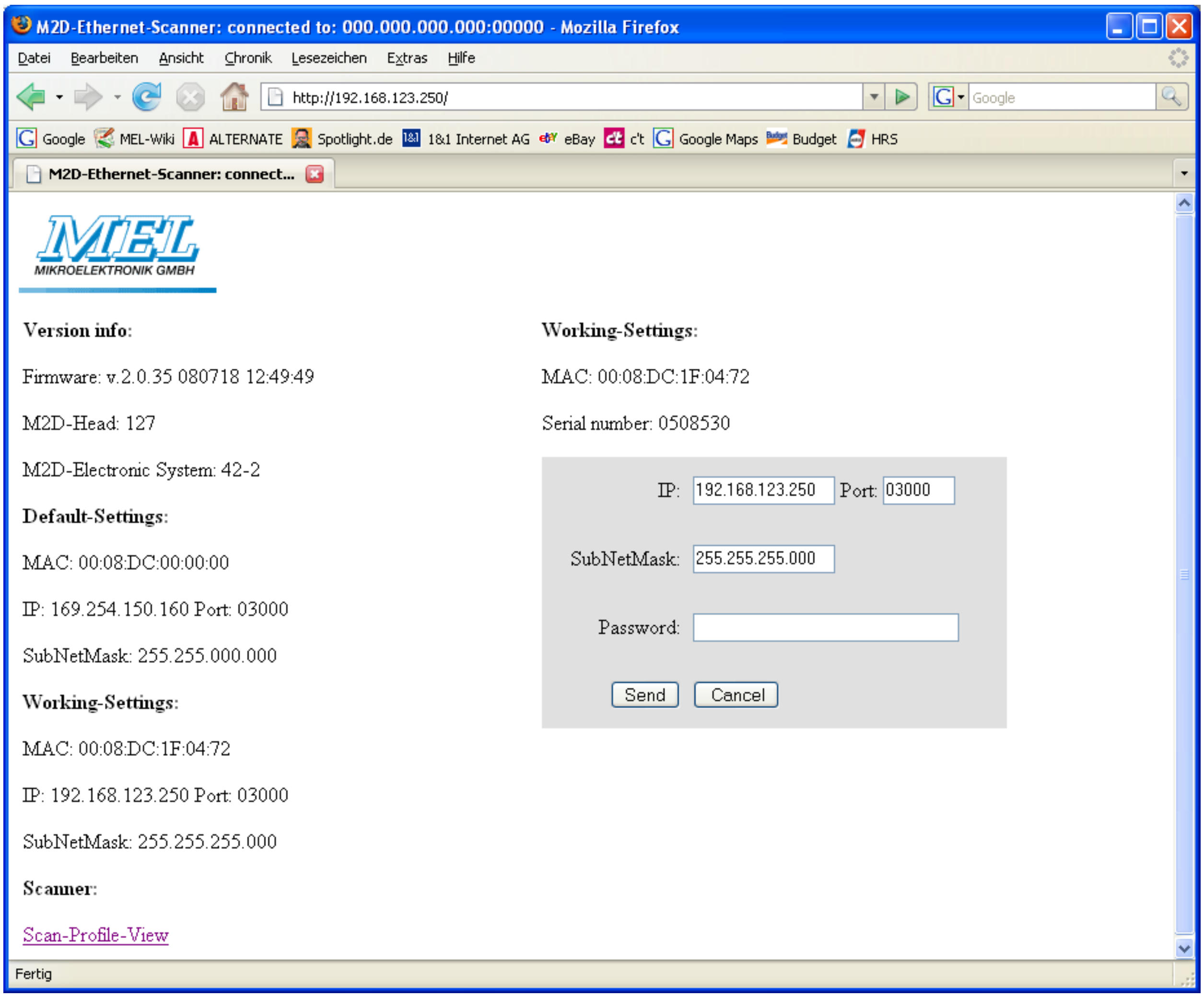
Available Read out Functions

IP address, MAC-address
default settings, Firmware Version, serial numbers.

The IP-address can be changed the from a remote location using the Web-Browser.

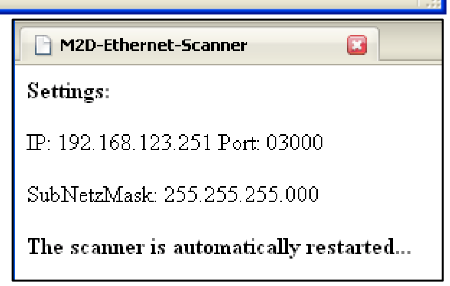
How to set the IP address over the Web

In your Web-Browser, enter the IP-address of the Scanner. The screenshot pictured below will appear.

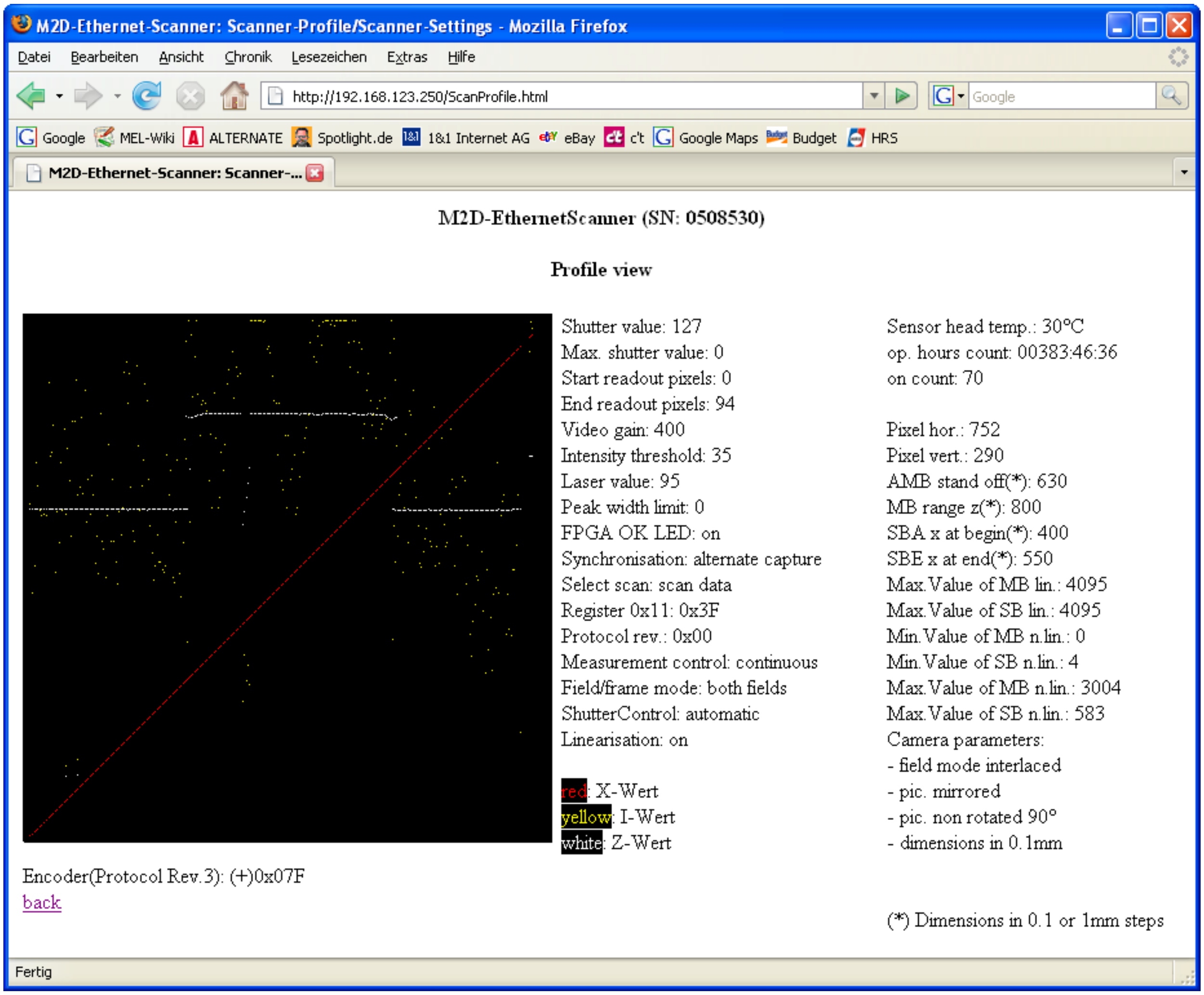


You may now enter a new working IP address in the segment marked in grey color. Type the password MELSENSOR, all capital letters and hit ENTER or press the *Send* button.

Wait for the second prompt from the Scanner, stating that the new address has been accepted and restarted. When you receive the message shown at the right side connect to the scanner using this new IP address.



📖 *Tipp:* *the Web-Server provides a profile snapshot, when you click on the link Scan-Profile-View: see picture next page!*



Tipp: make sure, that your PC is in the same network segment (subnet mask).

Requirements for connection to scanner

- Firmware Rev. 2.0x or higher
- Scanner powered up
- Remote PC connected over Ethernet to the Scanner
- RIP mode *not set*
- Programming mode *not set*

please note: *the default IP address can* ***not*** *be changed.*
Factory setting for the working IP address on delivery is set to 192.168.123.245.
The service PC needs to be in the same subnet as the Scanner.

Tipp: *try a ping command in the* ***cmd*** *shell when you are not sure about the reaction of the scanner.*
When the ping does not respond and the scanner is unreachable for the PC, check your network settings.

Tipp: *IP address set up can not be accessed while the scanner is in flashing operation mode.*

Updating (Flashing) Firmware

The Firmware is responsible for the functions supported by the Micro-Controller in the electronic unit. It can be updated by flashing (re-writing) the firmware. Current Firmware version is 2.0.55.

Hardware Requirements

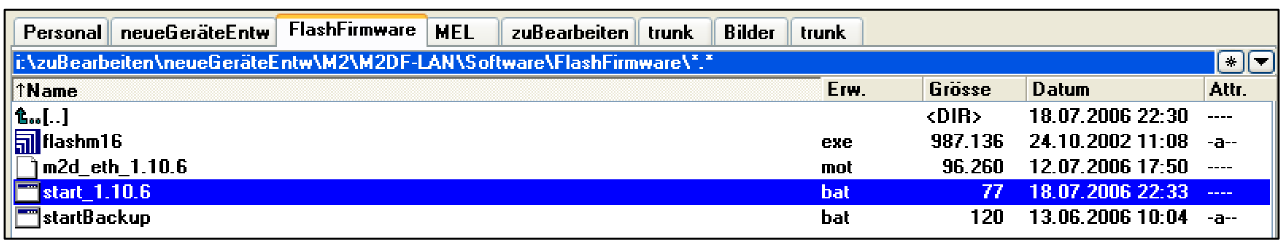
Updating (Flashing) the firmware is made over the RS-232 interface in the electronic unit.
A 1:1 serial connection to the D-Sub-25 pin connector (see page …) and a PC with a RS-232 (COM-port) is required. The upload speed is 57.600 Baud. The serial port in the PC may be set to 115.320 Baud.
The Flash-Tool can access COM-1 to COM-4.
COM-ports mapped to higher or other port addresses (eventually by USB adapters or other) may not work.

Software Requirements

The COM-port must be set in the *start.bat* batch-file. Default is Com 1.
The batch file also names the firmware binary file which is used for the upload. Make sure, that the file name in the batch file is the same as your firmware binary. If this is not the case, edit the batch-file in a text editor and save it. See the screenshot below as an example.

The files in the screenshot below are the files you may need for the update procedure:

Flashm16.exe	executable updater
M2d_eth_2.0.55	firmware binary
Start.bat	batch file
startBackup	*optional report generator*



Before you rewrite the firmware, optionally you may use the **startBackup.bat** to read out the memory contents for diagnosis. StartBackup.bat creates a text file. Send this text file to MEL for diagnosis in case of problems!

The Flash-Tool "**Flashm16.exe**" will be delivered from MEL on request together with the binary data file of the firmware. Flasm16.exe must be started with the start.bat batch file. Start.bat defines the parameters for flashm16.exe.

📖 **Please note:** Flashing the firmware is at customers risk. MEL does not make any warranty, that the flash process will be always successful. When the Scanner does not work any more after flashing the software, this is not covered by any warranty*. With the download of the binary data file* you accept this condition.

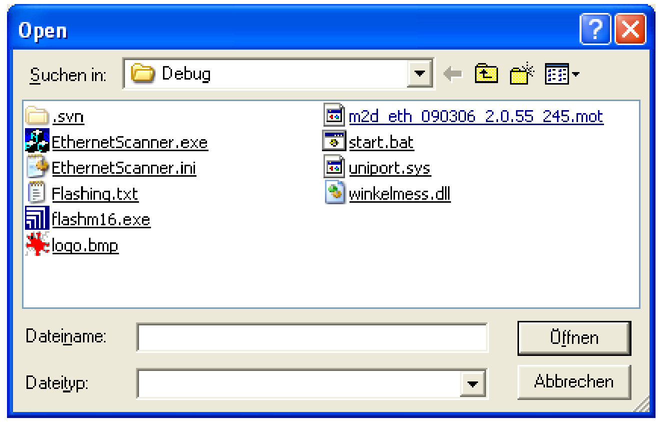
* MEL makes repair for units broken while trying to update in the factory at Eching / Germany. Customer must cover shipping cost both ways and a handling fee.
For details please contact MEL services: info@MELsensor.com

Updating (Flashing) procedure

📖 *Please note:* *the sync out pin must be connected to D1, when the Scanner is powered up.*
A M2-RS232ProgBox with switches is available as a special accessory.
See "special accessories" on page 7.

Step	Action	Remarks
1	Connect the Scanner	Power = off
2	Connect RS-232 to Com-1	Baudrate of PC Com-1 = 115.200
3	Power up the Scanner	*Press and hold the Prog switch* for 1 second while you switch power on*
4	The scanner enters Prog mode	
5	With start.bat launch the Flash-Tool "Flashm16.exe"	Before launching check start.bat with a text editor Use start.bat to launch the Flash-Tool. The upload speed is 57.600 Baud, Baud rate adjusts automatically
6	Wait until Flashing END	This may take a few minutes!
7	Close Flash Tool	
8		Normal operation

* The Prog. switch connects Sync-Out to Digital Input 1
A service adapter is available as *special accessory* from MEL, see page 7: M2-RS-232 ProgBox.



Updating (Flashing) over the Ethernet

The Controller-Firmware of the Scanners can be uploaded through the Ethernet connection.

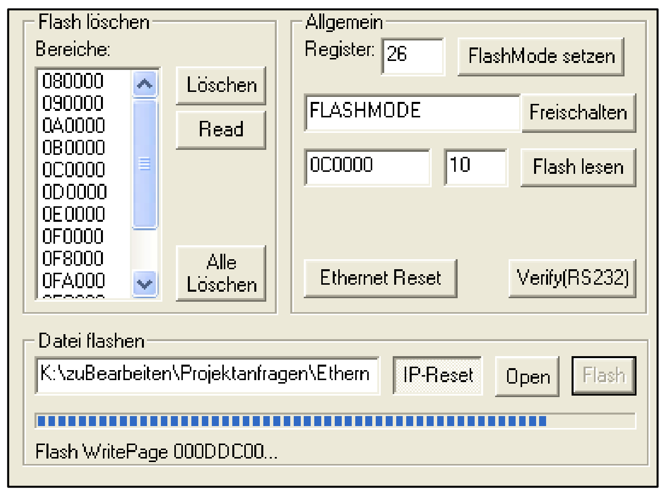
Use the MEL configuration software Ethernet-Scanner-2009 (Release date 26.Jan.2009) and have a _valid_ Firmware file for the version of scanner hardware in use. The prog switch must be set as described in the chapter before.

In the configuration software Tab 15 is selected, with the button *Datei flashen* and „Open“ the firmware file is selected.

The Firmware-File defines, which IP Address the scanner will have after the flash process, when the option "*IP-Reset*" is activated. Please note: the IP address defined by the firmware file can be set only by MEL.

When the option IP-Reset is not activated, the IP-Address of the scanner remains unchanged.

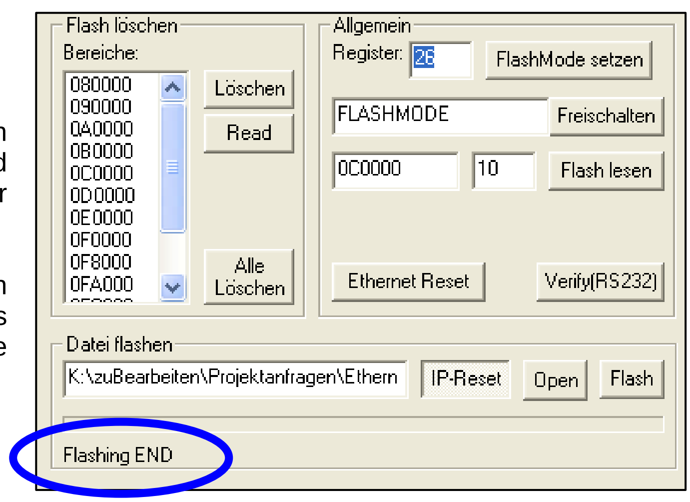
After uploading of the Firmware you can change the IP Address of the Scanner with the Web Browser.



<u>How to proceed:</u>

1. open EthernetScanner-2009 - Tab 15
2. load Firmware file: click „Open"
3. Wait until file has loaded
4. click „Flash“
5. Wait until „Flashing END“ appears.

While the Flashing process is active, do not switch off the scanner – this will ruin the programming and render the scanner to a status where further programming is not possible.



The FPGA Firmware can be uploaded with an internal connector inside the scanner head. This upload procedure can be made only at the headquarter at MEL in Eching / Germany.

Trouble Shooting

Function	First action	Second action	Remarks
No LAN connection to the PC	Check network cable	**Use other cable** **Use an Ethernet switch** **Use other PC**	Check network card settings, use x-link cable and direct connection
	check RS-232 prompt on power up of scanner	Check if head is properly connected, laser lit and LED's lit or blinking	Terminal Software should be active on power up of scanner
	Check if the IP is free Power down the scanner to see if another device has taken the IP **Try M2D-iVision.exe**	**Ping the IP**	Cmd: ping xxx.xxx.xxx.xxx When the scanner is down, no other device should answer the ping. When the scanner has power, the ping should be successful
	Check network settings of the PC, check if network card in the PC recognizes other network devices	Check if Scanner and PC are in the same network subnet Check Gateway IP	PC and Scanner **must** be in the same logical network segment
Connection to PC works, yet several seconds of delay in the scan profile	Connection or PC is too slow, check if Anti-Virus Software has effect.	Use faster PC – CPU clock 800 MHz minimum Use 100MBit network	**Use M2D-iVision for display**
Eth.-Link and 100Mbit are lit, but no connection	**Check network settings**	Check cable connection	Reset switches, restart PC could help
Slow connection	Check if your network card is 100 MBit	**Use other PC, use an Ethernet switch**	Do not use hubs
Can not flash with new firmware	**Check the RS-232 monitor prompt**	When RS-232 monitor prompt does not work: check RS-232 cabling	Rx and Tx pins inversed? COM port set to 115.200 Baud?
	Did you start with the batch-file?	Is the firmware file correctly set in the batch file?	Check error messages Did you use the correct binary file?
Flashing over the network Message: can not load firmware file	**Has scanner been set to programming mode?**	Power down and set programming mode switch, then power up again holding the Prog switch for a second	Make sure that you have the correct firmware file

Example code for Software engineers

Our software-engineers use Delphi and C++ (Visual Studio). Example code may be released in C++, Ruby or Delphi. We will not be able to supply code in VB or other dialects.

Disclaimer: MEL does not make warranty for correct function and completeness of the example code nor does MEL assume that code fragments are bug free. The use of the example code is at customers risk and with the sole responsibility of the customer's software engineers. MEL can not take any responsibility, that the code provided is free from third party rights, pending software patents or bug free.

Data format for register addresses and commands

MSB (Bit 7) is used as a differentiator between register addresses and command or data:
MSB (Bit 7) = "0" for all register addresses
MSB (Bit 7) = "1" for all commands and data
To change a value, transmit first the register number and then the new value. The register number remains set, until a new register number or a value is sent. It is not possible to read the registers. Dual registers: the value is active only when the high byte has been transferred.

Bit order

register	data	register	data
Lo-Byte	Lo-Byte	Hi-byte	Hi-byte
76543210	76543210	76543210	76543210
0xxxxxxx	1xxxxxxx	0xxxxxxx	1xxxxxxx

Software Examples

1. Switch on / Off FPGA-LED

```
        Lo-Byte     Hi-Byte
        0x0B        0x80  // LED on
        0x0B        0x81  // LED off
0x0B = register address (decimal 11)      0x80 = command: set LED on
```

2. Set Laser intensity in manual mode: set register 0x15 = 1

```
        Lo-Byte     Hi-Byte
0x00  0xff  0x01  0x87  0x3FF     // laser Off
0x00  0x80  0x01  0x80  0x000    // Laser full power
0x00  0x8F  0x01  0x84  0x0x20F // Laser 527 decimal; range = 0…1022; 1023 = 0ff
```

```
masking is done a prevention method:
Byte 0:      0x00                         register address
Byte 1:     (0x20F & 0x7F)    | 0x80      data of Lo-Byte
Byte 2:      0x01                         register address
Byte 3:     (0x20F>>7)& 0x7F | 0x80       data Hi-Byte

Used logic function terms
>>7   = shift right by 7 bit
&     = binary (bit by bit) AND
|     = binary (bit by bit) OR
0x7F  = mask            the mask is used to prevent false readings
```

📖 *Tipp:* *Lo and Hi-Byte can be sent either in separate chunks or in one "telegram".*
There will be no difference for the controller. A telegram may make up to 4 Byte.

The received register values are written to the registers as they were received from the incoming data stream.
There is no specific order of sending the register information, except a few ones, which provide a specific mode like set trigger mode on. The method of "writing" or "reading" register address and data applies to all of the following registers.

3. Command: 0x21 (dump)

The complete 64 EEPROM registers are transmitted
The Protocol version is set to 0x10h
The Sync raster is written as 8-Null Bytes
Starting with Byte 64 of the Scan-Data range, the Firmware-Version number is sent.

The length of the Firmware-String is variable. The ending character is 0x00h. In each block data is 2048 Bytes. In the end of the block, old data could be contained.
The transmission is triggered by the PC by sending „0x21" from the PC.

Data format of the packet:

Header =	Bytes 0 ... 65	including Sync = 8 Bytes 0x00h
Eprom Data	Bytes 66 ... 97	= Register of electronic unit (data related to scanner head, see page 14)
Eprom Data	Bytes 98 ... 129	= Register Eprom data (user data at status register 32..63, see page 17)
Firmware	Bytes 130 ...xxx	= String

After reception of the command 0x21h a packet of 2048 Bytes is sent to the PC. The packet contains the contents of all registers who can be read out, but no scan data. From Byte 64 of the Scan-data range, the firmware version is sent out as string. The end of the string is 0x00. The register contents are determined when switching on power of the electronic box. These values are stored until next reset (command 0x1F= Reset Ethernet) in memory.

4. examples of accessing registers, functions and data

These examples have been taken from the M2Dmini.c. For use as a complete package, download M2Dmini.c and header file M2Dmini.h from the MEL FTP-Server. Please consider updates and bug fixes, when you have earlier versions of M2Dmini. [version = as of 2006-08-10] Headlines in blue!

Disclaimer: *the given source code examples were taken from a collection of modules used with ISA hardware. Some of the commands may not work with the Ethernet hardware. Nevertheless these examples may show basic techniques how to read data from the scanner. In future releases of the documentation will be bundled into a separate documentation and be more specific considering this aspect.*
Source codes were given on a basis of "test and verify – then use it for free and on your own risk".

```
// Linux RTAI
#ifdef __RT__
#define inp(port) inb(port)
#define inpw(port) inw(port)
#define outp(adr,val)  outb(val,adr)
#endif /* __RT__*/

// read one byte from sensor (one try)
// return 0xFF if sensor busy of FiFO empty (nothing read)
int ByteFromM2FF(M2Dinfo *inf)
{
  if(inf->m2FirstByte){
     inf->m2FirstByte=0;
     inf->m2Word=inpw(inf->ioPort);
#ifdef debug
     M2Dwait(inf, wait_cnt);
     if(inf->bbf){
       if(!(remove_ffff && inf->m2Word==0xFFFF)){
         inf->bbf[inf->bbf_pos]=inf->m2Word;
         inf->bbf_pos++;
         if(inf->bbf_pos==BBF_ANZ) inf->bbf_pos=0;
       }
     }
#endif
     return inf->m2Word & 0xFF;
  }
  else{
    inf->m2FirstByte=1;
    return (inf->m2Word >> 8) & 0xFF;
  }
}
```

```
// read one byte from sensor
// try until some data found, return -1 if operation times out (error)
int ByteFromM2(M2Dinfo *inf)
{
   int val, emptyCnt=0;
   do{
      emptyCnt++;
      if(emptyCnt == 32767){
        return -1;
      }
      val=ByteFromM2FF(inf);
   } while(val == 0xFF);
   return val;
}
// low level write used by M2Dwrite
void M2Dwait(M2Dinfo *inf, int cnt)
{
   outp(inf->ioPort,value);
}
// low level wait used by M2DWrite
void M2Dwait(M2Dinfo *inf, int cnt)
{
  while(cnt--)
    inp(inf->ioPort+2);
}
// M2DWrite
// write to M2D-Register
// anz:      0 -> registerNr == command e.g. 0x1C – reset FIFO; value not used
//           1 -> write value to registerNr
//           2 -> write value to registerNR & registerNr+1, shift value to fit
//           low & high register; e.g. INTENSITY
void M2DWrite (M2DInfo * inf, int registerNr,int value, int anz)
{
   switch(anz){
   case 2:
       //first register
       ByteToM2 (inf, registerNr);
       M2DWait (inf,fio_Wait);
       ByteToM2(inf, (value & 0x7F) | 0x80);
       M2DWait(inf,fio_Wait),
       //register+1
       ByteToM2(inf,registerNr+1);
       M2DWait(inf,fio_Wait);
       ByteToM2(inf,((value >> 7)& 0x7F | 0x80);
       M2DWait(inf,fio_Wait);
       break;
   case 1:
       ByteToM2 (inf, registerNr);
       M2DWait (inf,fio_Wait);
       ByteToM2(inf, (value & 0x7F) | 0x80);
       M2DWait(inf,fio_Wait),
       break;
   case 0:
       ByteToM2 (inf, registerNr);
       M2DWait (inf,fio_Wait);
       break;
   }
}
```

```
// M2DSync
// return: 0 = ok, sync found           >0 = ok, sync after retval bytes found
// -1 = timeout                          -2 = no sync
// inf->sync
// sync [0] data format version          sync [1] status1
// sync [2] running number        sync [3] status2
int M2DSync (M2DInfo *inf)
{
      int
        i,
        val,
        syncCnt=0
        tryCnt=0
     while(1){
        val=ByteFromM2(inf);
        if(val == -1)
          return -1 // timeout
        if(val == 0){
          synCnt++;
        }
        else{
          if(synCnt >= 8){
#ifdef M2D_DEBUG
           if(synCnt > 8){
#ifdef __RT__
            printk("\n[%d]\n",synCnt);
#else
            printf("\n[%d]\n",synCnt);
#endif
           }
#endif
      // ---sync info 4 byte ---
      inf->sync[0] = val ;
      for(i=1 ; i<4 ; i++){
        val=ByteFromM2(inf) ;
        if(val == -1)
          return -1;  // timeout
        inf->sync[i] = val;
      }
      if(inf->sync[0]==3){
        //data format 3 : throw away useless encoder data here because of
        //special behaviour in trigger mode
      for(i=0;i<4;i++){
      val = ByteFromM2(inf); if(val == -1) return -1;
      }
    }
    return (tryCnt-8); // OK
    }
    else{
      synCnt = 0;
    }
  }
  tryCnt++;
  if(tryCnt == 8192){
    return -2; // snyc not found
  }
 }
}
```

```
// M2DReadFrame, use only after M2DSync suceeded
// read count points (4 or 5 bytes) of
// scan data and put values to x, z, and intensity
// return 0 = OK, -1 = timeout, -3 = data format not known
int M2DReadFrame(M2Dinfo *inf, int count, int *x, int *z, int *intensity)
{
  int i,v0,v1,v2,v3,v4;
  switch (inf->sync[0]){
  case 1: // data ver.1
  case 4: // like 1 but number of points depends on sensor
    // read 4 bytes / point
    if((inf->sync[1] & 1)==1){
       // linear
       for(i=0;i<count;i++){
         v0=ByteFromM2(inf); if(v0 == -1) return -1;
         v1=ByteFromM2(inf); if(v1 == -1) return -1;
         v2=ByteFromM2(inf); if(v2 == -1) return -1;
         v3=ByteFromM2(inf); if(v3 == -1) return -1;
         x[i]=v0+((v1 & 0x60)<<2)+((v3 & 0x07)<<9);
         z[i]=v2+((v1 & 0x1F)<<7);
         v3=v3 & 0xF8;
         intensity[i]=(v3<=128) ? (v3 & 127) << 1 : -((v3 & 127) << 1);
        }
     }
     else{
       // not linear
      for(i=0;i<count;i++){
         v0=ByteFromM2(inf); if(v0 == -1) return -1;
         v1=ByteFromM2(inf); if(v1 == -1) return -1;
         v2=ByteFromM2(inf); if(v2 == -1) return -1;
         v3=ByteFromM2(inf); if(v3 == -1) return -1;
         x[i]=v0+((v1 & 0x70)<<3);
         z[i]=v2+((v1 & 0x0F)<<7);
         intensity[i]=(v3<=128) ? (v3 & 127) << 1 : -((v3 & 127) << 1);
        }
     }
     break;
   case 3:
      //  encoder data already done in sync
   case 2:
      // read 5 bytes / point
      for(i=0;i<count;i++){
         v0=ByteFromM2(inf); if(v0 == -1) return -1;
         v1=ByteFromM2(inf); if(v1 == -1) return -1;
         v2=ByteFromM2(inf); if(v2 == -1) return -1;
         v3=ByteFromM2(inf); if(v3 == -1) return -1;
         v4=ByteFromM2(inf); if(v4 == -1) return -1;
         x[i]=v0+(v1<<7);
         z[i]=v2+((v3<<7);
         intensity[i]=v4;
        }
     break;
default:
return -3
}
return 0;
}
```

```
// DataToInt
// helper function to make integer from sensor status
// dat: array with status info
// cnt: array size
// return: calculated number
long DatToInt(int *dat,int cnt)
{
   int i;
   long res;
   res=0;
   for(i=0;i<cnt;i++){
     res = res | ((long)dat[i] << i*7);
   }
   return res;
}
// M2DStatus
// select status by num and return value
// Hint: avoid trigger mode
// return:
// 0 = ok, -1 = fail
// inf -> sync [3] = status
int M2DStatus (M2DInfo *inf, int num)
{
   int i,res,result;         // int res = M2DStatus(sens,3)
   result=-1;
   M2DWrite(inf,0x11,num,1); // select status register
   M2DWrite(inf,0x1C,0,0);   // clear FIFO
   for(i=0;i<5;i++){
      res = M2DSync(inf);
      if(res>=0 && (((inf->sync[1]>>1) & 0x3F) == num)){
         result=0;
         break;
      }
}
// M2DStatusInt
// select count status bytes by num and make integer
// Hint: avoid trigger mode
// return:
// 0 = ok,  -1 = fail
// value
int M2DStatusInt(M2DInfo *inf, int start, int count, long *value)
{
int i, res;
int dat[8];
res=-1 ;
if(count>8) count=8;
for(i=0;i<count;i++){
res= M2DStatus(inf,i+start);
if(res<0)
break;
dat[i] = inf->sync[3];
}
*value = DatToInt(dat,count);
return res;
}
```

```
// M2DHardwareInfo
// get some information about sensor hardware
// fills out the M2DHWInfo struct
int M2DHardwareInfo (M2DInfo *inf, M2DHWInfo *hw)
{
   int i, res;
   long tmp;
   do{
   res = M2DStatusInt (inf,32,2,&hw->ccdh); if(res<0) break;
   res = M2DStatusInt (inf,34,2,&hw->ccdh); if(res<0) break;
   res = M2DStatusInt (inf,36,4,&hw->ccdh); if(res<0) break;
   res = M2DStatusInt (inf,32,2,&hw->ccdh); if(res<0) break;
   res = M2DStatusInt (inf,40,2,&tmp); if(res<0) break;
   hw->amb=tmp;
   res = M2DStatusInt (inf,42,2,&tmp); if(res<0) break;
   hw->mb=tmp;
   res = M2DStatusInt (inf,44,2,&tmp); if(res<0) break;
   hw->sbAmb=tmp;
   res = M2DStatusInt (inf,46,2,&tmp); if(res<0) break;
   hw->sbEmb=tmp;
   res = M2DStatusInt (inf,30,1,&hw->opt1); if(res<0) break;
   if (hw->opt1 & 8 == 0) {
      hw->amb /=10.f;
      hw->mb /= 10.f;
      hw->sbAmb /=10.f;
      hw->abEmb /=10.f;
   }
   res = MwDStatusInt(inf,48,2,&hw->maxZ); if (res<0) break;
   res = MwDStatusInt(inf,50,2,&hw->maxX); if (res<0) break;
   res = MwDStatusInt(inf,2,1,&hw->controllerVersion); if (res<0) break;
   res = MwDStatusInt(inf,3,1,&hw->cameraVersion); if (res<0) break;
   res = MwDStatusInt(inf,0,1,&hw->temperature); if (res<0) break;
   if (hw->temperature>127){
      hw->temperature -= 128;
   }else{
      hw->temperature = ~hw->temperature + 1;
   }
 } while (0);
 return res;
}
// M2DFilter
// minimalistic filter, removes all definitive invalid values and
// converts to float
void M2DFilter (int anz, int *x, int *z, int *intens, int minIntens,
                  int *fanz, float *xf, float *zf)
{
int i, cnt;
cnt = 0;
for (i=0;i<anz;i++){
if (x[i]>0 && x[i]<4095 && z[i]>0 && z[i]<4095 && intens[i]>=minIntens){
Xf[cnt] = x[i];
Zf[cnt] = z[i];
cnt++;
}
}
*fanz = cnt;
}
```

example: // read out FiFo status and sensor temperature

```
*scanner = 0x18;
*scanner = 1 | 0x80;
      For (y = 0; y < 31 ; y++)
      {
            *scanner = y |0x80 ;
```

```
            Scanner_data.scan [uiBufferUARTTx] = *scanner ;
                  If (y==17)
                  ucRegister17Temp = *scanner ;
            uiBufferUARTTx++ ;
      }
      Scanner_data.scan[uiBufferUARTTx] = 0xFF
      uiBufferUARTTx++ ;
      *scanner = 126 |0x80 ;
      *scanner = 123 |0x80 ;
      Scanner_data.scan[uiBufferUARTTx + 0] = *scanner;
      *scanner = 124 |0x80 ;
      Scanner_data.scan[uiBufferUARTTx + 1] = *scanner;
      *scanner = 125 |0x80 ;
      Scanner_data.scan[uiBufferUARTTx + 2] = *scanner & 0x07;
      uiBufferUARTTx += 3 ;
      Scanner_data.scan[uiBufferUARTTx] = 0xFF;
      uiBufferUARTTx++ ;


      *scanner = 31 | 0x80;
      *scanner = 17 ; // Temperatur abfragen
      For (y = 0; y < 31 ; y++)
      {
            *scanner = y |0x80 ;
            Scanner_data.scan[y] = *scanner ;
            uiBufferUARTTx++ ;
      }
      scanner_data.scan[uiBufferUARTTx] = 0xFF ;
      uiBufferUARTTx++ ;
      For (y = 32; y < 64 ; y++)
      {
            *scanner = y |0x80 ;
            Wait_n(10, 20) ;    // the length[1] of the wait depends from
                                //CPU and system load!
            Scanner_data.scan[y] = *scanner ;
            uiBufferUARTTx++ ;
      }
      Scanner_data.scan[uiBufferUARTTx] = 0xFF ;
      uiBufferUARTTx++;

      *scanner = ucRegister17Temp | 0x80;
*scanner = 0x18;
*scanner = 127 | 0x80;
```

…

[1] Approximately 15 µsec of wait are necessary to make sure, that data is valid. This timing depends on the CPU.

Syntax of HDR shutter control

In HDR shutter mode, the scanner captures profiles with alternating shutter value. A High- and a Low value is defined by the software. The scanner captures profiles depending on the command sent before sending the shutter values for High and Low shutter capture. The HDR feature is unique for M2-iLAN scanners. Earlier models do not provide this feature.

Syntax:

<24h> = command: highest Bit not set!
Fro all following Bytes the highest Bit is set!
- -
<24h><80h> = deactivate
<24h><81h> = activate
- -
<80h> = each field is captured with different shutter value (= field switching mode)
<81h> = each second field is captured with a different shutter value (= full frame switching mode)

Example of a sequence:
B1 = shutter value 1
B2 = shutter value 2
Between both shutter values, for EACH field is switched
- - -
<24h><81h><80h><87h><D4h><80h><E5h> = B1=980 B2=101
- - -

Explanation of the syntax:
<24h><81h> = command "activate" // activate the HDR mode
<80h> = Definition "field switching mode"
<87h><D4h> = <Hi-Byte1><Lo-Byte1>
Shutter value Hi Byte: (X >> 0x07) | 0x80
Shutter value Lo-Byte: (X & 0x7F) | 0x80
... the value is = 980

<80h><E5h> = <Hi-Byte2><Lo-Byte2>
Shutter value Hi-Byte: (X >> 0x07) | 0x80
Shutter value Lo-Byte: (X & 0x7F) | 0x80
... the value is = 101

Ports

Ports are a part of the IP-address definition. When setting the working IP address, the port is defined accordingly. As a factory default setting at MEL, we always use port: 3000.
You can use “any” port. Please consider, that some services use ports: Internet (http) = port : 80.

Port addresses above 1024 are used not very often, but there are some exceptions like VNC (uses ports 5000 … 6000). Ask your system administrator for restrictions and guidelines for port numbers.

Ethernet WinSock Implementation

To communicate with a M2D Ethernet–Scanner so called Windows Socket „WinSock"-functions are used. These functions are part of all windows operating systems. Other operating systems provide similar functions.
„WinSock"-functions are encapsulated into a „ws2_32.dll"-file. These files belong to Windows.
The communication uses a TCP/IP-protocol with M2D Scanner working as a server. So the communication partner, in our case the PC, has to be set up as a client. Before using the network functions the „WSAStartup"-function has to be called. So the regular functions from „WinSock" can be used.

Next a valid TCP-SOCKET (a kind of object) has to be received from the system. This is done by calling the function „socket". If there is a valid SOCKET, it is possible to establish a connection to the M2D Scanner. This will be made by the function „connect". After a successful call of this function, PC and scanner are connected and it is possible to exchange data with the two commands „send" and „recv".

At the end of each communication stream all memories and sockets have to be released. This is done by using the two functions „closesocket" and „WSACleanup" at the end.

Example: ("*C*") using WinSock and Scanner functions:

```
// receiving block size should be modulo 2.048
// the Scanner sends blocks in size of 2.048 Bytes.
#define TCPBUFSIZE 2048
       // structure  for WinSocket
WSADATA wsaData;
       // socket-Variable
SOCKET sTCP;
       // receive buffer of the size mentioned above
char chBuffer[TCPBUFSIZE];
       // number of received bytes from scanner
DWORD dwReceived = 0;
       // permanently try to reach socket, when connection is broken
BOOL bRunSocket = TRUE;
  // permanently try to establish connection, when receive function returns an error
BOOL bRunConnect = TRUE;
       // permanently receive data from scanner
BOOL bRunRead = TRUE;
       // define TimeOut in [ms] for receive function
       // after this time, the "recv"-Function returns with dwReceived=0
       // then the link will be locked and transmission established
DWORD dwRecvTimeOut = 10000;
       //  try to run WinSocket  Version 2.1
if (WSAStartup (MAKEWORD(2, 1), &wsaData) != NULL)
{
   AfxMessageBox("Fehler: WSAStartup", MB_OK | MB_ICONEXCLAMATION, NULL);
   return;
}
       //Structure for the "connect"-command
SOCKADDR_IN serv_addr;
       // MUST have the value "AF_INET"
serv_addr.sin_family = AF_INET;
       // transmit Port-Number of Scanner
serv_addr.sin_port = htons(atoi("3000"));
       // transmit IP-Adresse of Scanner
serv_addr.sin_addr.S_un.S_addr = inet_addr("192.168.123.224");
while(bRunSocket)
{
   bRunConnect = TRUE;
        // get Socket for TCP=SOCK_STREAM
   sTCP = socket(AF_INET, SOCK_STREAM, 0);
         // socket-Error?
   if (sTCP == INVALID_SOCKET)
   {
      sTCP = 0;
      bRunConnect = FALSE;
      TRACE("SocketError\n");
   }
```

```
      // set TimeOut for "recv"-Function
   setsockopt(sTCP, SOL_SOCKET, SO_RCVTIMEO, (const char*)&dwRecvTimeOut, sizeof(int));

   while(bRunConnect)
   {
        // establish connection
      if (connect(sTCP, (SOCKADDR*) &serv_addr, sizeof(SOCKADDR)) == INVALID_SOCKET)
      {
               // when it is not possible to establish a connection…
      }
      else
      {
               // connection established: get data …
         bRunRead = TRUE;
         while(bRunRead)
         {
            dwReceived = recv(sTCP, chBuffer, TCPBUFSIZE, NULL);
            if ((dwReceived == 0) || (dwReceived == INVALID_SOCKET))
            {
               // probably the connection has been disrupted, make sure,...
               // to establish new connection
               bRunRead = FALSE;
               bRunConnect = FALSE;
               closesocket(sTCP);
            }
            else
            {
               // here are the received data
               // the total number of received data is in dwReceived
            }
         }
      }
   }
   // finally free up the socket
   closesocket(sTCP);
   WSACleanup();
}
```

To send data to the scanner the „send“ – function is used.. Similar to example with the „recv“ – function it is nessesary to have an existing connection to the scanner, this means to have a valid socket.

```
// create Data-Buffer
// this Data-Buffer has already a command to set the Scanner
// in Single-Shot-Mode
char chBuffer[2] = {0x14, 0x01};
send(sTCP, chBuffer, 2, NULL);
```

Tipp: more information about Windows Socket functions and network programming can be found in the documentation of the Windows operating–system.
the Scanner commands are listed in the register tables of this manual.

UDP Implementation

Scanner-IP/Scanner-Port, Rechner-IP/Rechner-Port

The Scanner electronic system sends scan data to computer = PC-IP: PC-Port-Number
The computer sends commands to the scanner electronic system = Scanner-IP: Scanner-Port-Number

In the Scanner-electronic unit a UDP-Server is active, as well as in the PC. The scanner data format for UDP transmission is the same as for TCP. The interface register description is valid for UDP as well as for TCP.

IP-Programming for UDP-transmission protocol:

```
chTemp[0] = 0x22;   // command
chTemp[1] = 0x55;   // command
chTemp[2] = 0xAA;   // command
chTemp[3] = 0x55;   // command
chTemp[4] = 0xAA;   // command
chTemp[5] = 0x55;   // command
chTemp[6] = 0x00;
chTemp[7] = 0x00;
chTemp[8] = 0x00;
chTemp[9] = 0x00;
chTemp[10] = 0x00;
//RechnerIP (IP of the PC)
chTemp[11] = atoi("192");
chTemp[12] = atoi("168");
chTemp[13] = atoi("123");
chTemp[14] = atoi("117");
//Gateway
chTemp[15] = atoi("192");
chTemp[16] = atoi("168");
chTemp[17] = atoi("123");
chTemp[18] = atoi("1");
//Subnetzmaske (subnet mask)
chTemp[19] = atoi("255");
chTemp[20] = atoi("255");
chTemp[21] = atoi("255");
chTemp[22] = atoi("0");
//ScannerIP (working IP)
chTemp[23] = atoi("192");
chTemp[24] = atoi("168");
chTemp[25] = atoi("123");
chTemp[26] = atoi("224");
//ScannerPort
chTemp[27] = atoi("3000") & 0x00FF;
chTemp[28] = (atoi("3000") & 0xFF00) >> 8;
//RechnerPort (port of the PC)
chTemp[29] = atoi("3000") & 0x00FF;
chTemp[30] = (atoi("3000") & 0xFF00) >> 8;
chTemp[31] = m_bTCPUDPFlag;

m_pCEthernetScannerDlg->SendToCurrentSelectedIP((unsigned char*)chTemp, 32);
```

Note: the UDP transmission protocol has been implemented in a new project of the MEL EthernetScanner demo software. Typical TCP functions have been dropped as for example TCPClientThread (global)and EthTCPClientXReceivedData (global).

This function receives the raw data from the Scanner and checks if it is correct:
void EthUDPServerReceivedData(CWnd *theDlg, char *chBuffer, DWORD dwBuffer)

This function checks the protocol version and unpacks the data accordingly:
void CUDPScannerDlg::ScannerViewScan(unsigned char *ucBuffer, DWORD dwBuffer, DWORD dwStatus, DWORD dwModus)

This function displays the profile:
void CUDPScannerDlg::ShowScannerXZI(DWORD *dwBufferX, DWORD *dwBufferZ, DWORD *dwBufferI, DWORD dwBuffer, DWORD dwPosGeber, DWORD dwPosGeberRichtung, DWORD dwProtokoll, DWORD dwLiniarisierung)

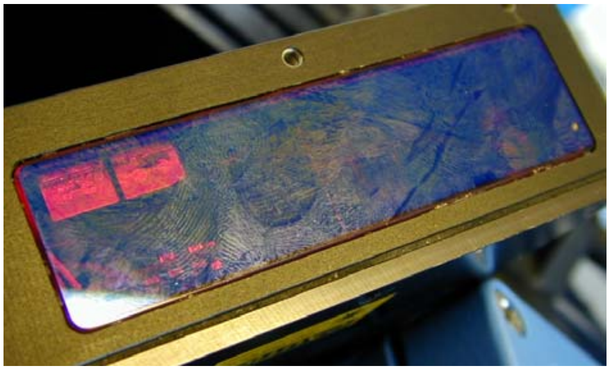
This function sends data to the scanner:
void CUDPScannerDlg::SendToScanner(CString strSend)

On the GUI, a double click takes you to the individual functions.
There you will find a description for the commands.

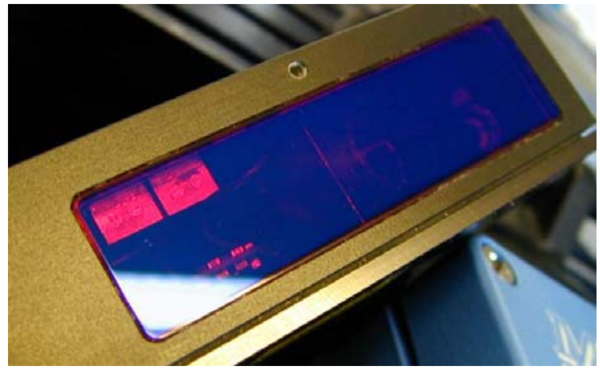
This function starts and stops the UDP server on the PC:
OnCheckEthudpscannerstart

Maintenance

The M2D Laser-Scanners are virtually maintenance free. The Scanners are optical measurement tools, which sometimes need cleaning and inspection from time to time. Cleaning the front window should be done with a soft, clean cloth. Do not use scratching tools, concentrated solvent liquids are aggressive chemicals. Dirt should be removed with 15 ... 20 % of isopropyl alcohol + distilled water or cleaning benzine. Finger prints will cause additional noise in the scan profiles, keep your front windows clean!



Picture: dirty Front window



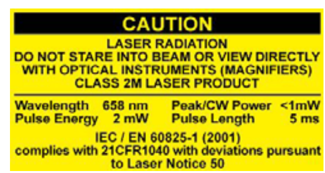
picture: clean Front window

Laser Safety

All *standard* M2-iLAN-Laser scanners models are class 2 / 2M. The emitted laser radiation is below 1 mW. According to international standards, this amount of laser radiation is not dangerous to the eye. When the laser beam touches the human skin, this is not giving a problem.

Scanner models having higher laser power are clearly marked with sticker stating the maximum laser power and laser safety class.

The scanner has a optic system dispersing the laser energy: the line projector distributes the laser energy to an angle of 20 or 30°. This means, the energy gets very weak in a distance of a few inches from the laser front window. When the laser is shining against a mirroring device, the laser power will be reflected and highly visible to the eye, yet not dangerous.



Products of laser class 2 have a yellow / black label as pictured on the right side.

📖 *a few general rules which may help to use controls, adjustments of performance and recommended practice to avoid any hazardous radiation exposure:*

- never do look into the laser beam
- never direct the laser beam against any other person
- before cleaning switch off the Laser Scanners
- use a piece of paper when you search for the laser beam
- never use a mirroring device close to the laser output
- always use *automatic shutter* mode
- provide a shield for the scanners vision range (protection cover or dark plastic glass)

Picture on the right side:
Position of Laser output and labels on the scanner

The Laser output is marked with an arrow on the case.
The yellow label indicates laser power and wavelength.
The table on the following page lists the typical laser parameters.

Laser Power

M2-iLAN Model	10/13/15	20/10/13	40/20/26	60/30/40	80/40/60	120/60/80	220/120/160
Max. Laser power * [mW]	0.14	0.69	0.35	0.35	0.38	0.41	0.23
Average Laser Power [mW]	0.10	0.59	0.23	0.26	0.27	0.35	0.17
Wavelength typ. ** [nm]	658	658	658	658	658	658	658
Pulse repetition rate [Hz]	93.5	93.5	93.5	93.5	93.5	93.5	93.5
Duty cycle max. [% on]	89	89	89	89	89	89	89
Duty cycle min. [% on]	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1
Line optic projector [° opening angle]	30	30	30	30	20	20	30

Type of Power Meter : OPHIR PD 200
distance: < 100 mm

* optical output Laser power has been measured under worst condition. The measurement values in the table specify the maximum power measured in a distance smaller than 100 mm, as specified in the international standards (FDA, DIN, EN, ISO).
** Depending on model and technical requirements the wavelength can vary from 645 up to 675 nm

For special purpose the wavelength can be 690 nm. For these special models, all other laser parameters and controls are the same. These products will be equipped with a special label defining the wavelength.

Environment

Sunlight

When installing the Laser-Scanners, check if there is direct sunlight shining into the receiver (camera window). Morning and evening *sunlight* has a lot of red light passing through the filters. This may cause problems for the scanner.

Water

Water drops make optic distortions. Avoid water on the front windows. Air streams should be used to push away the water before the measurement is started. When water sparkles come on to the front window, the front window shall be cleaned from time to time to avoid that the front window gets dirty.

Changes and additions

Nr.	Date	Change / addition
1	Oct 2008	Rev.4 hardware, protocol version = 3 (protocol 1 and 2 not available)
2	Oct 2008	HDR shutter control
3	Nov 2008	UDP protocol additional, can be set alternatively instead of TCP
4	Jan 2009	Firmware Rev.2.0.53
5	Feb 2009	Firmware Rev.2.0.55 improvements in profile recognition algorithms – less sensitive to odd reflections
6	Mar 2009	Flashing Controller Firmware over the network
7	Dec 2009	Firmware Rev.2.0.59 / java applet Beta – current Beta rev. 2.0.64