M-PLC
Programming Interface

PLC programming interface according to IEC 61131-3

M-PLC is a complete development environment for programming the M1 controller in accordance with IEC 61131-3. The M-PLC puts a simple approach to the powerful IEC language at the disposal of the PLC programmer. Use of the edit and debugging functions is based on the proven development program environments of advanced programming languages.

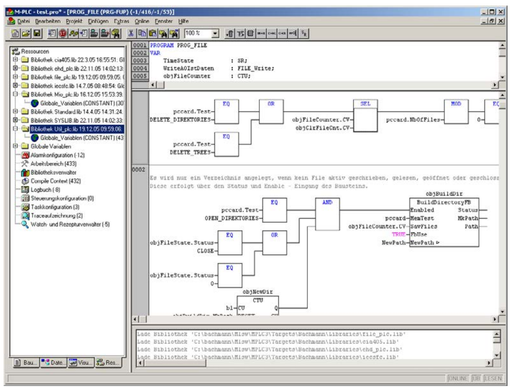
- Easy entry into IEC 61131-3
- All languages defined in IEC standard 61131-3 are supported (instruction list, structured text, ladder diagram, function block diagram, continuous function chart, sequential function chart).
- Editor and debugging functions
- Provided libraries with pre-finished and tested functions

Programming

- Editors for programming in all IEC 61131-3 languages

IL	Instruction List
ST	Structured Text
LD	Ladder Diagram
FBD	Function Block Diagram
SFC	Sequential Function Chart
CFC	Continuous Function Chart

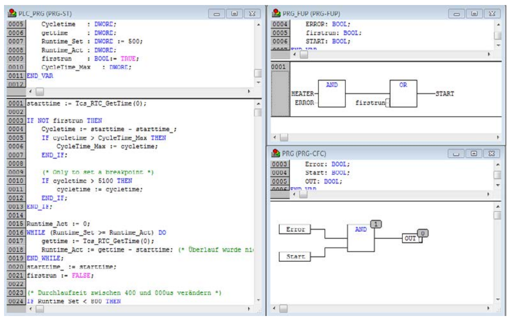
- Syntax highlighting for key words defined in IEC 61131-3
- Graphic project navigation bar (based on Windows Explorer)
- Support of all elementary IEC 61131-3 data types to 64 bit
- Support of fields, structures and pointers
- Graphic controller and task configuration
- CAN objects from *.dcf files in M-PLC can be applied as symbols
- PB objects from *.2bf files in M-PLC can be applied as symbols
- Library management for creation and management of libraries
- Watch manager and recipe manager (freely definable variable monitoring)
- M-PLC can be started several times (multiple projects open and online in parallel)
- Automatic project backup and creation of backup
- Project archiving on the controller
- Password protection for projects
- Offline simulation
- Online help
- Data exchange system-wide via SVI/SMI interface programming


M-PLC - test.pro* - [PROG_FILE (PRG-FUP) (-1/416/-1/53)]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Ressourcen
Bibliothek cia405.lib 22.3.05 16:55:51
Bibliothek ehd_plc.lib 22.11.05 14:02:13
Bibliothek file_plc.lib 19.12.05 09:59:05
Bibliothek iecsfc.lib 14.7.05 08:48:54
Bibliothek Mio_plc.lib 16.12.05 15:53:39
Globale_Variablen (CONSTANT) (30)
Bibliothek Standard.lib 14.4.05 14:31:24
Bibliothek SYSLIB.lib 22.11.05 14:02:33
Bibliothek Util_plc.lib 19.12.05 09:59:06
Globale_Variablen (CONSTANT) (43)
Globale Variablen
Alarmkonfiguration (-12)
Arbeitsbereich (433)
Bibliotheksverwalter
Compile Context (432)
Logbuch (-8)
Steuerungskonfiguration (0)
Taskkonfiguration (3)
Traceaufzeichnung (2)
Watch- und Rezepturverwalter (-5)
EQ
OR
SEL
MOD
AND
CTU
objBuildDir
BuildDirectoryFB
objNewDir
ONLINE
ÜB
LESEN

```
PROGRAM PROG_FILE
VAR
    TimeState           : SR;
    WriteAOIstDaten     : FILE_Write;
    objFileCounter      : CTU;
```

Es wird nur ein Verzeichnis angelegt, wenn kein File aktiv geschrieben, gelesen, geöffnet oder geschloss
Diese erfolgt über den Status und Enable - Eingang des Bausteins.

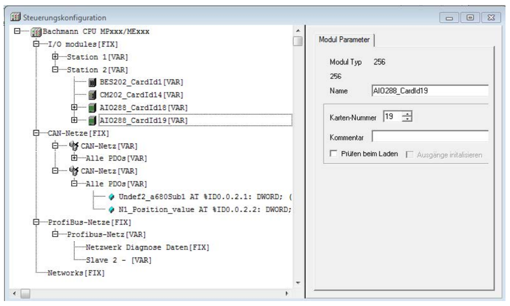
```
Lade Bibliothek 'C:\bachmann\Misw\MPLC3\Targets\Bachmann\Libraries\file_plc.lib'
Lade Bibliothek 'C:\bachmann\Misw\MPLC3\Targets\Bachmann\Libraries\cia405.lib'
Lade Bibliothek 'C:\bachmann\Misw\MPLC3\Targets\Bachmann\Libraries\ehd_plc.lib'
Lade Bibliothek 'C:\bachmann\Misw\MPLC3\Targets\Bachmann\Libraries\iecsfc.lib'
```



```
    Cycletime   : DWORD;
    gettime     : DWORD;
    Runtime_Set : DWORD := 500;
    Runtime_Act : DWORD;
    firstrun    : BOOL:= TRUE;
    CycleTime_Max   : DWORD;
END_VAR

starttime := Tcs_RTC_GetTime(0);

IF NOT firstrun THEN
    Cycletime := starttime - starttime_;
    IF cycletime > CycleTime_Max THEN
        CycleTime_Max := cycletime;
    END_IF;

    (* Only to set a breakpoint *)
    IF cycletime > 5100 THEN
        cycletime := cycletime;
    END_IF;
END_IF;

Runtime_Act := 0;
WHILE (Runtime_Set >= Runtime_Act) DO
    gettime := Tcs_RTC_GetTime(0);
    Runtime_Act := gettime - starttime; (* Überlauf wurde ni
END_WHILE;
starttime_ := starttime;
firstrun := FALSE;

(* Durchlaufzeit zwischen 400 und 800us verändern *)
IF Runtime_Set < 800 THEN
```

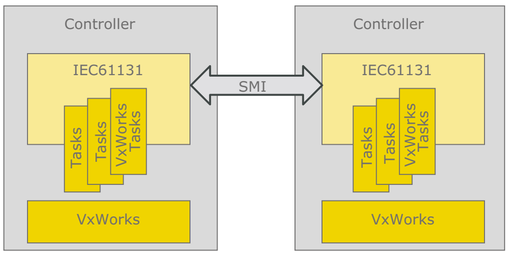
```
    ERROR: BOOL;
    firstrun: BOOL;
    START: BOOL;
```

```
    Error: BOOL;
    Start: BOOL;
    OUT: BOOL;
```



```
Bachmann CPU MPxxx/MExxx
  I/O modules[FIX]
    Station 1[VAR]
    Station 2[VAR]
      BES202_CardId1[VAR]
      CM202_CardId14[VAR]
      AIO288_CardId18[VAR]
      AIO288_CardId19[VAR]
  CAN-Netze[FIX]
    CAN-Netz[VAR]
      Alle PDOs[VAR]
    CAN-Netz[VAR]
      Alle PDOs[VAR]
        Undef2_a680Sub1 AT %ID0.0.2.1: DWORD; (
        N1_Position_value AT %ID0.0.2.2: DWORD;
  ProfiBus-Netze[FIX]
    Profibus-Netz[VAR]
      Netzwerk Diagnose Daten[FIX]
      Slave 2 - [VAR]
  Networks[FIX]
```

Modul Typ 256
256
Name AIO288_CardId19
Karten-Nummer 19
Kommentar
☐ Prüfen beim Laden ☐ Ausgänge initialisieren

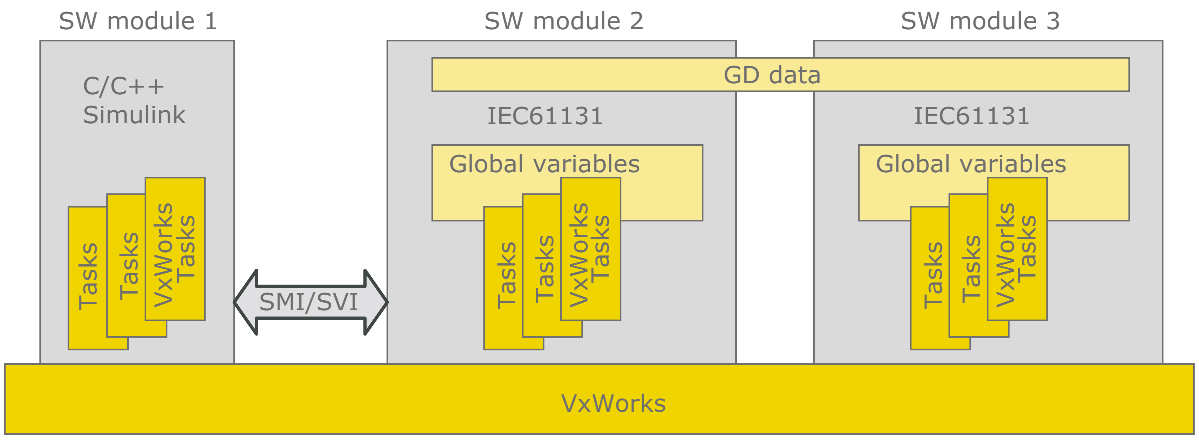


Runtime system

- Any number of running M-PLC projects on one processor[1)]
- Every PLC project supports up to 16 separate tasks
- Each task can be configured independently of the others in terms of (priority, task call-up mode [event, timetrigger, free running, sync], watchdog)
- Actual multitasking via operating system tasks (VxWorks)
- Communication between software modules (pre-compiled units)
- Between different tasks of an IEC 61131 module: global variables or SMI/SVI
- Between IEC 61131 modules: GD flag range or SMI/SVI
- Between IEC 61131 modules and other modules (C/C++, Simulink): SMI/SVI

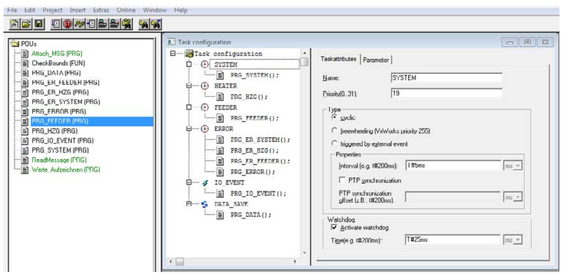
1) Maximum 256 user software modules

- Between any modules on different controllers: SMI
- Interface to visualization/SCADA/BDE
- Standard protocols: OPC DA, IEC 61850 IEC 61400, Modbus etc.
- Libraries for integration into separate software (for C/C++, C#.NET, Java)
- Support of battery-buffered data (RETAIN flag)



Online functions

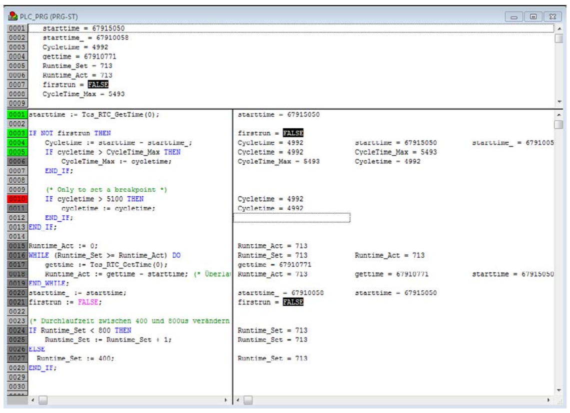
- Online change (exchange of blocks in running operation) also for multitasking projects
- Monitoring of all project variables
- Writing and forcing of variables
- Single cycle, single step and break points
- Sequence control (program lines that have been run through will be displayed)
- Recording and graphic presentation of project variables – trace



POUs
- Attach_MSG (PRG)
- CheckBounds (FUN)
- PRG_DATA (PRG)
- PRG_ER_FEEDER (PRG)
- PRG_ER_HZG (PRG)
- PRG_ER_SYSTEM (PRG)
- PRG_ERROR (PRG)
- PRG_FEEDER (PRG)
- PRG_HZG (PRG)
- PRG_IO_EVENT (PRG)
- PRG_SYSTEM (PRG)
- ReadMessage (PRG)
- Werte_Aufzeichnen (PRG)

```
Task configuration
  SYSTEM
    PRG_SYSTEM();
  HEATER
    PRG_HZG();
  FEEDER
    PRG_FEEDER();
  ERROR
    PRG_ER_SYSTEM();
    PRG_ER_HZG();
    PRG_ER_FEEDER();
    PRG_ERROR();
  IO_EVENT
    PRG_IO_EVENT();
  DATA_SAVE
    PRG_DATA();
```

Name: SYSTEM
Priority(0..31): 19
Type: [✓ cyclic] freewheeling (VxWorks priority 255) triggered by external event
Properties
Interval (e.g. t#200ms): T#5ms ms
PTP synchronization ☐
PTP synchronization offset (z.B.: t#200ms): ms
Watchdog
Activate watchdog ☑
Time(e.g. t#200ms): T#25ms ms



```
    starttime = 67915050
    starttime_ = 67910058
    Cycletime = 4992
    gettime = 67910771
    Runtime_Set = 713
    Runtime_Act = 713
    firstrun = FALSE
    CycleTime_Max = 5493
```

```
starttime := Tcs_RTC_GetTime(0);                  starttime = 67915050

IF NOT firstrun THEN                              firstrun = FALSE
    Cycletime := starttime - starttime_;          Cycletime = 4992        starttime = 67915050     starttime_ = 6791005
    IF cycletime > CycleTime_Max THEN             Cycletime = 4992        CycleTime_Max = 5493
        CycleTime_Max := cycletime;               CycleTime_Max = 5493    Cycletime = 4992
    END_IF;

    (* Only to set a breakpoint *)
    IF cycletime > 5100 THEN                      Cycletime = 4992
        cycletime := cycletime;                   Cycletime = 4992
    END_IF;
END_IF;

Runtime_Act := 0;                                 Runtime_Act = 713
WHILE (Runtime_Set >= Runtime_Act) DO             Runtime_Set = 713       Runtime_Act = 713
    gettime := Tcs_RTC_GetTime(0);                gettime = 67910771
    Runtime_Act := gettime - starttime; (* Überla Runtime_Act = 713       gettime = 67910771       starttime = 67915050
END_WHILE;
starttime_ := starttime;                          starttime_ = 67910058   starttime = 67915050
firstrun := FALSE;                                firstrun = FALSE

(* Durchlaufzeit zwischen 400 und 800us verändern
IF Runtime_Set < 800 THEN                         Runtime_Set = 713
    Runtime_Set := Runtime_Set + 1;               Runtime_Set = 713
ELSE
  Runtime_Set := 400;                             Runtime_Set = 713
END_IF;
```

Herstellerbibliotheken

• STANDARD	IEC 61131-3 Standard functions
• SMI_PLC	Functions for fast ommunication between modules and controllers (SMI/SVI)
• PLC	7 standard PID controllers
• MIO_PLC	Functions for direct access to hardware I/Os
• EHD_PLC	Functions for inputting and managing errors in the EHD
• UTIL_PLC	Functions for access to RTC, runtime system info, special conversions
• FILE_PLC	Functions for access to files, directories and serial interfaces
• CIA405	Functions for access to CAN
• DN_PLC	Functions for access to device net
• PB_PLC	Functions for access to Profibus
• USS_PLC	Functions for communication with „Micromaster“ frequency converters

Extensions relative to CoDeSys

- process image (controller configuration) runtime-optimized process image (only channels being used are processed)
- CAN objects with symbolic names in the process image
- Addressing for inputs/outputs: undelayed access that bypasses the process image
- Adjustment of the interval time and watchdog time in the controller configuration
- Multiple, concurrently running PLC projects on one CPU with different priorities / interval times
- Actual operating system tasks within the PLC projects
- GD flag range for shared data from multiple projects on one CPU
- RD flag range for remnant (battery-buffered) data
- Optionally, the normal flag range (MX .. MR) can also be placed in the battery-buffered memory area.
- Interface in the runtime system for libraries
- Initialization (C), deinitialization, version check, memory management, management of background tasks, provision for online change
- Support for exception handling in the runtime system
- Display of the status „in error handling“ (ERROR) in the interface
- Implementation of the I/O modules of the process image in the runtime system
- Optimized for performance, flexibility and combinability, shared use of an I/O module by multiple projects/SW modules possible
- Saving/loading of the entire project including sources to/from the controller
- Transparency of the global variables (flags, symbolic variables, structures) controller- and visualization-wide

Integration in the SolutionCenter (projects are SW modules)

- Generation of an executable *.m file with configuration information
- Task interval time measurement, can be queried in the project itself and in Device Manager
- Watchdog integration
- Execution possible in an application layer (memory protection)
- Install, start, stop, reset and delete of PLC projects in Device Manager
- The state machine of a PLC project in the runtime system always corresponds to the model defined for SW modules.
- Integration via M1 system debug mode and M1 system logbook
- Start of a project when booting through entry in MConfig.ini