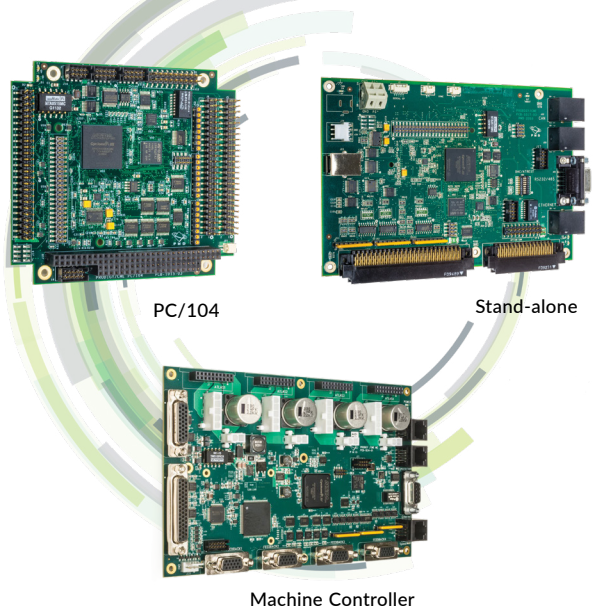
DATASHEET

Prodigy®
Motion Boards



Prodigy® Motion Boards provide high performance board-level motion control for scientific, automation, industrial, and robotic applications. Available in PC/104, standalone, and machine controller configurations, these boards support multiple motor types including Brushless DC, step, and DC Brush motors, and are available in 1, 2, 3, and 4-axis configurations.

Programmable

CME versions of the board include PMD's C-Motion Engine that allows user application code to run directly on the board, off-loading the system host or enabling stand-alone operation. The Machine controller version has on-board Atlas amplifiers that eliminate the need for external amplifier.

Powerful and Easy to Use

Based on PMD's industry-leading Magellan® Motion Control IC, the Prodigy boards provide user-selectable profile modes including S-curve, trapezoidal, velocity contouring, and electronic gearing with on-the-fly parameter change. Servo loop compensation utilizes a full 32-bit position error, PID with velocity and acceleration feedforward, integration limit and dual biquad filters for sophisticated control of complex loads.

Built on the Magellan Motion Control IC

The Pro-Motion GUI makes it easy to set-up and analyze system parameters and motion performance. PMD's C-Motion library simplifies the program development process and allows the use of industry standard C/C++ or .NET programming languages.

PERFORMANCE MOTION DEVICES
MOTION CONTROL AT ITS CORE

FEATURES

- Uses PMD's advanced Magellan® Motion Control IC
- PC/104, Stand-alone, and Machine-controller configurations
- Available in 1, 2, 3, and 4-axis configurations
- Supports Brushless DC, step, and DC Brush motors
- S-curve, trapezoidal, electronic gearing, and velocity-contouring
- PC/104 (ISA), Ethernet, CANbus or serial communications
- Advanced PID filter with feedforward and dual biquad filters
- High speed loop rate: 50 µsec/axis
- Up to 256 microsteps per full step resolution
- Incremental quadrature and Absolute SSI encoder support
- Includes Pro-Motion® and C-Motion® development software
- 6-step commutation and field oriented control modes
- High precision 16-bit DAC or PWM amplifier output
- General purpose digital I/O and analog I/O
- Two directional limit switches, plus high speed index, and home inputs per axis

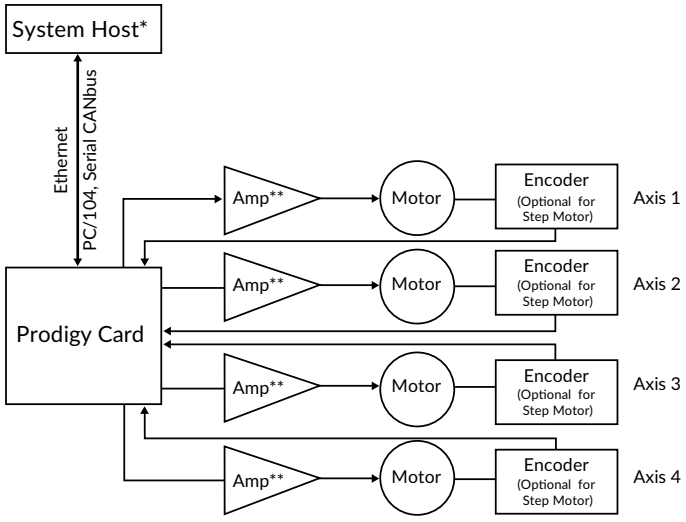
C-MOTION® ENGINE VERSIONS

- Board-level execution of C-Motion code
- Downloaded user application code runs at 96 MIPs
- C-Motion Engine development tools

MACHINE CONTROLLER VERSION

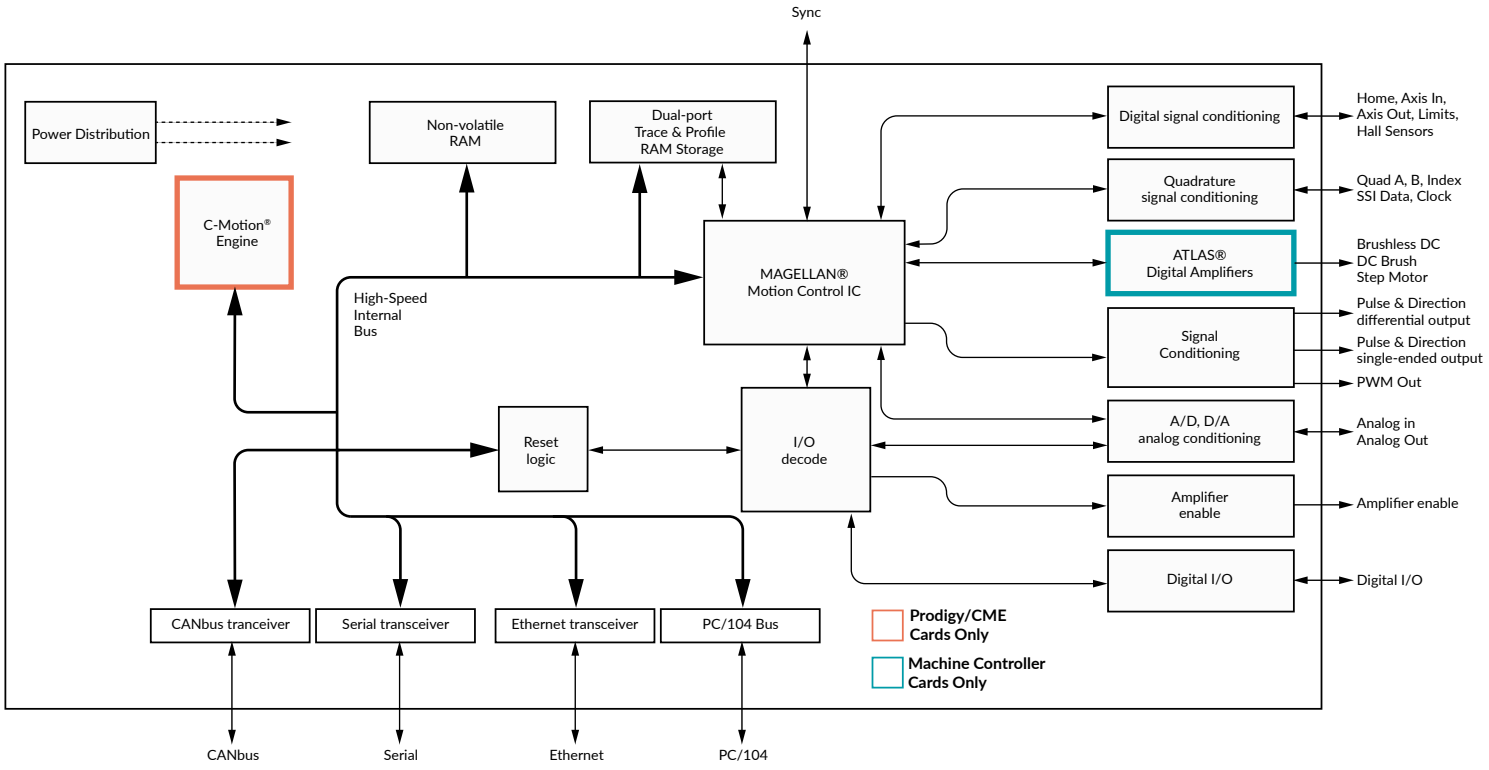
- On-board high performance Atlas amplifiers
- Extensive fault detection including over and undervoltage, motor short, and overtemp
- Up to 1KW peak output power per axis
- Single voltage supply drives motors and board logic

CONFIGURATION



*System host optional for Prodigy Programmable PC/104 and Stand-Alone cards
**External amps used with non-Machine Controller card

Technical Overview



SPECIFICATIONS

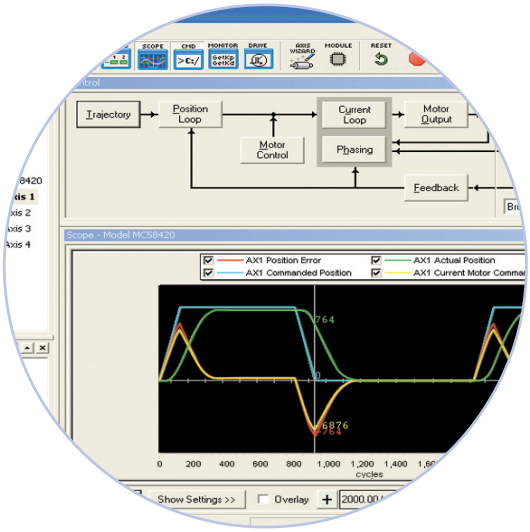
	PC/104	Stand-alone	Machine Controller
Configurations	Standard or CME	CME	CME
Model	PR82 or PR83	PR13	PR33
Number of axes supported	1, 2, 3 or 4 axes		
Supported motor types	DC Brush, Brushless DC, Step motor		
Servo loop rates	51.2 µsec to 1.6 sec. Minimum depends upon number of enabled axes and use of trace		
Encoder formats supported	quadrature, Absolute SSI		
Quadrature decode rate	8 Mcounts/sec	8 Mcounts/sec	40 Mcounts/sec
Capability for onboard amplifier	No	No	Yes, Atlas Digital Amplifier
Motor output signals	Analog ± 10V, PWM, pulse & direction	Analog ± 10V, PWM, pulse & direction	Analog ± 10V
General purpose digital I/O	8 input, 8 output	8 input, 8 output	8 bi-directional, 4 input, 4 output
General purpose analog input	8 10-bit channels (0 to 3.3V)	8 10-bit channels (0 to 3.3V)	8 16-bit channels (-10V to +10V)
General purpose analog outputs	N/A	N/A	8 16-bit channels (-10V to +10V)
Limit switches	2 per axis: one for each direction of travel		
CME version user program memory	256 KB Flash / 8 KB RAM		
CME version stack memory	8 KB RAM		
Dual ported RAM memory	40KB (standard), 64KB (CME)	64KB	128K or 468K (enhanced memory option)
Communication modes	Standard: PC104 bus, serial, CANbus CME: PC104 bus, serial, CANbus, Ethernet	serial, CANbus, Ethernet	serial, CANbus, Ethernet
On-board amplifier voltage range	N/A	N/A	12-56V
On-board amplifier max current, continuous	N/A	N/A	Brushless DC Motor: 10 Arms, Step motor: 9 Arms, DC Brush Motor: 14 ADC
Dimensions	4.35" x 3.78" x 0.6" (11.1cm x 9.6cm x 1.5cm)	6.30" x 4.23" x .8" (16.0cm x 10.7cm x 2.0cm)	7.80'' x 4.88'' x .78" (19.8cm x 12.4cm x 1.98cm)

Development Tools

1 EASY START-UP
Developers Kit

INCLUDES

- Prodigy Developer Kits
- Pro-Motion software
- Software Development Kit (SDK) with C-Motion
- Complete manual set
- Complete cable and prototyping connector set



2 TUNE & OPTIMIZE
Pro-Motion GUI®

Pro-Motion is a sophisticated, easy-to-use Windows-based exerciser program for use with PMD motion control ICs, modules, and boards.

FEATURES

- Motion oscilloscope graphically displays processor parameters in real-time
- Autotuning
- Ability to save and load settings
- Axis wizard
- Distance and time units conversion
- Motor-specific parameter setup
- Axis shuttle performs programmable motion between two positions
- Communications monitor echoes all commands sent by Pro-Motion to the board
- Advanced Bode analysis for frequency machine response

3 BUILD THE APP
C-Motion®

C-Motion is a complete, easy-to-use, motion programming language that includes a source library containing all the code required for communicating with PMD motion ICs, board, and modules.

C-MOTION FEATURES INCLUDE:

- Extensive library of commands for virtually all motion design needs
- Develop embeddable C/C++ applications
- Complete, functional examples
- Supports PC/104, serial, CAN, Ethernet, and SPI communications

```
code for executing a profile and tracing
captured in this example could be used for tuning the PID
race buffer wrap mode to a one time trace
aceMode(hAxis1, PMDTraceOneTime);
et the processor variables that we want to capture
tTraceVariable(hAxis1, PMDTraceVariable1, PMDAxis1,
etTraceVariable(hAxis1, PMDTraceVariable2, PMDAxis1,
SetTraceVariable(hAxis1, PMDTraceVariable3, PMDAxis1, P
// set the trace to begin when we issue the next update command
SetTraceStart(hAxis1, PMDTraceConditionNextUpdate);
// set the trace to stop when the MotionComplete event occurs
SetTraceStop(hAxis1, PMDTraceConditionEventStatus,
  PMDEventMotionCompleteBit, PMDTraceStateHigh);
SetProfileMode(hAxis1, PMDTrapezoidalProfile);
set the profile parameters
tPosition(hAxis1, 200000);
Velocity(hAxis1, 0x200000);
celeration(hAxis1, 0x1000);
leration(hAxis1, 0x1000);
ion
```