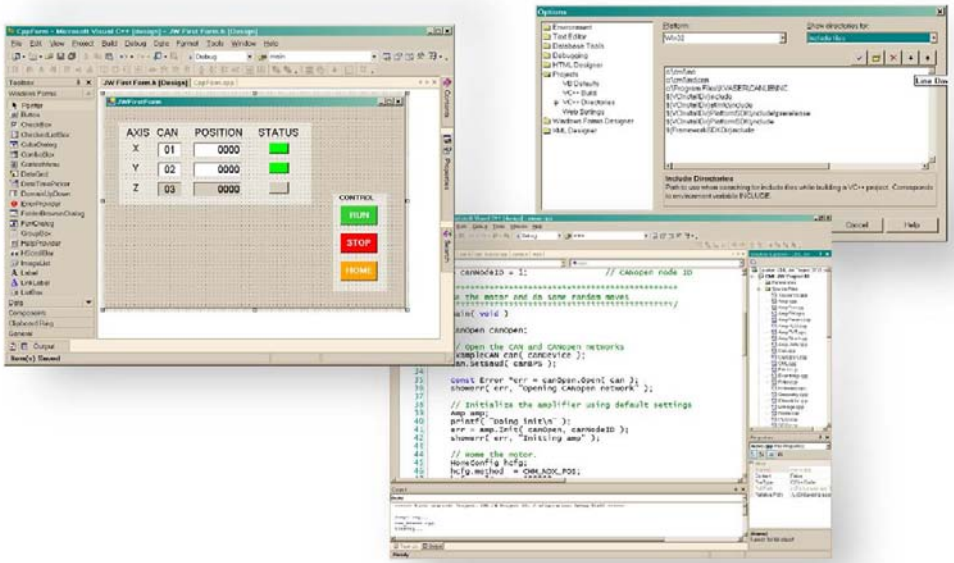
C++ Software Tools for CANopen Distributed Control



- **Automatic Network Management of CAN Bus**
- **Point-to-Point and Coordinated Motion**
- **C++ Source Code for PC's or Embedded Systems**
- **Supports CANopen Servo & Stepper Drives**

CML is a collection of object-oriented C++ classes designed to simplify the integration of intelligent CANopen servo amplifiers and stepping motor drivers into a PC based, or embedded control architecture. The development of low-level code to control the network is eliminated. Communication-card interfacing, mapping PDO's, SDO data packing, synchronization and node-guarding are taken care of automatically by a few simple commands.
The application programmer has direct access to CANopen, DS-402 compliant motion functions (Enable, Homing, Get/Set parameters, & single-axis moves). For a multi-axis control, a coordinated set can be created by the linkage class.

Complex, multi-axis moves are possible when the application program generates a sequence of points that define position, velocity, and time (PVT). The drives buffer the PVT points, perform a cubic polynomial interpolation algorithm and synchronously update commanded position to generate the path through N-dimensional space.
CML is designed to enable the C++ programmer to create motion applications on a range of operating systems and processor boards. It can be used on a PC using the Microsoft Windows or Linux operating systems. CML can also run on an embedded processor with any real-time operating system.

Description

WHAT IS CML?

CML is a collection of C++ classes that, when compiled in a C++ application, provide a programming interface to Copley Controls CANopen-enabled digital servo amplifiers and stepping motor drivers.

WHY USE CML?

CML adds CANopen functionality to C++ applications and eliminate the low-level coding to support communications over a CAN bus. In addition, CML eliminates the additional coding needed to support communication with devices operating under the CANopen protocol, the application layer that works over a CAN bus that is designed for motion control and other specialized types. CANopen devices have object-dictionaries that combine dedicated addresses for standard functions and other addresses for device-specific ones.

CML provides a high-level language interface to low-level functions that is efficient and robust. This greatly reduces development time and time-to-market. At the same time, it enables programmers to focus on their application development and to treat the CANopen interface simply as a library of objects that are ready to use.

HOW DOES IT WORK?

CML provides an object-oriented interface during program development. CML communicates with a CAN interface card via the interface driver provided by the manufacturer. At run-time, CML provides CAN bus control of Copley CANopen products by managing all of the low-level bus communications necessary to provide those services.

GENERAL SPECIFICATIONS

PRODUCT TYPE
C++ class source-code files

OPERATING SYSTEMS SUPPORTED
Microsoft Windows, Linux, or other POSIX compliant operating system

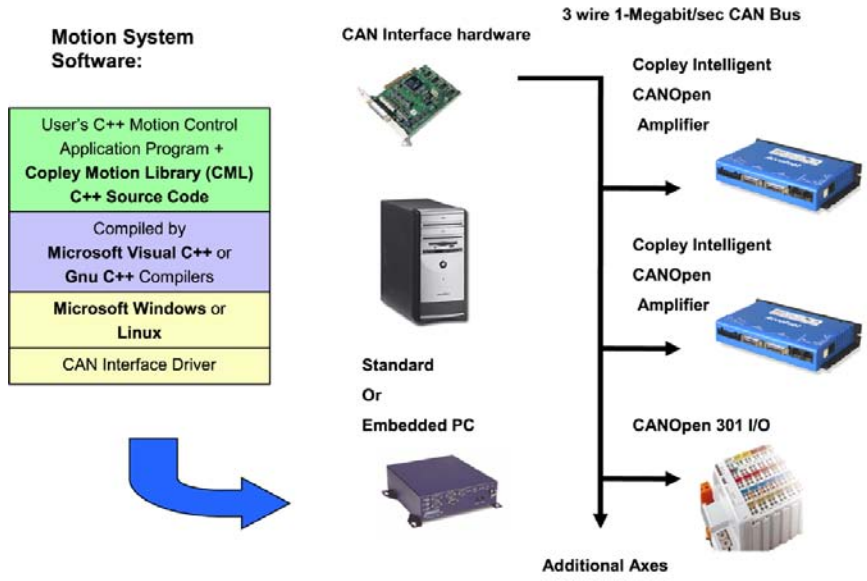
CANopen COMPLIANCE
CiA DSP-402, CANopen Device Profile for Drives & Motion Control
CiA DS-401, CANopen Device Profile for Generic I/O Modules
CiA 301, Application Layer and Communication Profile

HARDWARE REQUIREMENTS (MINIMUM)
Processor running POSIX compliant software
One CAN interface device
Copley Controls CANopen servo amplifier or stepping motor driver:
Xenus, Accelnet, Stepnet

SOFTWARE REQUIREMENTS
Any C++ compiler for use with CML. Some examples are:
Microsoft Visual C++, C++ .NET, gcc
CME 2 Version 3.2 or higher ; Copley Controls' application for amplifier setup, tuning, and configuration

CANopen HARDWARE SUPPORTED
CAN bus interface products:
Kvaser, Ixxat, Vector, National Instruments
I/O products:
Wago

CML SYSTEM CONFIGURATION



Commonly Used Classes and their Member Functions and Variables

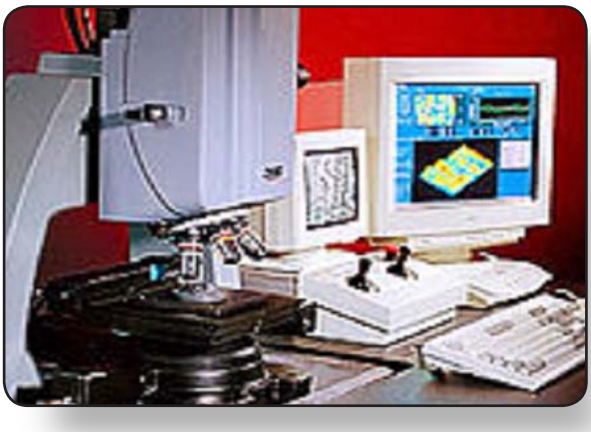
General Function	Method/Property Name	Description
CANopen Object (handles all CANopen Motion DSP-402 standard CAN communications between the PC and the amplifiers)		
CANopen	PortName	The CAN card name and port to be used for CANopen network
	BitRate	The CANopen Bit Rate to be used
	Initialize	Initialize the CANopen network commuications
Amplifier Object (handles all control and monitoring communications with an amplifier using CANopen Motion DSP-402 compatible objects)		
Amplifier Initialization	Initialize	Initialize the amplifier on the CANopen network with the appropriate CAN address and start communications
Amplifier Modes and Status Information	Disable	Disable the amplifier
	Enable	Enable the amplifier
	ClearFaults	Clear any latching faults on the amplifier
	ReadEventSticky	Read events with automatic clear
Position and Velocity	PositionActual	Read the actual encoder position
	PositionError	Read the position error (difference between position command and actual position)
Homing	GoHome	Go home executes the homing routine as specified in the home settings object
Quick Stop Support	HaltMove	Halts current move using pre-programmed halt mode
Point-to-Point Move Support	MoveAbs	Perform an absolute point-to-point move using the pre-configured profile settings
	MoveRel	Perform a relative point-to-point move using the pre-configured profile settings
Amplifier Event Processing	WaitMoveDone	Waits for the currently running move to finish, or for an error to occur
	CreateEvent	Create an event that monitors amplifier events for specific conditions
Unit Conversion Functions	CountsPerUnit	Stores a scaling factor for converting between an amplifier's default units (encoder counts) and user-defined units
Profile Settings Object (for setting up motion profile)		
Profile Settings	ProfileType	Gets/sets the profile type (SCurve, Trap, Velocity)
	ProfileVel	Gets/sets profile velocity value (velocity that the motor attempts to reach during the move)
	ProfileAcc	Gets/sets the profile acceleration value (acceleration that the motor uses when starting the move)
Linkage Object (for performing coordinated multi-axis motion)		
Linkage Object Methods	Initialize	Initializes a linkage object by assigning Amplifiers to it
	MoveTo	Multi-axis move. Target positions are passed as an array.
	WaitMoveDone	Wait until the multi axis move is complete. If the move does not complete by the time specified, the function will return
Event Object (for monitoring events from a given amplifier)		
Event Object	Start	Begins monitoring for an event to occur. Generates a single callback to a user subroutine when the chosen event occurs, or timeout has expired
	Stop	Stop monitoring
CopleyMotionLibrary Object (high level object that enables sophisticated debugging)		
Debug logging	DebugLevel	Gets/sets the debug message level. When enabled will record CAN messages from and to all the amplifiers along with actions and faults that are useful in debugging motion programs
IO Object (controls communications with a CANopen DS-401 compatible I/O module)		
Digital I/O	Din8Read, Dout8Write …	Read a group of 8 digital inputs or write a group of 8 digital outputs
Analog I/O	Ain16Read, Ain16Write …	Read a 16-bit analog input or write a 16-bit analog output
Event driven I/O	ICreateEvent	Create an event that monitors I/O events for specific conditions

CML Applications

Laboratory Automation

Screen Printing

Metrology



Textile Cutting

Conformal Coating

Pick & Place