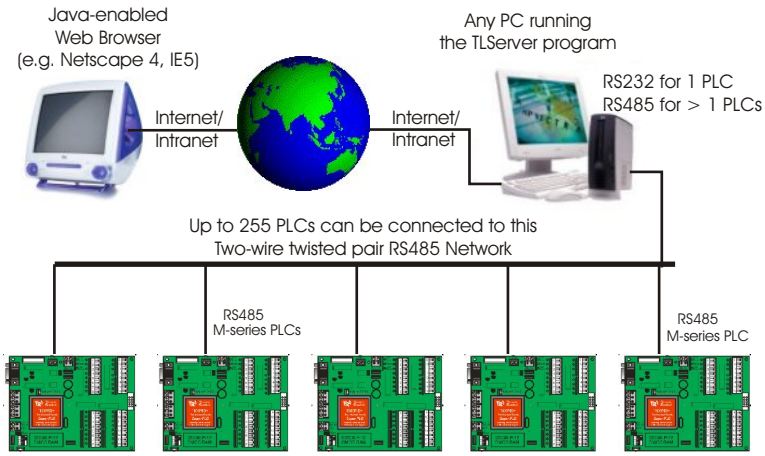
Internet Programmable Super PLCs



Since their launch in 1998, the extremely powerful and easy to program M-series PLCs have become the definitive choice among many OEMs. And for good reasons too.

Our proprietory "Ladder+BASIC" hybrid programming language is highly popular with users. The many features offered by the M-series PLCs such as Analog and digital I/Os, PWMs, stepper motor controls, Interrupts, high speed encoder inputs, real-time clock, PID control, Human-Machine Interface (HMI) , MODBUS compatibility etc are also welcomed. And the affordably low cost of our PLCs is a real bonus.

Now, we are topping all the above with the debut of our Internet-Programmable Super PLCs. Yes, the day has arrived when more and more clients expect their production equipment to be monitored or programmed remotely on-line round the clock. And we can make that a reality now.

Our latest Internet TRiLOGI client/server software enables our M-series PLCs to be fully programmable over the Internet or within your corporate Intranet. Even better, Internet TRiLOGI is so easy to use that it won't take you more than 10 minutes after opening our Internet PLC starter kit to start monitoring your PLC via the Internet!

In addition, the same TRiLOGI "Ladder+BASIC" client/server editor software is being used for both local PLC programming as well as remote programming via the Internet. Therefore, you don't need special training to remotely program the PLC via the Internet. In fact, you get to enjoy the same user interface and the full power of the TRiLOGI software whether it is locally connected or over the Internet!

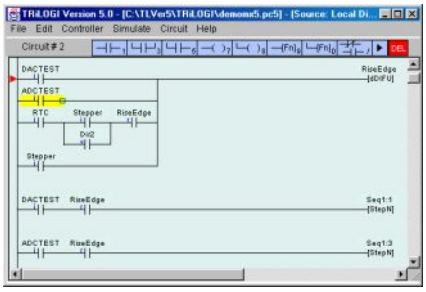
The Internet TRiLOGI software also works as an **email** post office to enable any connected PLCs to send email reports to anybody. The email content and the event trigger for sending the email is completely defined by the programmer. **Remote data logging** has never been easier!

Now start talking about Smart Machines and cutting edge control technology!!!

Benefits of Internet Programmability

- ★ Remote troubleshooting - Save big on travel cost
- ★ Remote Software Update - Upgrade user's equipment while keeping your source code secret!
- ★ Program the PLC to email event-specific reports periodically or upon certain event occurrence
- ★ Set 3 different access levels ("Visitor", "User" and "Programmer") to unlimited number of users

TLServer


TRiLOGI Client

Put Your PLC On-Line in 4 Easy Steps!

1. Connect the M-series PLC's RS232 port to any Windows or Linux PC or workstation.
2. Connect the PC to the Internet using any common networking method, including dialup, DSL, cable modem or T1 line.
3. Run the "TLServer" software on the PC (TLServer is part of the Internet TRiLOGI Client/Server suite). Note down the IP address of the PC shown on the TLServer's screen. This PC now becomes a web-server which serves up the TRiLOGI client and data files when accessed by a web-browser from other computers on the Internet.
4. Use a Java web browser such as Netscape 4 or IE5 from any computer on the Internet or Intranet. Enter the URL of the TLServer computer, e.g.

   http://101.223.45.80:9080

   The TRiLOGI client program will start running. The Client software comprises the ladder logic editor, compiler and simulator software that gives you complete programming control of the PLCs.

That's it! If you are programming the PLC locally, you can run the TRiLOGI client on the same PC as the TLServer. Simply use a localhost IP Address: http://localhost:9080 to access the PLC. Everything else works exactly the same way! The PC does NOT need to be connected to the Internet or Intranet for local programming. In other words, the Internet TRiLOGI works like a normal Window-based PLC software.

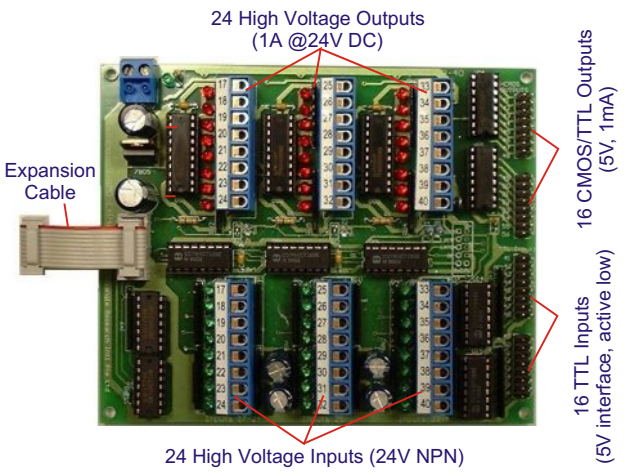
The Embedded PLC Specialist! - Since 1993

Visit us at: http://www.lt-automation.com

Standard Features on All M-series PLCs

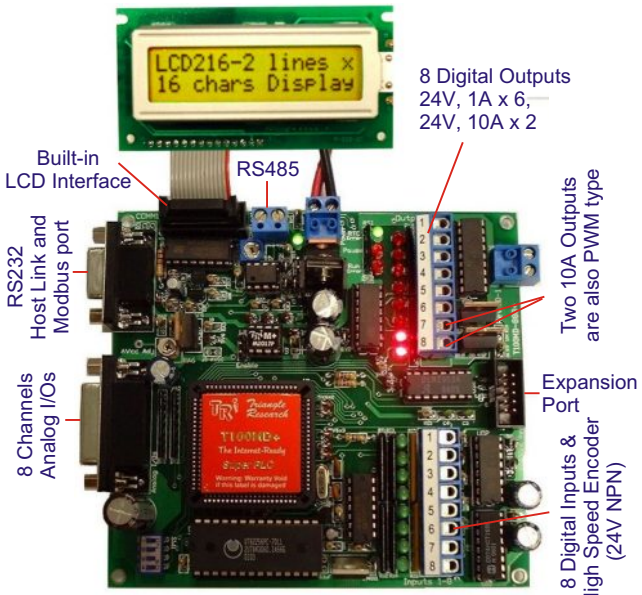
1) Up to 8 channels of 10-bit Analog Inputs.
2) Up to 2 channels of 8-bit Analog Outputs.
3) 2-channel programmable Motion Controllers for controlling stepper motors up to 20,000 pps.
4) 2-channel Pulse Width Modulated (PWM) outputs (share with 2 digital outputs).
5) 2-channel 32-bit High Speed Counters (HSC) count up to 10,000 Hz (share with 4 digital inputs).
6) 4-channel Interrupt Inputs (share with the 2 HSCs).
7) 2-channel pulse measurement inputs capable of measuring frequency and pulse-width of incoming pulses up to 10,000 Hz. (share with 2 digital inputs)
8) Real time Clock/Calendar for programming multiple scheduled ON/OFF events.
9) 6016 Words (16-bit) EEPROM Program Memory.
10) 1700 Words (16-bit) EEPROM for user's data.
11) Built-in 16 channels of PID-computation engines let T100MD+ PLCs directly provide Proportional, Integral and Derivative (PID) type digital control for process automation.
12) One Independent RS232 port for connection to a host PC for programming or monitoring.
13) One independent RS485 port for networking or for connecting to external peripherals such as LCD display and RS485-based analog I/O cards, etc.
14) Industry Standard Protocols: Both RS232 and RS485 serial ports support multiple communication protocols:
    i) Native ASCII based Host Link Commands.
    ii) MODBUS RTU protocols
    iii) MODBUS ASCII Protocols
    iv) OMRON C20H Host Link Commands.
    v) EMIT 3.0 protocol by emWare Inc.
15) Watch-Dog Timer (WDT) which resets the PLC if the CPU malfunctions due to hardware or software error.

EXP4040: Expand 40 Inputs, 40 Outputs



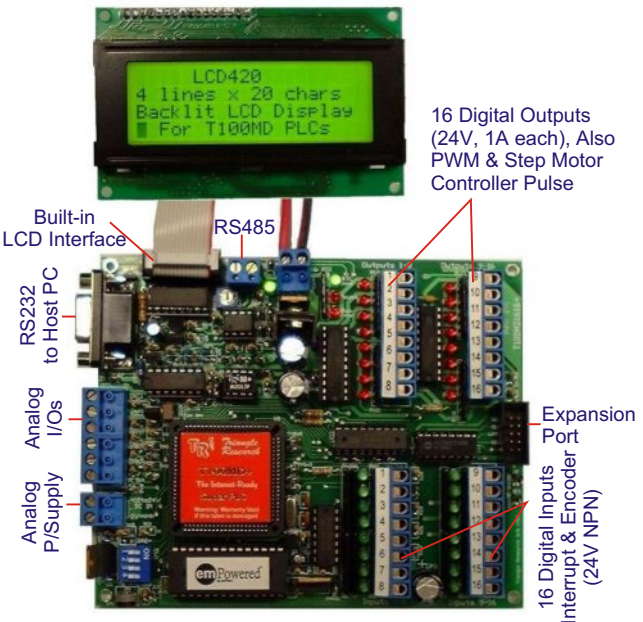
T100MD888+: 8 Analog I/O, 8 DI, 8 DO

10A PWM x 2 Channels



T100MD1616+: 16 DI, 16 DO, 4 AI, 1 AO

0-1V Analog Inputs, 0-20mA Analog Outputs



MX-RTC

Battery-backed Real Time Clock Module for T100MD and T100MX PLCs.

MODBUS is a trademark of Schneider Automation Inc.

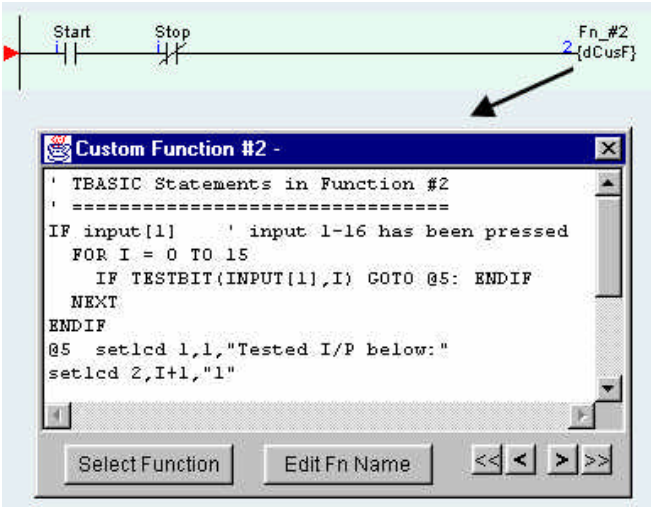
M-series PLC Selection Table

	T100MD888+	T100MD1616+	T100MX3224R+	T100MX4832+
Digital Input	**8** (24V npn)	**16** (24V npn)	**32** (24V, opto)	**48** (24V, opto)
Digital Output Max. Current	**8** 24V, 1A npn x 6, 24V,10A npn x 2	**16** 24V, 1A npn x 16	**24** 24V, 1A npn x 16 250V, 2A Relay x 8	**32** 1A npn x 32
Analog I/O A.I. Interface A.O. Interface	**8** (8 or 6, 0 to 5V) (0 or 2, 0 to 5V)	**4** In / **1** Out 0-1V x 2, 0-5V x 2 0-20mA x 1	**8** (8 or 6, 0 to 5V) (0 or 2, 0 to 5V)	**8** (8 or 6, 0 to 5V) (0 or 2, 0 to 5V)
PWM (current)	**2** (10A)	**2** (1A)	**2** (1A)	**2** (1A)
Stepper Motor Controller	**2** (20,000 pps)	**2** (20,000 pps)	**2** (20,000 pps)	**2** (20,000 pps)
High Speed Counter	**2** (10KHz)	**2** (10 KHz)	**2** (4 KHz)	**2** (4 KHz)
RS232 ports Configuration	**1** (DCE)	**1** (DCE)	**2** (1 DCE and 1 DTE)	**2** (1 DCE, 1 DTE)
RS485 port	**1**	**1**	**1**	**1**
14-pin LCD port	**1**	**1**	-	-
Max. Expansion I/O (Digital)	Additional 80 In, 80 Out	Additional 80 In, 80 Out	-	-
Dimension (L x W x H)	4.825" x 4.45" x 1.0"	5.425" x 4.45" x 1.0"	7.2" x 4.95" x 1.0"	8.55" x 4.95" x 1.0"
Operating Temperature	0-70°C	0-70°C	0-70°C	0-70°C

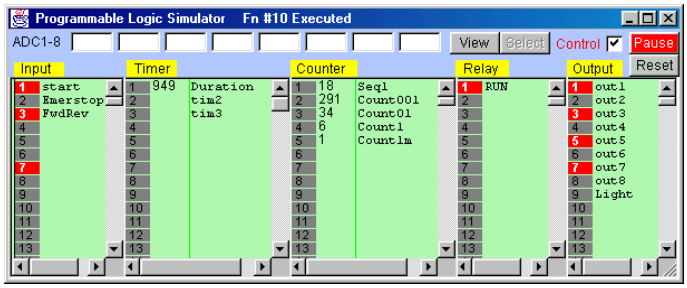
The single most important advantage of the M-Series PLC is that they are **INCREDIBLY EASY** to program - **We guarantee It!** You will find that programming the T100MD+ using our TRiLOGI software is much easier than any other similar jobs you have ever experienced!!

There is no need for long training sessions just to learn to harness the power of all the built-in hardware. The unprecedented ease of programming is delivered by our highly acclaimed programming Editor, Compiler and Simulator software – TRiLOGI. This software cleverly integrates the industry's two most popular programming languages - Ladder Logic and BASIC into a single, seamless entity. Like all PLCs, a T100MD+ runs industry-standard ladder logic programming language. But its biggest breakthrough enables you to easily create your own "Custom Function" using the very popular BASIC language and then connect the Custom Function to the ladder program as a function coil which will be executed when the execution condition is true.

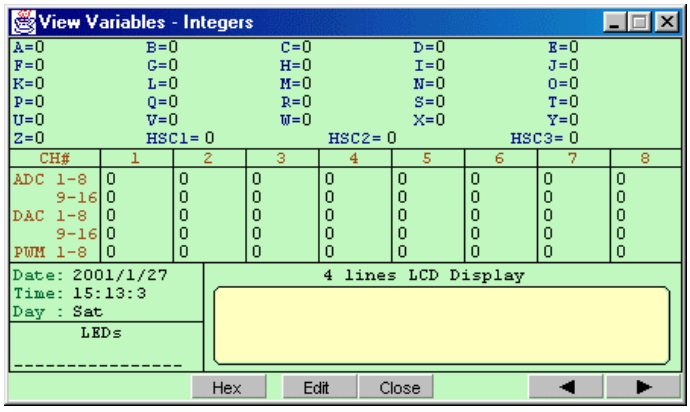
Think of it - you can use the efficient, easy to understand relay ladder language for programming the ON/OFF logic. At the same time, the full power of BASIC language is available for handling computationally intensive tasks such as 32-bit integer arithmetic, string processing. The availability of many powerful built-in functions makes programming its special I/Os a breeze.



Full simulation of all TBASIC commands is supported using the built-in **Simulator**, which allows extensive testing of your program before connecting to the PLC. Try the FREE software today at: **http://www.tri-plc.com** and you will be completely convinced.



TRiLOGI's On-Line Monitoring mode lets you view the values of ALL the internal variables and the logic states of ALL the I/Os, timers and counters instantly and continuously in real time. Break-points can be set in your software to pause the PLC for you to examine the internal state of affairs. Logic states of I/Os and values of variables can also be modified from the host, giving immense flexibility to program development.



Test Run TRiLOGI from our website. No Installation required. The software runs directly off your browser!

http://www.lt-automation.com

THE TBASIC LANGUAGE COMPONENTS

1. Integer Variables

The TBASIC compiler in TRiLOGI Version 4/5 supports full 32-bit integer computations. All integer constants are treated as 32-bits and there are 26 variables (A to Z) that are 32 bits in length. These variables can represent numbers between -2^{31} to $+2^{31}$. The remaining system variables and data memory DM[n] are all 16-bit variables which means that they can only store numbers between -32768 to +32767. These 16-bit variables are:

DM[1]..DM[4000] (Total = 4000), input[n], output[n], relay[n], timerbit[n], ctrbit[n], timerPV[n],ctrPV[n], HSCPV[n], (High Speed Counter). TIME[n], DATE[n].

However, all numeric computations in TBASIC are carried out in 32-bit signed integers, regardless of the bit-length of the variables involved in the numerical expression. All variables used in TBASIC are "GLOBAL variables - this means that all these variables are shared and accessible by any of the 128 custom functions.

2. Integer Operators

a) Arithmetic Operators: +, -, *, /, MOD

b) Bitwise Logical Operators:

Symbol	Operation	Example
&	logical AND	IF input[1] & &H02 ...
|	logical OR	output[1] = A | &H08
^	Exclusive OR	A = RELAY[2] ^ B
~	logical NOT	A = ~timerPV[1]

c) Relational Operators : Used exclusively for decision making expression in statement such as IF expression THEN ..... and WHILE expression ....

Symbol	Operation	Example
=	Equal To	IF A = 100
<>	Not Equal To	WHILE CTR_PV[0]<> 0
>	Greater Than	IF B > C/(D+10)
<	Less Than	IF TIME[3] < 59
>=	Greater Than or Equal To	WHILE X >= 10
<=	Less Than or Equal To	IF DM[I] <= 5678
AND	Relational AND	IF A>B AND C<=D
OR	Relational OR	IF A<>0 OR B=1000

3. String Data

A string is a sequence of alphanumeric characters (8-bit ASCII codes) which collectively form an entity. A string constant may contain from 0 to 70 characters enclosed in double quotation marks. e.g.

"TBASIC made PLC numeric processing a piece of cake!"
"$102,345.00"

TBASIC supports a maximum of 26 string variables A$, B$ ... Z$. Each string variable may contain from 0 (null string) up to a maximum of 70 characters.

4. String Operators

a) Assignment Operator: A string variable (A$ to Z$) may be assigned a string expression using the assignment statement:

```
A$ = "Hello, Welcome To TBASIC"
Z$ = MID$(A$,3,5)
```

b) Concatenation Operators: Two or more strings can be concatenated (joined together) simply by using the "+" operator. e.g.

```
M$ = "Hello " + A$ + ", welcome to " + B$
```

If A$ contains "James", and B$ contains "TBASIC", M$ will contain the string: "Hello James, welcome to TBASIC.

c) Comparison Operators: Two strings may be compared for equality by using the function STRCMP(A$,B$). However, the integer comparator such as "=", "<>", etc cannot be used for string comparison.

5. TBASIC Statements and Functions

Most of the statements and functions of the powerful TBASIC language is tabulated below together with brief explanations for their purposes.

Command	Purpose
ABS(*x*)	To return the absolute value of *x*
ADC(*n*)	To return the value from the A/D Converter #*n*.
ASC(*x$,n*)	To return the ASCII code for the *n*th char. of *x$*.
CALL *n*	To call another Custom Function CusFn #*n*.
CHR$(*n*)	To convert a number *n* into its corresponding ASCII character. *n* must be between 0 and 255)
CLRBIT *v,n*	To clear the Bit #*n* of the integer variable *v* to '0'.
FOR ... NEXT	To execute a series of instructions for a specified number of times in a loop.
GetCtrSV (*n*)	Return the Set Value (S.V) of Counter or Timer #n.
GetTimerSV (*n*)	
GETHIGH16(*v*)	returns the upper 16-bit of a 32-bit integer variable
GOTO @ *n*	To branch unconditionally to a specified line with label @*n* within the present Custom Function.
HEX$(n)	To return a hexadecimal string expression of *n*.
HEXVAL(*x$*)	To convert hexadecimal number in string x$.
HSTIMER *n*	To define PLC Timer #1 to #n as "High Speed Timers" (HST) with time-base of 0.01s.
HSCDEF *ch, fn,v*	Enable and set up parameters for the High Speed Counters channel *ch*.
HSCOFF *ch*	Disable High Speed Counter #*ch*
IF .. THEN .. ELSE .. ENDIF	To make a decision regarding program flow based on the result returned by an expression
INPUT$(*ch*)	To get a string from communication port *ch*.
INTRDEF *ch, fn,e*	Enable Interrupt Input channel *ch* and the interrupt service routine CusFn # *fn*
INTROFF *ch*	Disable Interrupt Input channel *ch*.
LEN(*x$*)	To return the number of characters in *x$*.
LOAD_EEP(*addr*)	To obtain16-bit integer value from the EEPROM.
LSHIFT *i,n*	To shift 1 bit to the left the 16-bit integer variable *I*,
MID$(*x$,n,m*)	To extract a string of *m* characters from *x$*, beginning from the *n*th character
PAUSE	To set a breakpoint for executing the CusFn
PIDdef *ch, lmt, P,I,D*	To set up the parameters for a Proportional, Integral and Derivative (PID) Controller function.
PIDcompute (*ch,E*)	Computes the output for the error E using the P,I,D value set by the PIDdef command.
PRINT# *n x$;.. y; z....*	To send the string formed by parameter list (*x$; y; z*) to communication port # *n*.
PMON *ch*	PMON enables Pulse Measurement Function at channel #*ch*. PMOFF disables the channel
PMOFF *ch*	
PULSEFREQUENCY	Return in Hz the frequency of the last input pulse ;
PULSEPERIOD	Return in milli-seconds the width and period of the
PULSEWIDTH	input pulses arriving at channel ch.
REFRESH	To immediately refresh of the physical outputs.
RESET	To perform a software reset of the PLC
RETURN	Unconditionally return to the caller function.
REM (or ')	Put explanatory remarks in the program.
RSHIFT *i,n*	To shift the integer variable *i* 1 bit to the right.
SAVE_EEP *data, addr*	To store a 16-bit integer *data* in the user's definable EEPROM address *addr*.
SETBAUD *ch, baud*	To set the communication "Baud Rate" of the PLC's serial channel #ch
SETBIT *v,n*	To set the bit #*n* of the integer variable *v* to '1'.
SetCtrSV *n, val*	Change the Set Value (S.V,) of the Counter #*n*
SetTimerSV *n, val*	or Timer #n to *value*
SETDAC *n, x*	To set D/A converter #*n* to *x*
SETTHIGH16 *v,d*	To assign the upper 16-bit of a 32-bit variable *v*
SETPWM *n, x, y*	To set PWM channel #*n* with duty cycle *x%*.
STEPCOUNT(*ch*)	Track the number of steps sent since STEPMOVE.
STEPSPEED *ch, pps, acc*	Set the speed *pps* and *acc* for the PLC's stepper motor controller (pulse-generator) channel #*ch*.
STEPMOVE *ch, count, r*	Activate the stepper motor pulse generator channel #*ch* to output *count* number of pulses.
STEPSTOP	Immediately terminate Stepper controller #n
STR$(*n*)	Convert the number *n* into a string expression.
STRCMP(*A$, B$*)	Comparison of two strings: A$ and B$.
STRUPR$(*A$*)	To return all upper-case copy of A$.
STRLWR$(*A$*)	To return all-lower-case copy of A$.
TESTBIT *(v,n)*	To return the logic state of bit #*n* of the variable *v*.
WHILE *expression* .... ENDWHILE	To execute a series of statements in a loop as long as a given condition is true.
VAL(*x$*)	To return a value of a number in the string *x$*.